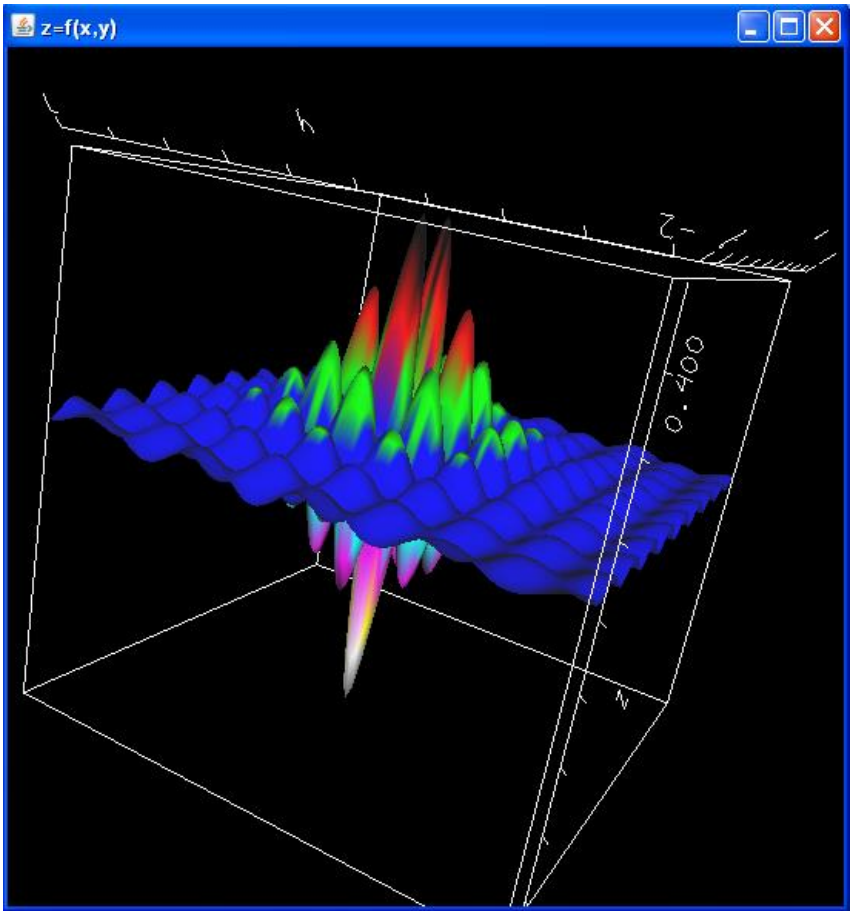
JAVA PROGRAMLAMA DİLİ ÖRNEKLERİ İLE SAYISAL ÇÖZÜMLEME



Dr. M. Turhan ÇOBAN

EGE ÜNİVERSİTESİ

Mühendislik Fakultesi

Makine Mühendisliği

Bornova, İZMİR

JAVA PROGRAMLAMA DİLİ ÖRNEKLERİYLE

SAYISAL ÇÖZÜMLEME

Dr. M. Turhan ÇOBAN

EGE ÜNİVERSİTESİ

Mühendislik Fakultesi

Makine Mühendisliği

Bornova, İZMİR

İçindekiler

BÖLÜM-1 Giriş Kavramları

1.1 Java Programlama Dilinin Sayısal Analiz Temel Dili Olarak Kullanımı
1.2 Abstract fonksiyon, interface ve Mathd fonksiyon kütüphanesi
1.3 İki Boyutlu Veri ve fonksiyonların Grafiklerin Çizilmesi
1.4 Üç boyutlu Veri ve fonksiyonların Grafiklerin Çizilmesi
1.5 Gerçek Zaman Grafik Ortamı
1.6 Grafik ortamından veri okuma
1.7 Taylor serileri ve hata terimi
1.8 Çeşitli fonksiyonların seri açılımları

BÖLÜM-2 Fonksiyonların kökleri

2.1 İkiye Bölme Yöntemi (Bisection)
2.2 Yer değiştirme yöntemi (false position)
2.3 Müller Yöntemi
2.4 Newton-Raphson Yöntemi
2.5 Kiriş Yöntemi
2.6 Aitken Extrapolasyon Düzeltmeli Newton-Raphson Yöntemi
2.7 Aitken Ekstrapolasyon Düzeltmeli Direk Yerine Koyma : Steffensen metodu
2.8 İkinci ve daha üst türev dereceli Yöntemler
2.9 Newton-Raphson ve ikiye bölme metodlarının birlikte kullanılması
2.10 İkinci derece ters Lagrange interpolasyon yöntemi
2.11 Brent kök bulma yöntemi
2.12 Kök sınırlarının saptanması
2.13 Ridder metodu
2.14 Üçüncü dereceden polinomun kökleri
2.15 Le Guerre metodu ile polinomların kompleks köklerini bulma
2.16 Horner metoduyla polinomların kompleks köklerinin bulunması
2.17 Bairstow metoduyla polinomların kompleks köklerinin bulunması
2.18 Problemler

BÖLÜM-3 Doğrusal Denklem Sistemleri

3.1 Doğrusal denklem sistemleri, genel
3.2 Gauss eleme metodu
3.3 Kısmi pivot işlemi, kısmi pivotlu gauss eleme metodu
3.4 Gauss-Jordan metodu
3.5 Kısmi pivotlu Gauss-Jordan metodu
3.6 LU-alt-üst üçgen matris- Gauss (Dolittle) metodu
3.7 Craut LU metodu
3.8 Jacobi, Gauss-Seidel ve rahatlamalı interpolasyon iteratif metodları
3.9 Cholesky simetrik matris LU metodu
3.10 3'lü ve 5'li bant matris
3.11 Konjuge gradyan metodu
3.12 Matris temel işlemleri toplama ve çarpma tanımları
3.13 Norm tanımı
3.14 Problemler

BÖLÜM-4 Doğrusal olmayan denklem Sistemlerinin kökleri

4.1 Newton - Raphson yöntemi
4.2 Sherman-Morrison formüllü Broyden kiriş yöntemi

4.3 Süreklilik yöntemi
4.4 Optimizasyon yöntemleri : Nelder-Mead amip yöntemi
4.5 Optimizasyon yöntemleri : Genetik algoritma yöntemi
4.6 Optimizasyon yöntemleri : en dik yamaç metodu
4.7 Optimizasyon yöntemleri : Davidon-Fletcher-Powell metodu
4.8 Jacobi, Gauss-Seidel ve rahatlamalı interpolasyon iteratif metodları
4.9 Problemler

BÖLÜM-5 Optimizasyon(Minimizasyon ve Maksimizasyon)

5.0 Optimizasyonun tanımlanması ve grafik yöntemle tek boyutlu optimizasyon
5.1 Bir bilinmiyenli-lineer olmayan-geometrik-altın oran (Fibonacchi) arama
5.3 Bir bilinmiyenli-lineer olmayan-geometrik-ikinci derece polinom metodları
5.4 Bir bilinmiyenli-lineer olmayan kübik polinom arama yöntemi
5.5 Bir bilinmiyenli-lineer olmayan-geometrik-newton-raphson metodu
5.6 Bir bilinmiyenli-lineer olmayan-geometrik brent metodu
5.7 Bir bilinmiyenli-lineer olmayan-geometrik-türevli brent metodu
5.8 Optimizasyon sınır tahmin değerlerinin iyileştirilmesi
5.9 Çok bilinmiyenli-lineer olmayan-geometrik-newton-raphson metodu
5.10 Çok bilinmiyenli-lineer olmayan-geometrik-Nelder ve Mead simpleks metodu
5.11 Çok bilinmiyenli-lineer olmayan-geometrik-en dik yamaç metodu
5.12 Çok bilinmiyenli-lineer olmayan-geometrik- değiştirilmiş Newton metodu
5.13 Çok bilinmiyenli-lineer olmayan- geometrik Fletcher - Reeves ve Polak - Ribiere metodları (konjuge gradyen metodu)
5.14 Çok bilinmiyenli-lineer olmayan-geometrik davidon-fletcher-powell metodu
5.15 Çok bilinmiyenli-lineer olmayan-geometrik direk hessian değiştirme metodu : Broyden-Fletcher-Goldberg-Shanno (BFGS)
5.16 Çok bilinmiyenli-lineer olmayan-geometrik süreklilik metodu
5.17 Çok bilinmiyenli-lineer olmayan-geometrik Sherman Morrison formüllü Broyden kiriş metodu ile optimizasyon
5.18 Global tesadüfi metodlar: Genetik Algoritmalar
5.19 Global tesadüfi metodlar:Monte-Carlo optimizasyonu
5.20 Global tesadüfi metodlar: iterasyonlu tepe tırmanma
5.21 Global tesadüfi metodlar: tavlama simulasyonu (simulated annealing)
5.22 Global tesadüfi metodlar : Ateşböceği algoritması
5.23 Global tesadüfi metodlar : sürü elemanları algoritması
5.24 Global tesadüfi metodlar : Yarasa simülasyonu
5.25 Global tesadüfi metodlar : Ahenk ve Melodi arama metodları
5.26 Global tesadüfi metodlar : Guguk kuşu arama metodu
5.27 Global tesadüfi metodlar : av arama metodu
5.28 Global tesadüfi metodlar : yerçekimi arama metodu
5.29 Global tesadüfi metodlar: Büyük Patlama – Büyük Kırılma algoritması
5.30 Global tesadüfi metodlar: Arı kovanı optimizasyonu
5.31 Global tesadüfi metodlar: Quantum davranışlı parçacok sürü optimizasyonu
5.32 Global tesadüfi metodlar: Diferansiyel arama algoritması
5.33 Global tesadüfi metodlar: Hibrit Ahenk Arama+Yapay arı kolonisi algoritması
5.34 Sınırlamalı lineer olmayan optimizasyon
5.35 Gezgin satıcı problemi
5.36 Lineer optimizasyon, simpleks metodu
5.36 Problemler

BÖLÜM-6 Eğri uydurma ve aradeğer hesabı

6.0 Giriş
6.1 Polinom en küçük kareler metodu
6.2 Ortogonal polinom en küçük kareler metodu

6.3 Genelleştirilmiş en küçük kareler metodu
6.4 Genel en küçük kareler metodu ile birden fazla değişkenli fonksiyonların eğri uydurması
6.5 Newton interpolasyonu
6.6 Lagrange interpolasyonu
6.7 Hermit interpolasyonu
6.8 Vandermonde interpolasyonu
6.9 Kübik şerit interpolasyonu
6.10 B şerit interpolasyonu
6.11 Lineer olmayan katsayılı bir ve çok boyutlu bağımsız değişkenli fonksiyonlara eğri uydurma
6.12 Eğri uydurmada indirek (parametrik) değişken kullanımı
6.13 Hareketli en küçük kareler metodu
6.14 Benstein polinomları ve bezier eğrileri
6.15 Kısmi devamlı fonksiyonlar kullanarak en küçük kareler metodu ile eğri uydurma
6.16 Bölümlü fonksiyon yaklaşımları : Pade yaklaşımı
6.17 Bölümlü fonksiyon yaklaşımları : Chebyschev yaklaşım metodu
6.18 Trigonometrik fonksiyon (Fourier) yaklaşımı
6.19 Legendre polinomlarıyla fonksiyon yaklaşımı
6.20 Sonlu Elemanlar(finite element) metodu ile yüzey uydurma
6.21 Problemler

BÖLÜM-7 İntegral ve türev

7.1 Newton-cotes formulleri : trapez(yamuk), Bole, Simpson 1/3 ve 3/8 metodları
7.2 Richardson extrapolasyonu (Romberg integrasyonu) ve Aitken extrapolasyonu
7.3 Gauss-Legendre, Gauss-Chebchev, Gauss-Jacobi, Gauss-Hermit, Gauss-Lequerre integral formülleri
7.4 Ayarlanabilir hata miktarlı integrasyon formülleri
7.5 Clenshaw-curtis integral formülü
7.6 Tek ve çok boyutlu Monte-Carlo integrali
7.7 Çok boyutlu integralleri parametrik metodla çözme
7.8 Verilerin direk olarak integrali
7.9 Türev formülleri
7.10 Türev işleminde Richardson extrapolasyonu
7.11 Türev yaklaşımında Pad'e formülü
7.12 Verilerin direk olarak türevinin alınması
7.13 Problemler

BÖLÜM-8 Diferansiyel denklem sistemleri

8.1 Euler denklemi
8.2 Heun denklemi
8.3 Orta nokta veya geliştirilmiş poligon denklemi
8.4 Runge-Kutta denklemleri
8.5 Çok adımlı yöntemler
8.6 Değişken adım boyutlu yöntemler
8.7 Birden fazla değişkenli denklemler
8.8 Diferansiyel denklemlerde başlangıç noktası, atış problemi
8.9 Problemler

BÖLÜM-9 Özdeğer - sınır değer problemleri

9.0 Giriş
9.1 En büyük sınır değer (kuvvet metodu)
9.2 En küçük sınır değer(kuvvet metodu)
9.3 Jacobi metodu ve Householder dönüşümü
9.4 QL algoritması

9.5 QR algoritması
9.5 Polinomların kökleri ve refakatçi matrisi
9.6 Tekil değer ayrıştırması (Singular value decompositıon –SVD) Lineer denklem sistemlerinin çözümü
9.7 Tekil değer ayrıştırması kullanarak polinom en küçük kareler metodu eğri uydurma
9.8 Problemler

BÖLÜM 10 Sonlu farklar Metoduyla kısmi diferansiyel denklemlerin çözülmesi

BÖLÜM 11 Sonlu elemanlar Metoduyla kısmi diferansiyel denklemlerin çözülmesi

BÖLÜM 12 Referanslar

Önsöz

Bir mühendis olarak daima bu problemi ne şekilde çözeceğim sorusuyla iç içe yaşamışızdır. Bu gereksinme beni sayısal yöntemlerle yakından uğraşmaya itti. Sayısal yöntemleri daha iyi anladıkça da mühendislik problemlerimi çözme yetim arttı. Sayısal matematik yapısı gereği bilgisayar ortamında işlenerek oluşturulan ve kullanılan metodlardır, bu yüzden temel olarak iyi bir bilgisayar programlama altyapısıyla desteklenmesi gerekir. Bu kitapla bu iki konudaki altyapımı birleştirerek bilimle uğraşan tüm arkadaşlarımıza aktarmaya çalışacağım.

Bu kitabı ne şekilde adlandıracağımdan tam olarak emin değilim, Sayısal yöntemlerin bilgisayara uygulanmasını anlatan bir bilgisayar kitabı veya bilgisayar kodlarıyla desteklenmiş bir sayısal çözümleme kitabı diyebiliriz. Bu tür kitaplar Matlab gibi popüler hazır sayısal çözümleme kodlarını kullanarak daha önce de mevcuttu. Ancak Matlab gibi kodlar oldukça pahalı ve ülkemiz ortamında kullanılmaları maddi açıdan her zaman için (en azından resmi yollarla satın alınarak) mümkün olamıyabiliyor. Benim bu kitabı geliştirirken temel amacım hem açık kodlar üzerinden dili geliştirirken hem de sayısal metodların daha iyi anlaşılabilmesini sağlamak, hem de matlab gibi bazı kodların sıfırdan başlamadan kallanıcılarımızın elinde hazır olarak bulunabilmesinin sağlanması. Kitapa bulacağınız programları geliştirmek için Java dilinden yararlandık, bu seçimdeki temel gayemiz Java'nın platform bağımsız bir dil olması ve aynı zamanda internet ortamında direk olarak çalıştırılabilmesidir.

Kitapla birlikte sunulan kodlar, 2 ve üç boyutlu grafiklerin çizilebilmesini sağlayacak böylece sayısal çıktıları grafik olarak irdeleyebileceğimiz 2 ve 3 boyutlu grafik kütüphanelerine de sahiptir. 2 boyutlu grafik sistemi benim tarafımdan geliştirildi. 3 boyutlu grafik sistemi için VisAD isimli açık bir kod bulduğumdan sadece kendi fonksiyonlarımın bu ortamda çalışması için ara program hazırlamakla yetindim. Ayrıca giriş ve çıkış proseslerinde yine benim tarafımdan geliştirilmiş Text sınıfı tüm programlarda yoğun olarak kullanılmıştır. Buradaki gayemiz Java giriş çıkış terimlerini çok bilmesek de yeterince basit bir yolla karmaşık giriş çıkış işlemlerini gerçekleştirebilmekti. Sayısal çözümleme kodları, temelde Ege Üniversitesi Mühendislik fakultesi makine mühendisliği bölümünde verdiğim lisans seviyesi Sayısal çözümleme ve yüksek lisans seviyesi optimizasyon ve yine yüksek lisans seviyesi mühendisler için iler sayısal metodlar dersleri içeriğinde geliştirilmiş, ancak bu derslerde işlemediğimiz ek kodlar ve konularla zenginleştirilmiştir. Buradaki bazı programların kökeni bu derslere göre çok daha gerilere gitmektedir, hatta bir kısmını daha java dili çıkmadan önce c++ dilinde geliştirdiğim programlardan java diline direk olarak tercüme ettim, iki dil yapı olarak aynı dil yapısını kullandığından bu göreceli olarak basit bir işlemdi.

Kitap şu şekliyle seviye olarak standart bir lisans seviyesi kursu için kullanılacağı gibi başlangıç yüksek lisans seviyesine de hitap edebilecek seviyededir. Mühendis ve bilimle uğraşan arkadaşlarımız için aynı zamanda bir referans ve gereksimelerini karşılayacak bir kaynak da olabilir. Elbette sayısal matematik gibi kendi başına bir bilim dalı olan bir alanda geliştirilmiş tüm yöntemleri bir kitapta sunmamız olasılı değildir, zaman içinde ihtiyaçlara göre kapsamı ve kavramı geliştirebiliriz. Şu ana kadarki kısım iki yıllık bir çabanın ürünü.

Benim kişisel olarak sayısal yöntemlerle ilgili çalışmalarım mühendislik uygulamalarını daha aktif olarak oluşturabilmek amacıyla idi. Bu kitabın bu konuda tüm kullanıcılarına yardımcı olacağına inanıyorum. Bazı örnekler ve problemler mühendislik problemlerinden seçilmiştir, ancak dikkatli okunduğunda formüller ve değerleri verildiğinden bu problemler rahatlıkla saf matematik formüllerine dönüşür, bu yüzden kitap sadece mühendislere değil sayısal metodlar kullanmak isteyen tüm bilim insanlarına hitap etmektedir. Kitaptaki tüm kodlar, kitapla birikte açık olarak sunulacaktır, buradaki gayelerimizden birisi de sayısal çözümlemeler için sizin daha etkin kodlar üretmeniz için bir başlangıç noktası oluşturabilmektir.

Saygılarımla,

Dr. M. Turhan ÇOBAN

İzmir, 15-04-2008

Özgeçmiş

Mustafa Turhan Çoban, 1957 yılında Bolu, Sebende doğdu. 1978 yılında Ege Üniversitesi Makine Fakultesi'nden makine mühendisi olarak mezun oldu. 1982 yılında Michigan Technological University(A.B:D.), Makina mühendisliği ve mühendislik mekaniği bölümünden makine mühendisliği yüksek lisans derecesi ile mezun oldu. 1986'da Utah üniversitesi(A.B.D.), Mühendislik fakultesi, makine mühendisliği bölümünden makine mühendisliği doktora derecesini aldı. 1995 yılında Victoria Technological University(Avustralya), matematik ve bilgisayar bilimleri bilgisayar mühendisliği yüksek lisans derecesi aldı. Ballarat University(Avustralya), University of Nebraska, Lincoln(A.B.D.), Victoria Technological University(Avustralya), Dokuz Eylül Üniversitesi, Mühendislik Fakultesi, makine bölümü, Gebze Yüksek Teknoloji Enstitüsü, Bilgisayar Mühendisliği bölümü, Ege Üniversitesi, Mühendislik fakultesi makine müh. bölümlerinde çalıştı ve çeşitli dersler verdi. Aras kompresörleri, maden tetkik arama enstitüsü, sondaj dairesi, jeotermal bölümü, Tübitak, marmara Araştıma merkezi, enerji bölümü, Tübitak ulusal metroloji enstitüsü, Ceramic Fuel Cells Limited(Avustralya), Imperial Chemical Industries(Avustralya) şirketlerinde çalıştı.

BÖLÜM-1 Giriş Kavramları

1.1 Java Programlama Dilinin Sayısal Analiz Temel Dili Olarak Kullanımı

Bu kitaptaki eksersizler ve örnek problemler temel olarak java programlama dilinde verilmiştir. Bu dilin seçilmesinin temel sebebi modern bir programlama dili olmasının yanı sıra zengin grafik ortamı, internette kullanılabilirliği, sistem bağımsız yapısı ve serbest kullanım olasılığıdır. Burada Java dilinin detayına girmeyeceğiz. Bu konuda yeterli kaynak bulunmaktadır. Kitabın ilk konularında, daha kodlar çok detaylı hale gelmeden önce Matlab (Octave), MS Excel gibi dillerde de örnek problemlere yer verildi. Bunun bir nedeni dillerin ne kadar benzer olduklarını kullanıcıya göstermektir, Üçüncü kuşak bilgisayar dillerinden herhangi birini bilen bir kullanıcı diğerlerine rahatça adapte olabilir. Sayısal çözümleme temel olarak bilgisayar ortamında kullandığımız matematik olduğundan ve Bilgisayar ortamının günümüzdeki yayılmasını göz önüne alırsak okuyucularımızın bilgisayar dili konusunda da kendilerini geliştirmelerinin elzem olduğunu düşünüyoruz.

1.2 Abstract fonksiyon, Interface ve Mathd fonksiyon kütüphanesi

Sayısal çözümlemede ve genel matematikte fonksiyonlar sıkça kullandığımız kavramlardır. Bir çok kodumuz fonksiyonlar üzerinden tanımlanır. Bu kitaptaki kodlar verilirken fonksiyonlardan bağımsız olarak kodların tanımlanması için java'daki abstract class (soyut sınıf) kavramı kullanıldı. Örnek verecek olursak :

Program 1.2.1 abstract sınıf örneği f_x

```
import static java.lang.Math.*;

abstract  class f_x extends Mathd
{
  // single function single independent variable
  // example f=x*x
  // Reference :"Generation of Finite Difference Formulas on Arbitrary Spaced Grids",
  // Bength Fornberg, Mathematics of Computation, Volume 51, Number 184, October 1988
  // pages 699-706

// abstract function func
abstract public double func(double x);

// 1st order derivative

public double dfunc(double x)
{double h=1.0e-3;
 int n=1;
 int M=30;
 return dfunc(x,n,M,h);
}
// nth order derivative

double dfunc(double x,int N,int Mi,double hi)
{// order of the maximum derivative
 // N order of derivative
 // M degree of difference formula
double c[][][];
double alpha[];
double h;
int M;
double a[]=new double[0];
h=0.01;
double x0=0;
```

```
M=20;
//central difference
double alphai[]={0,1,-1,2,-2,3,-3,4,-4,5,-5,6,-6,7,-7,8,-8,9,-9,10,-10,11,-11,12,-12,13,-13,14,-14,15,
-15,16,-16,17,-17,18,-18,19,-19,20,-20,21,-21,22,-22,23,-23,24,-24,25,-25,26,-26,27,-27,28,-28,29,-29,30,-30,
31,-31,32,-32,33,-33,34,-34,35,-35,36,-36,37,-37,38,-38,39,-39,40,-40,41,-41,42,-42,43,-43,44,-44,45,-45,46,
-46,47,-47,48,-48,49,-49,50,-50,51,-51,52,-52,53,-53,54,-54,55,-55,56,-56,57,-57,58,-58,59,-59,60,-60,
-61,61,-62,62,-63,63,-64,64,-65,65,-66,66,-67,67,-68,68,-70,70,-71,71,-72,72,-73,73,-74,74,-75,75,
-76,76,-77,77,-78,78,-79,79,-80,80,-81,81,-82,82,-83,83,-84,84,-85,85,-86,86,-87,87,
-88,88,-89,89,-90,90,-91,91,-92,92,-93,93,-94,94,-95,95,-96,96,-97,97,-98,98,-99,99,-100,100};
alpha=alphai;
int N1=alpha.length-1;
// M   degree of highest derivative
// N1+1 number of coefficients
double delta[][][]=new double[N1+1][N1+1][M+1];
double c1,c2,c3;
delta[0][0][0]=1.0;
c1=1.0;
for(int n=1;n<=N1;n++)
{  c2=1;
   for(int nu=0;nu<=(n-1);nu++)
   {c3=alpha[n]-alpha[nu];
    c2=c2*c3;
    if(n<=M) delta[n-1][nu][n]=0.0;
    for(int m=0;m<=Math.min(n,M);m++)
    {
       if(m==0)
       { delta[n][nu][m]=((alpha[n]-x0)*delta[n-1][nu][m])/c3;}
       else
       {delta[n][nu][m]=((alpha[n]-x0)*delta[n-1][nu][m]-m*delta[n-1][nu][m-1])/c3;}
    }//next m
   }//next nu
   for(int m=0;m<=Math.min(n,M);m++)
   {   if(m==0)
                {delta[n][n][m]=c1/c2*(-(alpha[n-1]-x0)*delta[n-1][n-1][m]);}
                else
                {delta[n][n][m]=c1/c2*(m*delta[n-1][n-1][m-1]-(alpha[n-1]-x0)*delta[n-1][n-1][m]);}
   }//next m
   c1=c2;
}//next n
c=delta;
  if(Mi<N) M=N;
  else M=Mi;
  h=hi;
  double deriv=0;
  double h1=1/h;
  double h2=1;
  for(int j=0;j<N;j++)
  {h2*=h1;}
  for(int i=0;i<c[0].length;i++)
  {   deriv+=c[M][i][N]*func(x+alpha[i]*h);}
  return deriv*h2;
}

public double dfunc(double x,int N)
{int M=30;
 double h=0.05*N;
 return dfunc(x,N,M,h);
}}
```

Görüldüğü gibi örnek abstract sınıfımız genel bir y= f(x) fonksiyonunu tanımlamaktadır. Burada Abstract sınıf Mathd sınıfından türetilmiştir. Bunun anlamı bu sınıfta bulunan tüm metod ve değişkenler bu abstract sınıfa, dolayısıyla bu sınıfın kullanıldığı diğer programlara aktarılmış olacaktır. Java'da matematik fonksiyonları içeren Math sınıfı bulumkatadır. Mathd bu kitap ve diğer uygulamalar için geliştirilen içeriğinde Bessel, gama fonksiyonları gibi daha geniş bilimsel fonksiyonların tanımlandığı bir kütüphane olarak tanımlayabiliriz. Kitabın sayfa içeriğinin çok büyümemesi için bu tür ana kodları www.turhancoban.com sitesindeki kitaba ait sitede yayınlayacağız , burada ayrıca

listelenmeyecektir. Abstract sınıfların Mathd sınıfından türetilmesinin bir avantajı da Math ve Mathd kütüphane fonksiyon adlarının direk olarak kullanımını sağlamasıdır. Örneğin java'da sinüs fonksiyonunu çağırmak için Mathd.sin(x) tanımını kullanmamız gerekirken abstract sınıftan türettiğimiz fonksiyonlarda sadece sin(x) olarak tanımlayabiliriz. Fonksiyonu kullanırken yeni bir sınıfı oluşturulur ve bu sınıf abstract sınıftan türetilir.

Program 1.2.2 abstract sınıf f_x ten türetilen 2.*sin(x)*cos(x) fonksiyonu

```
class f1 extends f_x
{  public double func(double x)
  {return 2.0*sin(x)*cos(x);}
}
```

Bu fonksiyon tanımı herhangi bir programda kullanılabilir.

Program 1.2.3 abstract sınıf f_x ten türetilen 2.*sin(x)*cos(x) fonksiyonunu kullanan bir integral alma programı

```
 public class f_xtest {
//INTEGRAL
public  static double integral(f_x f,double a,double b)
{ //10 point Gauss-Legendre formula
  //integral f(x)dx
 double r[]={-0.973906528517171,-0.865063366688984,-0.679409568299024,-0.433395394129247,-0.148874338981631,
 0.148874338981631,0.433395394129247,0.679409568299024,0.865063366688984,0.973906528517171};
 double c[]={0.066671344308684,0.149451349150580,0.219086362515982,0.269266719309996,0.295524224714752,
 0.295524224714752,0.269266719309996,0.219086362515982,0.149451349150580,0.066671344308684};
double z=0,x,y;
double k1=(b-a)/2.0;
double k2=(b+a)/2.0;
for(int i=0;i<r.length;i++)
{
x=k2+k1*r[i];
y=f.func(x);
z+=k1*c[i]*y;
}
return z;
}

public static void main(String arg[])
{ f1 ff=new f1();
  System.out.println(integral(ff,0.0,1.0));
}
}
```

Kullandığımız Java programlama dilinde Java 8'den itibaren Interface fonksiyonların program içinde kısa tanımları olan lambda terimlerinin kullanımı gerçekleştirilmiştir. f_x abstract sınıfının benzeri inter_f_x ainterface'ini tanımlayacak olursak:

Program 1.2.4 interface if_x programı

```
import static java.lang.Math.*;
import java.io.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.*;
  // single function single independent variable
  // example f=x*x
  // includes full set of derivatives
  // Reference :"Generation of Finite Difference Formulas on Arbitrary Spaced Grids",
  // Bength Fornberg, Mathematics of Computation, Volume 51, Number 184, October 1988
  // pages 699-706
  //interface version
@FunctionalInterface
interface if_x
{
public double func(double x);

default void Plot(double a,double b)
```

```
{   Plot pp=new Plot(this,a,b);
            pp.plot();
}
default void vplot(double a,double b,int n)
{   vplot.plot(this,a,b,n);}
default void vplot(double a,double b)
{   vplot.plot(this,a,b,400);}
default void Plot(double a,double b,int n)
{   Plot pp=new Plot(this,a,b,n);
            pp.plot();
}
// nth order derivative

default double dfunc(double x,int N,int Mi,double hi)
{// order of the maximum derivative
 // N order of derivative
 // M degree of difference formula
double c[][][];
double alpha[];
double h;
int M;
double a[]=new double[0];
h=0.01;
double x0=0;
M=20;
double alphai[]={0,1,-1,2,-2,3,-3,4,-4,5,-5,6,-6,7,-7,8,-8,9,-9,10,-10,11,-11,12,-12,13,-13,14,-14,15,
-15,16,-16,17,-17,18,-18,19,-19,20,-20,21,-21,22,-22,23,-23,24,-24,25,-25,26,-26,27,-27,28,-28,29,-29,30,-30,
31,-31,32,-32,33,-33,34,-34,35,-35,36,-36,37,-37,38,-38,39,-39,40,-40,41,-41,42,-42,43,-43,44,-44,45,-45,46,
-46,47,-47,48,-48,49,-49,50,-50,51,-51,52,-52,53,-53,54,-54,55,-55,56,-56,57,-57,58,-58,59,-59,60,-60,
-61,61,-62,62,-63,63,-64,64,-65,65,-66,66,-67,67,-68,68,-70,70,-71,71,-72,72,-73,73,-74,74,-75,75,
-76,76,-77,77,-78,78,-79,79,-80,80,-81,81,-82,82,-83,83,-84,84,-85,85,-86,86,-87,87,
-88,88,-89,89,-90,90,-91,91,-92,92,-93,93,-94,94,-95,95,-96,96,-97,97,-98,98,-99,99,-100,100};
alpha=alphai;
int N1=alpha.length-1;
// M   degree of highest derivative
// N+1 number of coefficients
double delta[][][]=new double[N1+1][N1+1][M+1];
double c1,c2,c3;
delta[0][0][0]=1.0;
c1=1.0;
for(int n=1;n<=N1;n++)
{  c2=1;
   for(int nu=0;nu<=(n-1);nu++)
   {c3=alpha[n]-alpha[nu];
    c2=c2*c3;
    if(n<=M) delta[n-1][nu][n]=0.0;
    for(int m=0;m<=Math.min(n,M);m++)
    {
       if(m==0)
       { delta[n][nu][m]=((alpha[n]-x0)*delta[n-1][nu][m])/c3;}
       else
       {delta[n][nu][m]=((alpha[n]-x0)*delta[n-1][nu][m]-m*delta[n-1][nu][m-1])/c3;}
    }//next m
   }//next nu
   for(int m=0;m<=Math.min(n,M);m++)
   {   if(m==0)
               {delta[n][n][m]=c1/c2*(-(alpha[n-1]-x0)*delta[n-1][n-1][m]);}
               else
               {delta[n][n][m]=c1/c2*(m*delta[n-1][n-1][m-1]-(alpha[n-1]-x0)*delta[n-1][n-1][m]);}
   }//next m
   c1=c2;
}//next n
c=delta;
  if(Mi<N) M=N;
  else M=Mi;
  h=hi;
  double deriv=0;
```

```
  double h1=1/h;
  double h2=1;
  for(int j=0;j<N;j++)
  {h2*=h1;}
  for(int i=0;i<c[0].length;i++)
  {   deriv+=c[M][i][N]*func(x+alpha[i]*h);}
  return deriv*h2;
}

default double dfunc(double x,int N)
{int M=30;
 double h=0.05*N;
 return dfunc(x,N,M,h);
}}
```

Bu interface tanımını kullanarak integral alan örnek program:

Program 1.2.5 interface sınıf inter_f_x 'i kullanan lambda deyimi tanımlı integral programı

```
public class inter_f_xtest {
//INTEGRAL
public  static double integral(if_x f,double a,double b)
{ //10 point Gauss-Legendre formula
  //integral f(x)dx
 double r[]={-0.973906528517171,-0.865063366688984,-0.679409568299024,-0.433395394129247,-
0.148874338981631,
 0.148874338981631,0.433395394129247,0.679409568299024,0.865063366688984,0.973906528517171};
 double c[]={0.066671344308684,0.149451349150580,0.219086362515982,0.269266719309996,0.295524224714752,
 0.295524224714752,0.269266719309996,0.219086362515982,0.149451349150580,0.066671344308684};
double z=0,x,y;
double k1=(b-a)/2.0;
double k2=(b+a)/2.0;
for(int i=0;i<r.length;i++)
{
x=k2+k1*r[i];
y=f.func(x);
z+=k1*c[i]*y;
}
return z;
}

public static void main(String arg[])
{ inter_f_x ff=x->2.0*Math.sin(x)*Math.cos(x);
  System.out.println(integral(ff,0.0,1.0));
} }
```

f(x0,x1,x2,..,xN) gibi birden fazla bağımsız değişken içeren fonksiyonları kullanabilmek amacıyla , diğer bir abstract fonksiyon olan f_xj tanımlanmıştır.

Program 1.2.6 f(x0,x1,x2,..xn) fonksiyonunu ve türevlerini abstract sınıf f_xj

```
 abstract class f_xj extends Mathd
{
  // single function  multi independent variable
  // a single value is returned indiced to equation_ref
abstract public double func(double x[]);

public double[] dfunc(double x[])
{int n=x.length;
 double c[]=new double[n];
 for(int i=0;i<n;i++)
 {c[i]=dfunc(x,i);}
 return c;
}
```

```
public double[][] d2func(double x[])
{int n=x.length;
 double c[][]=new double[n][n];

 for(int i=0;i<n;i++)
 {for(int j=0;j<n;j++)
   {   int xref[]={i,j};
               c[i][j]=d2func(x,xref);
   }
 }
 return c;
}

public double d2func(double x[],int x_ref[])
{// derivative of the function with respect to x_ref
double h0=0.0256;
int i,m;
int n=7;
double f1,f2;
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
x2[i]=x[i];
}
//derivative of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref[1]]+=h[i];
x2[x_ref[1]]-=h[i];
f1=dfunc(x1,x_ref[0]);
f2=dfunc(x2,x_ref[0]);
T[i][0]=( f1 - f2)/(2.0*h[i]);
x1[x_ref[1]]=x[x_ref[1]];
x2[x_ref[1]]=x[x_ref[1]];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
```

```
        }
        public double dfunc(double x[],int x_ref)
        { // derivative of the function with respect to x_ref
        double h0=0.0256;
        int i,m;
        int n=7;
        double f1,f2;
        double x1[];
        x1=new double[x.length];
        double x2[];
        x2=new double[x.length];
        for(i=0;i<x.length;i++)
        {
        x1[i]=x[i];
        x2[i]=x[i];
        }
        //derivative of a simple function
        double T[][];
        T=new double[n][n];
        double h[];
        h=new double[n];
        //vector<double> h(n,0);
        for(i=0;i<n;i++)
        {
          h[i]=0;
          for(int j=0;j<n;j++)
            T[i][j]=0;
        }
        h[0]=h0;
        double r=0.5;
        for( i=1;i<n;i++)
        {
        h[i]=h0*Math.pow(r,i);
        }

        for(i=0;i<n;i++)
        {
        x1[x_ref]+=h[i];
        x2[x_ref]-=h[i];
        f1=func(x1);
        f2=func(x2);
        T[i][0]=( f1 - f2)/(2.0*h[i]);
        x1[x_ref]=x[x_ref];
        x2[x_ref]=x[x_ref];
        }
        for(m=1;m<n;m++)
        {
          for(i=0;i<n-m;i++)
          {
          T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
          - h[i+m]*h[i+m]);
          }
        }
        double xx=T[0][n-1];
        return xx;
        }

}
```

Örnek bir kullanım :

Program 1.2.7 abstract sınıf f_xj örnek kullanımı

```
class ff2 extends f_xj
{public double func(double x[])
```

```
{return x[0]*x[0]*x[1]*x[2]+x[1]*x[1];}
}
public class f_xjtest
{
public static void main(String arg[])
{ff2 f=new ff2();
double x1[]={1.0,2.0,1.0};
 double f1[][]=f.d2func(x1);
 System.out.println("f1=\n"+Matrix.toString(f1));
}}
```

```
---------- Capture Output ----------
> "C:\java\bin\java.exe" f_xjtest
f1=
      4.000000001394953      2.000000000221440      4.000000004818655
      2.000000000224681      1.999999997354646      0.999999999182852
      4.000000004815128      0.999999999181636      0.000000000005512

> Terminated with exit code 0.
```

Eşdeğert interface if_xj programı :

Program 1.2.8 interface if_xj f(x0,x1,x2,..xn) fonksiyonu için tanımlanmıştır

```
@FunctionalInterface
interface if_xj
{
  // single function  multi independent variable
  // a single value is returned indiced to equation_ref
public double func(double x[]);

default double[] dfunc(double x[])
{int n=x.length;
 double c[]=new double[n];
 for(int i=0;i<n;i++)
 {c[i]=dfunc(x,i);}
 return c;
}

default double[][] d2func(double x[])
{int n=x.length;
 double c[][]=new double[n][n];
 for(int i=0;i<n;i++)
 {for(int j=0;j<n;j++)
   {   int xref[]={i,j};
               c[i][j]=d2func(x,xref);
   }
 }
 return c;
}

default double d2func(double x[],int x_ref[])
{// derivative of the function with respect to x_ref
double h0=0.0256;
int i,m;
int n=7;
double f1,f2;
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
```

```
x2[i]=x[i];
}
//derivative of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref[1]]+=h[i];
x2[x_ref[1]]-=h[i];
f1=dfunc(x1,x_ref[0]);
f2=dfunc(x2,x_ref[0]);
T[i][0]=( f1 - f2)/(2.0*h[i]);
x1[x_ref[1]]=x[x_ref[1]];
x2[x_ref[1]]=x[x_ref[1]];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}

default double dfunc(double x[],int x_ref)
{ // derivative of the function with respect to x_ref
double h0=0.0256;
int i,m;
int n=7;
double f1,f2;
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
x2[i]=x[i];
}
//derivative of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
```

```
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i];
f1=func(x1);
f2=func(x2);
T[i][0]=( f1 - f2)/(2.0*h[i]);
x1[x_ref]=x[x_ref];
x2[x_ref]=x[x_ref];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}}
```

Program 1.2.9 interface if_xj yi lambda deyimleriyle kullanan örnek program

```
public class if_xjtest
{
public static void main(String arg[])
{if_xj f=(double x[])->x[0]*x[0]*x[1]*x[2]+x[1]*x[1];
double x1[]={1.0,2.0,1.0};
 double f1[][]=f.d2func(x1);
 System.out.println("f1=\n"+Matrix.toString(f1));
}}
```

1.1 İki Boyutlu Veri ve fonksiyonların Grafiklerin Çizilmesi

İki boyutlu grafiklerin çizilebilmesi için göreceli olarak kullanımı basit, fakat mühendislik ve matematik uygulamalarına cevap verebilecek yeterince kompleks bir altyapı oluşturduk. Sayısal analizde sonuçarın grafik olarak görülmesi daima bir dizi rakamın irdelenmesinden daha basit olduğu için, analizlerimizde ve programlarımızda grafik çıktı ortamını sıkça kullanacağız. Plot programlarının kodları da çok uzun olduğundan burada direk olarak verilmemektedir. Açık kodlar internet adresimizden çekilebilir. Grafik çizen çeşitli ticari programlar bulunabilir, ancak biz burda oluşturduğumuz paketlerle aynı program içinde ve belli bir maliyeti olmadan bu gereksinimi karşılamaya çalışıyoruz. İki boyutlu grafik çizimi için geliştirdiğimiz ilk sınıf Plot sınıfıdır. Burada çeşitli örneklerle kullanımını göstermeye çalışacağız. İlk örnek olarak in.txt dosyasından veri okuyarak grafik çizen bir örnek verelim:

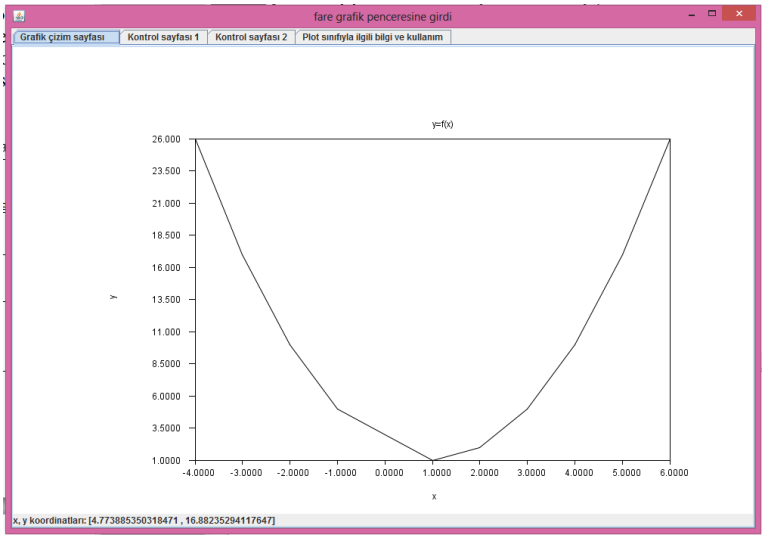
Program 1.3-1. PlotT1.java örnek programı

```
import java.io.*;
class PlotT1
{  public static void main(String args[]) throws IOException
```

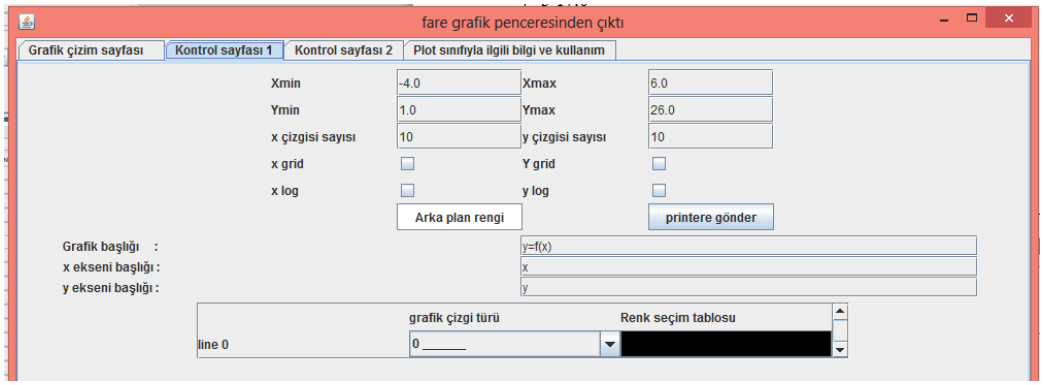
```
{Plot pp=new Plot("in.txt");
pp.plot();}}
```

Veri dosyası: in.txt

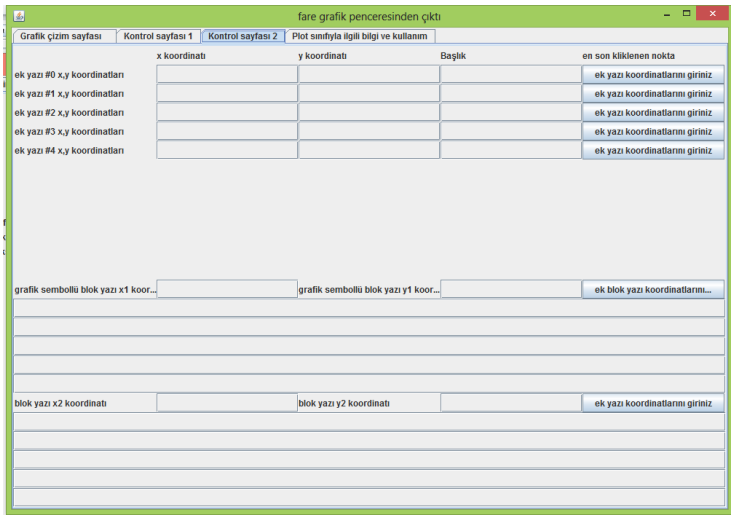
```
-4 26.0
-3 17.0
-2 10.0
-1 5.0
1 1.0
2 2.0
3 5.0
4 10.0
5 17.0
6 26.0
```



(bolum1_000.jpg,Resimaltı:PlotT1 örnek grafiği)



(bolum1_001.jpg,Resimaltı:Plot ikinci sayfa görünümü)



(bolum1_002.jpg,Resimaltı:Plot üçüncü sayfa görünümü)

Bu örneğimizde Plot pp=new Plot("in.txt"); terimi in.txt dosyasını okuyan kurucu metodu çağırır ve pp değişkenine atar. pp.plot(); metodu ile de çizdirir. İkinci sayfada renk, girdi, grid(kareli görüntü), logaritma skalası, grafik çizgi türü, grafik başlıklarını atama gibi işlevlerde kullanılır. Son sayfa ise grafik penceresine çeşitli alt ek yazılar yazılabilmesini sağlar. Aynı grafik çizdirme işlevi iki komut yerine tek bir komutla da verilebilir. Bu statik bir metodu çağırır. Statik metodlar genelde sadece tek bir eylem (tek bir grafik) çizdirilecekse kullanılır.

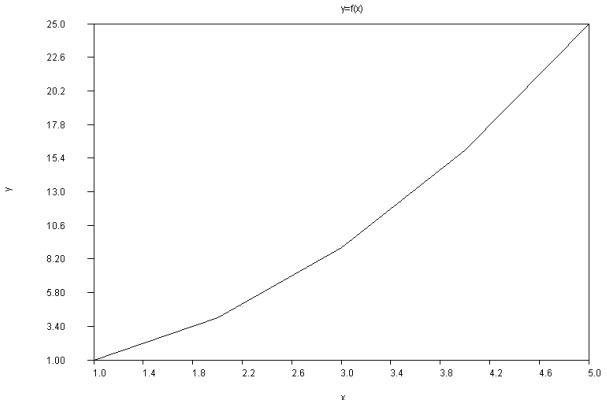
Program 1.3-2 PlotT1A.java örnek programı

```
import java.io.*;
class plotT1A
{
public static void main(String args[]) throws IOException
{Plot.file("in.txt");} }
```

Bir sonraki örnekte double vektör olarak verilen veri kullanılarak grafik çizdirilmektedir.

Program 1.3-3 PlotT2.java program plotting directly from arrays

```
import java.io.*;
class PlotT2
{    public static void main (String args[]) throws IOException
{double xx[]={1,2,3,4,5};
double yy[]={1,4,9,16,25};
Plot pp=new Plot(xx,yy);
pp.plot();
}
}
```



(bolum1_003.jpg,Resimaltı:PlotT2 örnek grafiği)

Aynı programın statik versiyonu:

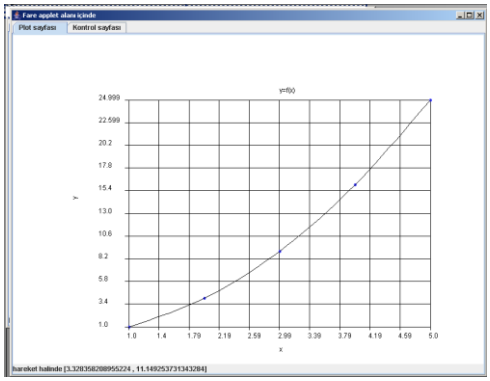
Program 1.3.4

```
import java.io.*;
class PlotT2A
{    public static void main (String args[]) throws IOException
     {double xx[]={1,2,3,4,5};
      double yy[]={1,4,9,16,25};
      Plot.data(xx,yy);}}
```

Çıktı yine üstte tanımlandığı gibidir. Yeni bir örnek olarak veriye aradeğer hesabı yaparak çizim yapan ek bir grafik ekleyelim. Bunun için addSpline metodu kullanacağız. Bu metod kubic şerit interpolasyonlarıyla aradeğer hesaplar ve çizer. Ara değer hesaplarının detaylarını aradeğer (interpolasyon) bölümünde inceleyeceğiz.

Program 1.3.5

```
import java.io.*;
class PlotT2B
{    public static void main (String args[]) throws IOException
     {double xx[]={1,2,3,4,5};
      double yy[]={1,4,9,16,25};
      Plot pp=new Plot(xx,yy);
      pp.addSpline(xx,yy,10);
      pp.plot();}}
```



(bolum1_004.jpg,Resimaltı: PlotT2B çıktısı, addSpline aradeğer çıktısı kullanılıyor)

Bu grafikte iki eğri çizdiğimizi not edelim. Birden fazla grafik komutunu bir arada kullanmak isterseniz statik metodlara başvuramazsınız, örneğin yukardaki programda hem veriyi hemd e verinin interpolasyon değerlerini çizdirdiğimizden statik kullanıma uygun değildir. Ancak kendi statik metodumuzda bu tür kompleks kullanımları birleştirerek kullanabiliriz. Program 1.3-6, Program 1.3.5 ile aynı çıktıyı verecektir.

Program 1.3.6

```
import java.io.*;
class PlotT2B
{
public static void plot(double xx[],double yy[])
{Plot pp=new Plot(xx,yy);
pp.addSpline(xx,yy,10);
pp.plot();}
public static void main (String args[]) throws IOException
{double xx[]={1,2,3,4,5};
double yy[]={1,4,9,16,25};
plot(xx,yy);}}
```

Aynı aradeğer hesaplarını **pp.addB_Spline(xx,yy,10);** deyimini ekleyerek **B şerit interpolasyonu** kullanarak, **pp.addOPEKK(xx,yy,2,10);** deyimiyle ikinci dereceden en küçük kareler yöntemi kullanarak, **pp.addLagrange(xx,yy,10);** Deyimiyle Lagrange interpolasyon formülü kullanarak da yapabiliriz.

Plot Programındaki kurucu metodumuz tamamen boş olabilir.

Program 1.3.7 (Dinamik çağırma)

```
import java.io.*;
class PlotT01
{public static void main (String args[]) throws IOException
{Plot pp=new Plot();
pp.plot();}}
```

Program 1.3.8 (Statik çağırma)

```
import java.io.*;
class PlotT2B1
{    public static void main (String args[]) throws IOException
{Plot.file();}}
```

Bu durumda programımız veri için Plot.txt dosyasına bakacaktır.Bu dosyada

```
Plot başlığı
x ekseni
y ekseni
2
in.txt  0 0 0 0
out.txt 3 0 0 255
```

verisi olduğunu varsayalım. Bunun anlamı "Plot başlığı" "x ekseni" ve "y ekseni" grafik başlıklarını kullanarak in.txt ve out.txt dosyalarındaki verilerin çizilmesidir. Verilerin yanındaki ilk rakam grafik çizim türünü ondan sonraki 3 rakam da renk kodunu (kırmızı yeşil ve mavi olarak) ifade eder.

Programın grafik çizim kodları :

0_____
1 __ __ __ __
2 . _ . _ . _
3 . _ _ . . _ _ .
4 - - - - - - - -
5 _____ kalın
6 __ __ __ __ kalın
7 . _ . _ . _ kalın
8 . _ _ . . _ _ .",
9 - - - - - kalın
10 ⊕
11 ⊗
12 ⊙
13 ⊚
14 ⊛
15 ⊞
16 ⊠
17 ⊡
18 ⊹
19 ⁕
20 kare

21 dolu kare
22 daire
23 dolu daire
24 üçken
25 eşkenar dörtgen
26 beşken
27 altıgen
28 dolu üçgen
29 dolu eşkenar dörtken
30 dolu beşgen
31 dolu altıgen
32 üç köşe yıldız
33 dört köşe yıldız
34 beş köşe yıldız
35 altı köşe yıldız
36 dolu üç köşe yıldız
37 dolu dört köşe yıldız
38 dolu beş köşe yıldız
39 dolu altı köşe yıldız
40 çubuk grafik
41 dolu çubuk grafik

şeklindedir. Grafik türü ve renk kodları burada bahsettiğimiz tüm metodlar için de mevcuttur. Örneğin

Bir fonksiyonun grafiğinin çizilmesi için en basit yol fonksiyonu abstract sınıf f_x ten türeterek yapılır.

Program 1.3.9 (Dinamik versiyon)

```
import java.io.*;
class f3 extends f_x
{double func(double x)
{return x*x*x+3.6*x*x-36.4;}}

class PlotT06
{public static void main (String args[]) throws IOException
{f3 ff2=new f3();
Plot pp=new Plot(ff2,0.0,7.0,400,2);
pp.plot();}}
```

Program 1.3.9 (Dinamik lambda deyimli versiyon)

```
import java.io.*;

class PlotT06B
{public static void main (String args[]) throws IOException
{ if_x f=x-> x*x*x+3.6*x*x-36.4;;
  Plot pp=new Plot(f,0.0,7.0,400,2);
  pp.plot();
}}
```

Program 1.3.10 (Statik versiyon)

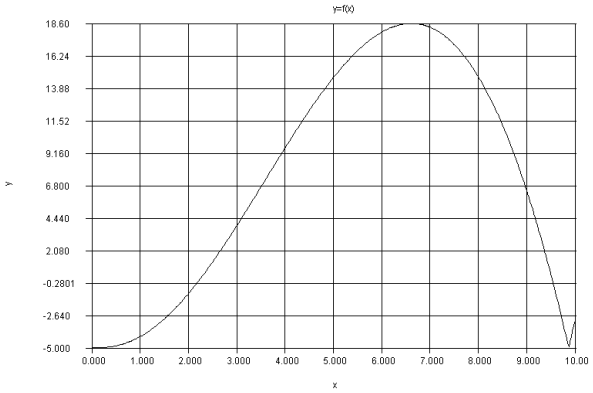
```
import java.io.*;

class f3 extends f_x
{double func(double x)
```

```
{ return x*x*Math.abs(Math.sin(Math.sqrt(x)))-5.0;}}

class PlotT06
{    public static void main (String args[]) throws IOException
{Plot.func(new f3(),0.0,10.0,400,0);}
}
```

Program 1.3.10A (Statik lambda deyimli versiyon)

```
import java.io.*;

class PlotT06A
{    public static void main (String args[]) throws IOException
{ if_x f=x->x*x*Math.abs(Math.sin(Math.sqrt(x)))-5.0;
  Plot.func(f,0.0,10.0,400,0);}
}
```

bu programın grafik çıktısı



(bolum1_005.jpg,Resimaltı: PlotT06 çıktısı)

şeklinde olacaktır. Bu fonksiyon matematik olarak ifade edilirse $f(x)=x^3+3.6x^2-36.4$ şeklindedir. 0 ile 7 arasında 400 nokta oluşturarak fonksiyonunun çizilmesi istenmiştir.

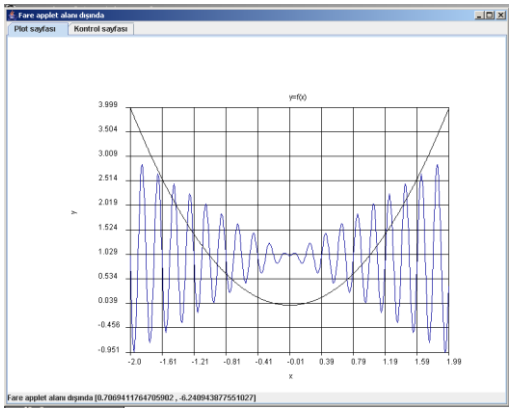
Birden fazla fonksiyonun grafiğini tek bir sınıfta tanımlıyabiliriz.

Program 1.3.11 (Dinamik versiyon)

```
import java.io.*;
class f3 extends fi_x
{
double[] func(double x)
{ double f[]=new double[2];
f[0]=x*Math.sin(10.0*Math.PI*x)+1.0;
f[1]=x*x;
return f;}}

class PlotT03
{public static void main (String args[]) throws IOException
{ f3 ff2=new f3();
  Plot pp=new Plot(ff2,-2.0,2.0,400,2);
  pp.plot();}}
```

Program 1.3.12 (Statik versiyon)

```
import java.io.*;

class f3 extends fi_x
{
double[] func(double x)
{ double f[]=new double[2];
f[0]=x*Math.sin(10.0*Math.PI*x)+1.0;
f[1]=x*x;
return f;}
}

class PlotT03
{public static void main (String args[]) throws IOException
{ Plot.func(new f3(),-2.0,2.0,400,2);}}
```



(bolum1_006.jpg,Resimaltı: PlotT03 çıktısı)

veya ayrı ayrı birden fazla sınıfta tanımlayarak da çizebiliriz.

Program 1.3.13

```
import java.io.*;
class f3 extends f_x
{double func(double x)
{return x*x;}}

class f4 extends f_x
{double func(double x)
{return x*Math.sin(10.0*Math.PI*x)+1.0;}}

class PlotT07
{public static void main (String args[]) throws IOException
{f3 ff3=new f3();
f4 ff4=new f4();
Plot pp=new Plot(ff3,-3.0,2.0,400);
pp.addFunction(ff4,-2.0,2.0,400);
pp.plot();}}
```

Program 1.3.13A (lambda deyimli versiyon)

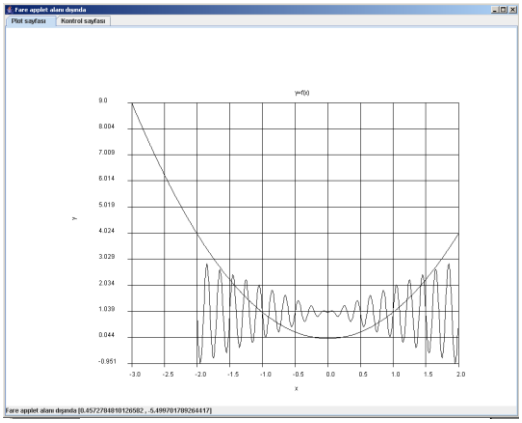
```
import java.io.*;
```

```
class PlotT07A
{public static void main (String args[]) throws IOException
{if_x ff3=x->x*x;
 if_x ff4=x->x*Math.sin(10.0*Math.PI*x)+1.0;
 Plot pp=new Plot(ff3,-3.0,2.0,400);
 pp.addFunction(ff4,-2.0,2.0,400);
 pp.plot();}}
```

Fonksiyonlar ayrı ayrı çağrıldığında değişik sınır değerleri de tanımlayabiliriz. Burada kurucu metot birinci fonksiyon tarafından çağrıldığından ikinci fonksiyonun çağrılmasında

pp.addFunction(ff4,-2.0,2.0,400);

kullanılmıştır.



(bolum1_007.jpg,Resimaltı: PlotT07 çıktısı)

addFunction metoduna benzer olarak addData metodu mevcuttur, java boyutlu değişken verilerinin eklenmesinde kullanılır.

pp.addData(xx,yy,0,0,0,0);

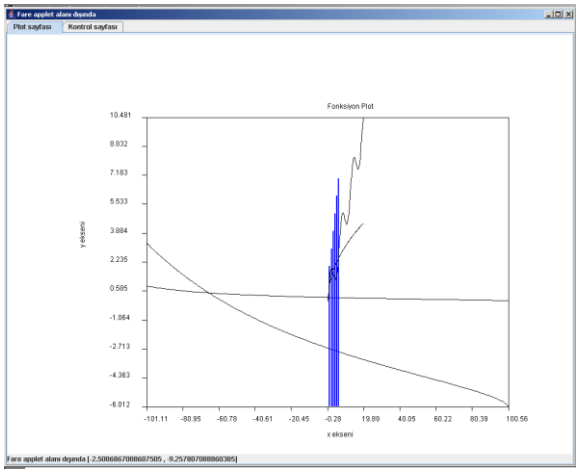
Son olarak çeşitli fonksiyon ve verileri kullanan ve diğer grafik verilerini de programın içinde değiştiren bir programa göz atalım

Program 1.3.14

```
import java.io.*;

class f3 extends fi_x
{
double[] func(double x)
{
double f[]=new double[2];
f[0]=Math.sin(x)+x/2;
f[1]=Math.sqrt(x);
return f;
}}
class PlotT3
{
public static void main (String args[]) throws IOException
{//Plot pp=new Plot();
String fi[]={"in.txt","out.txt"};
f3 ff=new f3();
int ipt[]={0,1};
```

```
int ir[]={0,0};
int ig[]={0,0};
int ib[]={0,2};
double xx[]={1,2,3,4,5,6};
double yy[]={2,3,4,5,6,7};
Plot pp=new Plot(fi); //read data from fi (in.txt,out.txt data are added up)
//pp.addData(fi);
pp.setPlotType(0,0);//
pp.setPlotType(1,1);//
pp.addData(xx,yy,0,0,0,0); //one set of addional data
pp.setPlotType(2,41);//bar chart
pp.setColor(2,0,0,255); //blue
pp.addFunction(ff,0.0,20.0,50,2,ipt,ir,ig,ib); //two additional function
pp.setPlabel("Fonksiyon Plot");
pp.setXlabel("x ekseni");
pp.setYlabel("y ekseni");
pp.plot();}}
```

Bu program 2 fonksiyonu f3 sınıfından almakta, 2 set veriyi fi string'i aracılığıyla in.txt ve out.txt dosyalarından okumaktadır. 3. veri seti(2 no'lu veri) bar grafik (grafik tipi 41) olarak seçilmiştir. Bu paketleri kullanarak çok daha karmaşık grafik bileşenleri oluşturulabilir. Örneğin nemli havanın termodinamik özellikleri programında fare ile aktif veri girişi (Fareye basıldığında x ve y koordinatlarını alarak bu değerler için hesap yapma) gibi ek özellikler katılabilir. Bu özellikler burada verdiğimiz paketin içerisinde kullanıma hazır olarak mevcutsa da verdiğimiz şekliyle kullanılamaz.



(bolum1_008.jpg,Resimaltı: PlotT3 çıktısı)

Grafik Paketi Temel Bileşenleri :
Grafik Paketinin temel bileşenleri :
Plot.java kurucu metotlar ve girdi çıktı metotlarını içerir
PlotW.java : çizimi gerçekleştiren ana metodudur. Grafik sayfalarını ve bu sayfaların kontrollerini içerir.
PlotShapesSW: çeşitli Plot alt elemanlarının çizim tekniklerini içerir, çizgi, üçgen,bar grafik gibi
Plot2D : PlotShapes alt elemanlarını grafik çizimi haline getirir, sırasıyla alt elemanları çağırır.

Şimdi bu paketlerin listelerini verelim. Bu tür program kodları programlamaya yeni başlayan arkadaşlarımız için kompleks olabilir, bu durumda sadece kullanımla yetinebilirler, ancak dileyen kodları alarak daha da geliştirebilir, Bu tür geliştirmelerde tek dileğim geliştirmelerden beni de haberdar etmeniz, eğer yeni kodlar uygunsa orijinal kodları değiştirebiliriz. Programlarımızın kodlarını vermeden önce tüm metodlarımızın bir listesini verelim :

Kurucu metodlar : (grafik programını dinamik olarak kurar ve veriyi kullanıma hazırlar)
public Plot() throws IOException

public Plot(String pl,String xl,String yl,int xnt,int ynt,int xgo,int ygo,String fn[],int ipt[],int ir[],int ig[]
,int ib[]) throws IOException
public Plot(String fn[],int ipt[],int ir[],int ig[],int ib[]) throws IOException
public Plot(String fn[]) throws IOException : Read file names from String fn[] and plot them
public Plot(double xi[][],double yi[][],int ipt[],int ir[],int ig[],int ib[]) : read plot data from xi[][] and yi[][]
public Plot(double xiyi[][]) : read plot data xi[] and yi[] from xiyi[][] string
public Plot(double xi[][],double yi[][]) : read plot data from xi[][] and yi[][]
public Plot(double xi[],double yi[]) : read plot data from xi[] and yi[]
public Plot(f_x f,double xm,double xma,int N,int ipt,int ir,int ig,int ib)
public Plot(f_x f,double xm,double xma,int N) : create plot data from a simple function f_x f
public Plot(fi_x f,double xm,double xma,int Number_of_Data,int Number_of_Function)

Statik metodlar : (verinin kısa yoldan grafiğinin çizilmesini sağlar)
public static double[] data(double xmin,double xmax,double dx)
public static double[] data(double xmin,double xmax,int n)
public static Plot func(f_x ff1,double xmin,double xmax,int n,int pnumber)
public static Plot func(f_x ff1,double xmin,double xmax,int n)
public static Plot func(f_x ff1,double xmin,double xmax)
public static Plot func(fi_x ff1,double xmin,double xmax,int n,int pnumber)
public static Plot func(fi_x ff1,double xmin,double xmax,int n)
public static Plot func(fi_x ff1,double xmin,double xmax)

x , y ve ana başlığı eklemek
public void setPlabel(String ip) {label=ip;}
public void setXlabel(String ix) {xlabel=ix;}
public void setYlabel(String iy) {ylabel=iy;}
public void setXYlabel(String ix,String iy) {xlabel=ix;ylabel=iy;}

kurucu metoddan sonra grafik penceresine ek veri eklemek
public void addData(double xi[],double yi[])
public void addData(double xi[],double yi[],int iplottype)
public void addData(double xi[],double yi[],int iplottype,int ir,int ig,int ib)
public void addData(double xi[][],double yi[][])
public void addData(double xi[][],double yi[][],int ip[])
public void addData(double xi[][],double yi[][],int ip[],int ir[],int ig[],int ib[])
public void addFunction(f_x f,double xmi,double xma,int N,int ipt)
public void addFunction(f_x f,double xmi,double xma,int N)
public void addFunction(f_x f,double xmi,double xma,int N,int ipt,int ir,int ig,int ib)
public void addFunction(fi_x f,double xmin,double xmax,int Number_of_Data,int Number_of_Function)
public void addFunction(fi_x f,double xmin,double xmax,int Number_of_Data,int Number_of_Function,int ipt[])
public void addFunction(fi_x f,double xmin,double xmax,int Number_of_Data,int Number_of_Function, ,int ipt[])

eğri uydurarak ek veri eklemek
public void addOPEKK(double xi[],double yi[],int npolinom,int aradegersayisi)
public void addOPEKK(double xi[],double yi[],int npolinom,int aradegersayisi,int iplottype)
public void addOPEKK(double xi[],double yi[],int npolinom,int aradegersayisi,int iplottype,int ir,int ig,int ib)
public void addEKK(double xi[],double yi[],int npolinom,int aradegersayisi)
public void addEKK(double xi[],double yi[],int npolinom,int aradegersayisi,int iplottype)
public void addEKK(double xi[],double yi[],int npolinom,int aradegersayisi,int iplottype,int ir,int ig,int ib)
public void addEKK(int linenumber,int npolinom,int aradegersayisi,int iplottype)
public void addEKK(int linenumber,int npolinom,int aradegersayisi,int iplottype,int ir,int ig,int ib)
public void addLagrange(double xi[],double yi[],int aradegersayisi)
public void addLagrange(double xi[],double yi[],int aradegersayisi,int iplottype)

public void addLagrange(double xi[],double yi[],int aradegersayisi,int iplottype,int ir,int ig,int ib)
public void addLagrange(int linenumber,int aradegersayisi,int iplottype)
public void addLagrange(int linenumber,int aradegersayisi,int iplottype,int ir,int ig,int ib)
public void addB_Spline(double xi[],double yi[],int nSpline)
public void addB_Spline(double ai[][],int nSpline)
public void addB_Spline(double xi[],double yi[],int nSpline,int iplottype)
public void addB_Spline(double xi[],double yi[],int aradegersayisi,int iplottype,int ir,int ig,int ib)
public void addB_Spline(int linenumber,int nSpline)
public void addB_Spline(int linenumber,int nSpline,int iplottype)
public void addB_Spline(int linenumber,int aradegersayisi,int iplottype,int ir,int ig,int ib)
public void addSpline(double xi[],double yi[],int nSpline)
public void addSpline(double ai[][],int nSpline)
public void addSpline(double xi[],double yi[],int nSpline,int iplottype)
public void addSpline(double xi[],double yi[],int aradegersayisi,int iplottype,int ir,int ig,int ib)
public void addSpline(int linenumber,int nSpline)
public void addSpline(int linenumber,int nSpline,int iplottype)
public void addSpline(int linenumber,int aradegersayisi,int iplottype,int ir,int ig,int ib)

grafik özelliklerini değiştirmek
==========**Grafik türü** ==========
public void setPlotType(int plot_type[])
public void setPlotType(int dataset,int plot_no)
public void setPlotType(int dataset,char plot_char)
==========**x ve y grid** ==========
public void setXgrid(int igx)
public void setYgrid(int igy)
public void setGrid(int igx,int igy)
public void setXtic(int ixt)
public void setYtic(int iyt)
public void setXYtic(int ixt,int iyt)
==========**logaritmik grafik skalası açma ve kapama** ==========
public void setXlogScaleOn()
public void setYlogScaleOn()
public void setXlogScaleOff()
public void setYlogScaleOff()
==========**renk seti**==========
public void setColor(int dataset,int ired,int igreen,int iblue)
public void setColor(int dataset,Color c)
public void setColor(int ired[],int igreen[],int iblue[])
public void setColor(Color c[])
========== **fonksiyon minimum ve maksimum değerleri** ==========
public void setMinMax()
public void setMinMax(double xi[][],double yi[][])
public void setMinMax(double iminx,double imaxx,double iminy,double imaxy)
========== **grafik penceresi formatlanması** ==========
public void set_plotwindow(int width,int height,double xip,double yip,double dxp,double dyp)
public void setabsMaxMin(int ixmin,int iymin,int idx,int idy)

oluşan verinin dinamik ortamda çizilmesi
public void plot()

1.2 Üç boyutlu Veri ve fonksiyonların Grafiklerin Çizilmesi

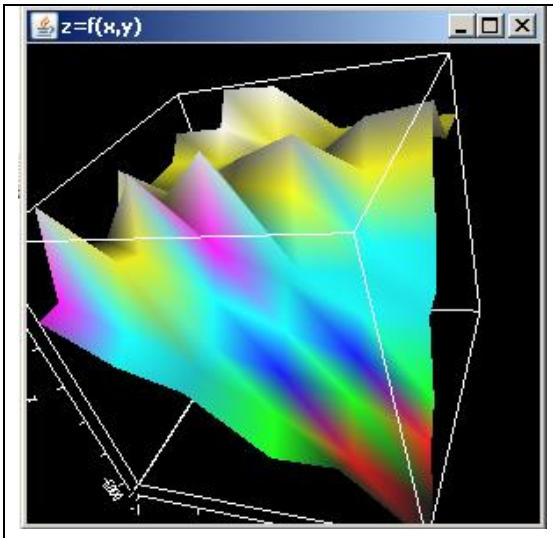
Java'da 3 Boyutlu grafik çıktıları oluşturmak için serbest dağıtım sisteminin içerisinde yer alan mükemmel bir java programlama seti olan VisAD programlama paketinden yararlanılmıştır. VisAD çok çeşitli 3 boyutlu uygulamalar için Java 3D grafik paketinikullanan bir program paketidir. Visad Paketini http://www.ssec.wisc.edu/~billh/visad.html veya çeşitli adreslerden indirebilirsiniz. İsterseniz kendiniz kodları derleyerek kurabilirsiniz, isterseniz, hazır visad.jar dosyasını javanın jre dosyası altındaki lib dosyası altındaki ext dosyasına kopyalıyarak (örneğin benim bilgisayarımda c:\co\java\jre\lib\ext) bilgisayarınızı VisAD paketini direk kullanabilir hale getirebilirsiniz. 3 boyutlu grafikler için aynı zamanda Java 3D grafik paketine de ihtiyacınız olacaktır. bu paketin en son versiyonuna www.java.sun.com adresinden temin edebilirsiniz (benim şu anda kullandığım versiyonu java3d-1_4_0_01-windows-i586) paket açtığınızda kendi kendine yüklenecektir.
Java 3D paketi default olarak Windows ortamında **openGL** kullanır (diğer ortamlardaki grafik kullanımı ile ilgili Java 3D paketini inceleyiniz). Eğer grafik kartınız openGL sistemini destekliyorsa bu bölümde verdiğimiz programları direk olarak kullanabilirsiniz. Eğer openGL'i desteklemeyen bir grafik kartınız varsa DirectX9.0 veya daha sonraki bir versiyonunu kullanmanız gerekecektir. Bu durumda programı çalıştırırken Java 3D paketine default değer olan openGL yerine directX kullandığınızı belirtmeniz gerekecektir. Bunun için java programını çalıştırırken :
java -Dj3d.rend=d3d sınıfismi
komutunu kullanmanız gerekir. directX'in Windows için en son versiyonunu
http://www.microsoft.com/windows/directx/ adresinden bulabilirsiniz.
Oluşturduğumuz 3D programı Plot3D için temel olarak VisAD paketinin örnek problemlerini kullandık, bu örnek programları bizim iki boyutlu grafik ortamlarında da kullandığımız fonksiyon abstract sınıf ve veri yapılarına uygun hale getirdik. Bunun için geliştirilen Plot3D Programı www.turhancoban.com adresinden alınabilir. Bu programı ne şekilde kullanabileceğimizi çeşitli örnek problemlerle görelim.Kodu bir araya getirirken Plot programında olduğu gibi çok basit bir şekilde kullanılır olmasını düşündük. Kartınız openGL'i desteklemiyorsa(Windows sistemleri için) programı directX ile çalıştırmak için java komutunu
java -Dj3d.rend=d3d Plot3DT1
şeklinde kullanınız.

Program 1.4.1 Plot3D 3 boyutlu bilimsel grafik programı kullanım örneği

```
import visad.*;
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Plot3DT1
{
public static void main(String[] args) throws RemoteException, VisADException
{
double a[][]=new double[][]{
{3000.0, 3000.0, 6500.0, 4000.0, 3500.0, 4000.0, 5500.0, 4000.0, 4000.0, 4000.0, 5000.0, 1000 },
{1000.0, 1500.0, 4000.0, 3500.0, 6300.0, 4500.0, 4000.0, 3800.0, 3800.0, 3800.0, 5000.0, 6400 },
{ 0.0, 0.0, 0.0, 1500.0, 3000.0, 6500.0, 4500.0, 5000.0, 4000.0, 3800.0, 3800.0, 6200 },
{-3000.0, -2000.0, -1000.0, 0.0, 1500.0, 1000.0, 4000.0, 5800.0, 4000.0, 4000.0, 3900.0, 3900 },
{-3000.0, -4000.0, -2000.0, -1000.0, 0.0, 1000.0, 1500.0, 4000.0, 5700.0, 4500.0, 4000.0, 4000.0
},
{-6500.0, -6000.0, -4000.0, -3000.0, 0.0, 100.0, 1500.0, 4500.0, 6000.0, 4000.0, 4000.0, 4000.0
}};
```

```
Plot3D.plot3D_CD1(a,-1,2.0,-1.0,1.0,40,"x","y","z=f(x,y)");
}
}
```

Çıktı 1.4 .1 Plot3D 3 boyutlu bilimsel grafik programı veri ile **kullanım örneği**



(bolum1_009.jpg,Resimaltı: Plot3DT1 çıktısı)

Gördüğünüz gibi grafik çıktısını alabilmek için çok fazla bir bilgiye ihtiyacınız olmayacak, veriyi verip çiz demeniz temel olarak yeterli olacaktır. Şimdi veriyi bir fonksiyondan alalım. . Şimdi veriyi bir fonksiyondan alalım. Fonksiyonları kullanırken bir abstract sınıf olan f_fj sınıfı üzerinden tanımlanacaklardır.

Program 1.4.2 f_xj abstract sınıfı

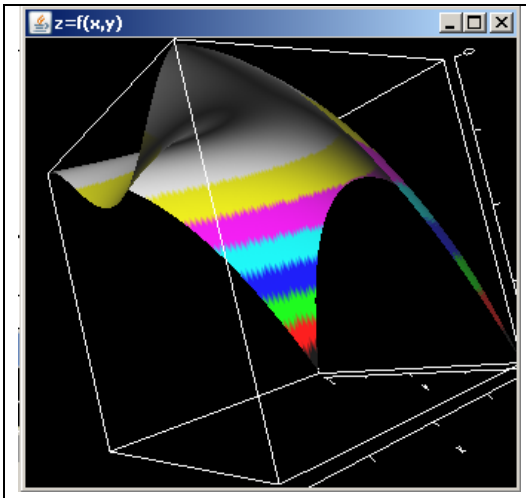
```
abstract class f_xj extends Mathd
{
  // single function  multi independent variable
  // a single value is returned indiced to equation_ref

  abstract double func(double x[]);
}
```

Program 1.4.3 Plot3D 3 boyutlu bilimsel grafik programı fonksiyon ile **kullanım örneği**

```
import visad.*;
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=(x[1]-x[0]*x[0])*(x[1]-x[0]*x[0])+0.01*(x[0]-1.0)*(x[0]-1.0);
return -ff;
```

```
}}
public class Plot3DT2
{
public static void main(String[] args) throws RemoteException, VisADException
{ f1 f=new f1();
Plot3D.plot3D_CD1(f,-1,1,-1,1.0,100,100,40,"x","y","z=f(x,y)");
}
}
```

Çıktı 1.4.4 Plot3D 3 boyutlu bilimsel grafik programı fonksiyon ile **kullanım örneği**



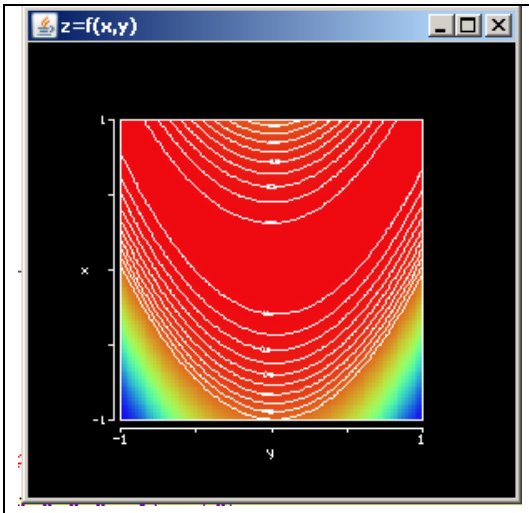
(bolum1_010.jpg,Resimaltı: Plot3DT2 çıktısı)

Burada fonksiyonumuz f_xj abstract sınıfından türetilmiştir. f_xj sınıfı

Program 1.4.5 Plot3D 3 boyutlu bilimsel grafik programı CONTOUR grafiği fonksiyon ile **kullanım örneği**

```
import visad.*;
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=(x[1]-x[0]*x[0])*(x[1]-x[0]*x[0])+0.01*(x[0]-1.0)*(x[0]-1.0);
return -ff;
}
}
public class Plot3DT3
{
public static void main(String[] args) throws RemoteException, VisADException
{ f1 f=new f1();
Plot3D.contour_Plot1(f,-1.0,1.0,-1.0,1.0,100,100,20,"x","y","z=f(x,y)");
}
}
```

Çıktı 1.4.6 Plot3D 3 boyutlu bilimsel grafik programı CONTOUR grafiği fonksiyon ile **kullanım örneği**



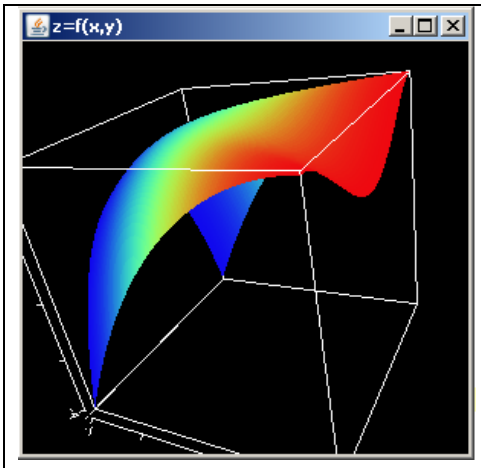
(bolum1_011.jpg,Resimaltı: Plot3DT3 çıktısı)

Contour grafikleri 3 boyutlu eğrinin iki boyuta izdüşümüdür. Haritalarda da bu yöntemi kullanmaktayız.

Program 1.4.7 Plot3D 3 boyutlu bilimsel grafik programı plot_RGB grafiği fonksiyon ile **kullanım örneği**

```
import visad.*;
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=(x[1]-x[0]*x[0])*(x[1]-x[0]*x[0])+0.01*(x[0]-1.0)*(x[0]-1.0);
return -ff;
}
}

public class Plot3DT5
{
public static void main(String[] args) throws RemoteException, VisADException
{ f1 f=new f1();
Plot3D.plot3D_RGB(f,-1.0,1.0,-1.0,1.0,100,100,80,"x","y","z=f(x,y)");
}
}
```

Çıktı 1.4.8 boyutlu bilimsel grafik programı RGB grafiği fonksiyon ile **kullanım örneği**



(bolum1_012.jpg,Resimaltı: Plot3DT5 çıktısı)

Burada verdiğimiz kadarı size 3 boyutlu grafiklere bir giriş sağlamak ve aynı zamanda sayısal çözümleme sonuçlarını gözlemleyebileceğimiz 3 boyutlu grafik ortamını oluşturmak amacı ileydi. Eğer daha detaylı kullanımlara girmek isterseniz VisAD veya benzeri paketleri detaylı inceleyerek bu konuda kendinizi rahatlıkla geliştirebilirsiniz.

Üç boyutlu harita (contour) tipi grafiklerin çizilmesinde http://thehuwaldtfamily.org/java/index.html adresinde listelenen Joseph A. Huwaldth tarafından geliştirilen program paketini üstte tanımladığımız gibi 3 boyutlu abstract fonksiyonlarla birlikte kullanılabilecek forma getirerekContourPlot.java programını oluşturduk. Bu programın kodunu da internet sitemizde bulabilirsiniz. ContourPlot programı fonksion tanımında f_xj abstract sınıfını kullanabildiği gibi f_xy abstract sınıfı üzerinden de fonnksiyon tanımı alabilir.

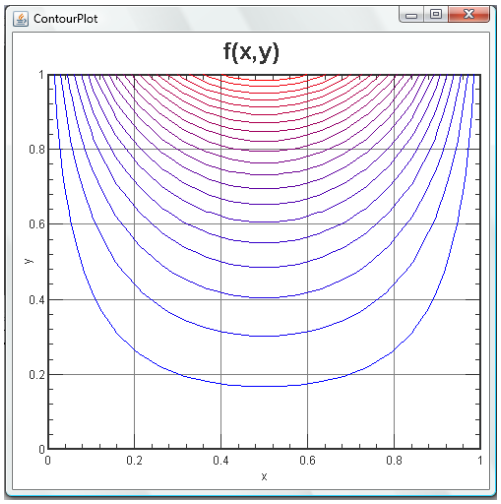
Program 1.4-8 Abstract fonksiyon f_xy

```
abstract class f_xy extends Mathd
{
  abstract double func(double x,double y);
}
```

Bu programla ilgili bir örnek grafik kullanımına bakalım:

Program 1.4-9 ContourPlot örneği Plot3DT2.java

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class f1 extends f_xy
{
public double func(double x,double y)
{//required function a series solution of a two dimensional temperature profile of a plate
double ff;
int artieksi=-1;
double L=1.0;;
double W=1.0;
double toplam=0;
for(int n=1;n<200;n++)
{   artieksi*=-1;
  toplam+=(artieksi+1)/n*Math.sin(n*Math.PI*x/L)*Math.sinh(n*Math.PI*y/L)/Math.sinh(n*Math.PI*W/L);}
toplam*=(2.0/Math.PI);
return toplam; }}
  public class Plot3DT2
  { public static void main(String[] args) { f1 fa=new f1();
    ContourPlot pp = new ContourPlot(fa,0.0,1.0,20,0.0,1.0,20,20, false, "f(x,y)", "x", "y");
    pp.plot();}}
```



(bolum1_013.jpg,Resimaltı: ContourPlot çıktısı)

1.3 Gerçek Zaman Grafik Ortamı

Gerçek zaman grafikleri zaman içinde değişebilen grafikler olduğu gibi program çalışırken dinamik olarak oluşturulan fonksiyonları çizen programlar da olabilir. İlk olarak zaman içinde 3 boyutlu (x-y-t) grafik çizimi yapabilen timeplot_n.java programını inceleyelim. Program kodu yeterince kısa olduğundan program listesini burada direk olarak verebiliriz. Zaman boyutu birbiri ardınca zaman içinde değişen grafik zincirleri şeklinde oluşmaktadır, böylece zaman üzerindeki değişimi bir film gibi görmek mümkün olabilmektedir.

Program 1.5-1 zamana bağlı grafik programı timeplot_n

```
abstract class fi_xt extends Mathd
{
// n function function  two independent variable : x,t
// n value is returned
abstract double[] func(double x,double t);
}

abstract class f_xt extends Mathd
{
// single function  two independent variable : x,t
// a single value is returned
abstract double func(double x,double t);
}

class timeplot_n extends Thread
{int n,nx,nt;
double x[][][];
double y[][][];
double dt;
double dt_real;
double xmin,xmax,ymin,ymax;

public timeplot_n(double xi[][][],double yi[][][],double dti,double dt_reali)
{
nx=xi[0][0].length;nt=xi[0].length;n=xi.length;dt=dti;
dt_real=dt_reali;
xmin=9.99e50;
xmax=-9.99e50;
ymin=9.99e50;
ymax=-9.99e50;
x=new double[n][nt][nx];
y=new double[n][nt][nx];
for(int k=0;k<n;k++)
{
for(int i=0;i<nt;i++)
{for(int j=0;j<nx;j++)
{x[k][i][j]=xi[k][i][j];
y[k][i][j]=yi[k][i][j];
if(x[k][i][j]<xmin) xmin=x[k][i][j];
if(x[k][i][j]>xmax) xmax=x[k][i][j];
```

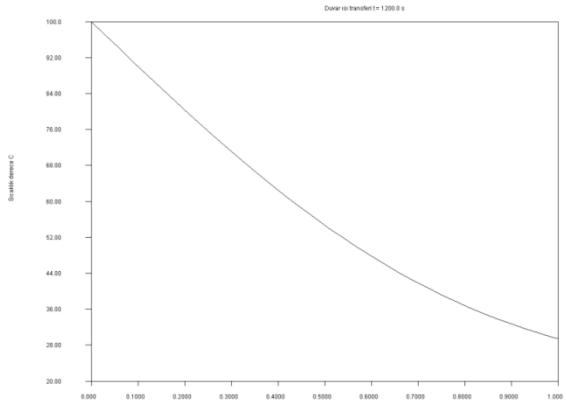
```
if(y[k][i][j]<ymin) ymin=y[k][i][j];
if(y[k][i][j]>ymax) ymax=y[k][i][j];
}
}
}
}

public timeplot_n(fi_xt f,double x0,double xn,double dx,double t0,double tn,double dti,double dt_reali)
{dt=dti;
dt_real=dt_reali;
nx=(int)((xn-x0)/dx+1);
nt=(int)((tn-t0)/dt_real+1);
double a[]=f.func(0.0,0.0);
n=a.length;
xmin=x0;
xmax=xn;
ymin=9.99e50;
ymax=-9.99e50;
x=new double[n][nt][nx];
y=new double[n][nt][nx];
double xx=0;
double t;
for(int i=0;i<nt;i++)
{for(int j=0;j<nx;j++)
{xx=x0+dx*j;
t=t0+dt_real*i;
a=f.func(xx,t);
for(int k=0;k<n;k++)
{x[k][i][j]=xx;
y[k][i][j]=a[k];
if(y[k][i][j]<ymin) ymin=y[k][i][j];
if(y[k][i][j]>ymax) ymax=y[k][i][j];
}
}
}
}

public timeplot_n(f_xt f,double x0,double xn,double dx,double t0,double tn,double dti,double dt_reali)
{dt=dti;
dt_real=dt_reali;
nx=(int)((xn-x0)/dx+1);
nt=(int)((tn-t0)/dt_real+1);
double a[]=new double[1];
n=1;
xmin=x0;
xmax=xn;
ymin=9.99e50;
ymax=-9.99e50;
x=new double[n][nt][nx];
y=new double[n][nt][nx];
double xx=0;
double t;
for(int i=0;i<nt;i++)
{for(int j=0;j<nx;j++)
{xx=x0+dx*j;
t=t0+dt_real*i;
a[0]=f.func(xx,t);
for(int k=0;k<n;k++)
{x[k][i][j]=xx;
y[k][i][j]=a[k];
if(y[k][i][j]<ymin) ymin=y[k][i][j];
if(y[k][i][j]>ymax) ymax=y[k][i][j];
}
}
}
}
public timeplot_n(double yi[][][],double x0,double xn,double dti,double dt_reali)
{nx=yi[0].length;nt=yi.length;
dt=dti;
dt_real=dt_reali;
xmin=x0;
xmax=xn;
ymin=9.99e50;
ymax=-9.99e50;
x=new double[n][nt][nx];
y=new double[n][nt][nx];
```

```
double dx=(xn-x0)/(nx-1);
for(int k=0;k<n;k++)
{
for(int i=0;i<nt;i++)
{for(int j=0;j<nx;j++)
{x[k][i][j]=x0+dx*j;
y[k][i][j]=yi[k][i][j];
if(y[k][i][j]<ymin) ymin=y[k][i][j];
if(y[k][i][j]>ymax) ymax=y[k][i][j];
}
}
}
}

public void run()
{
Plot pp=new Plot(x[0][0],y[0][0]);
for(int i=0;i<nt;i++)
try {
pp.ppA(x[0][i],y[0][i],0,0,0,255);
for(int k=1;k<n;k++)
{pp.addData(x[k][i],y[k][i]);}
pp.setPlabel("t = "+i*dt_real+" s");
pp.setMinMax(xmin,xmax,ymin,ymax);
int t=(int)(1000*dt);
pp.plot();
Thread.sleep(t);
pp.dispose();
}
catch(InterruptedException e)
{System.err.println(e.toString());}
}

public void plot() {start();}
}
```

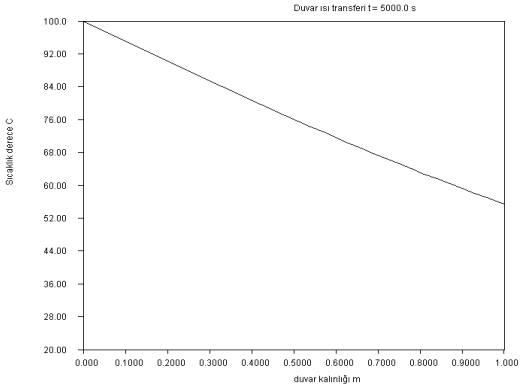
Örnek problemimizde zamana bağlı bir ısı transferi problemi verdik. Elbette kitapta zamana bağlı etkiyi gösteremeyiz, bunun yerine iki ayrı andaki grafiği vereceğiz.

Program 1.5-2 zamana bağlı grafik programı örneği timeplottest

```
class i_wall extends f_xt
{
public double func(double x,double t)
{double Ti=20.0;
double Ts=100.0;
double alfa=17.7e-4;
double T=Ts+(Ti-Ts)*Mathd.erf(x/(2*Math.pow((alfa*t),0.5)));
return T;
}
}
public class timeplottest
{
public static void main(String args[])
{
i_wall ff=new i_wall();
timeplot_n p=new timeplot_n(ff,0.0,1.0,0.05,0.1,1000.0,0.5,10.0);
p.plot();
}
}
```



(bolum1_014.jpg,Resimaltı: timeplottest çıktısı, t=1200 s)



(bolum1_015.jpg,Resimaltı: timeplottest çıktısı, t=5000 s)

Gerçek zaman grafik ortamı dediğimizde aynı zamanda fonksiyonları gerçek zamanda dinamik olarak oluşturup derhal görebileceğimiz ortamlar da anlaşılabilir. Bunu yapabilmek için programların gerçek zamanda derlenerek yeni sınıfların oluşturulması ve kullanılması gereklidir. Java bize bu imkanı sunan bir programlama dilidir.

Program 1.5-3 gerçek zaman grafik programı class ce1

```
import java.io.*;
import javax.swing.*;
import javax.tools.*;
import java.util.*;
import java.lang.Integer;
import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;
import java.awt.geom.*;
import java.awt.image.*;
import javax.swing.*;
import java.util.Locale;
import java.text.*;
import java.util.*;
import javax.swing.table.*;
import static java.lang.Math.*;

class ce1
{       String s;
            public  ce1()
            {input g=new input("f2D"," ");
            s=g.vericiktisi();
            try{ g.finalize();}
            catch(Throwable e1) {System.err.println("input error.");}
            }

            public  ce1(String function)
            {       functiongenerator g=new functiongenerator("f2D",function);
            s=g.createfunction();
            try{ g.finalize();}
            catch(Throwable e1) {System.err.println("input error.");}
            }

            public static void main(String arg[])
            { //String s1=JOptionPane.showInputDialog(" Fonksiyonu giriniz : ");
                        ce1 x0=new ce1();
```

```
double a[]=Text.readDoubleV("Enter minimum and maximum values of the function xmin xmax");
ce3 x1=new ce3(a[0],a[1],400,x0.s);
//ce8 x2=new ce8(a[0],a[1],x0.s);
        }
}
```

Program 1.5-4 gerçek zaman grafik programı ce3

```
import java.io.*;
import javax.swing.*;
import javax.tools.*;
import java.util.*;
import java.lang.Integer;
import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;
import java.awt.geom.*;
import java.awt.image.*;
import javax.swing.*;
import java.util.Locale;
import java.text.*;
import java.util.*;
import javax.swing.table.*;
import static java.lang.Math.*;

class ce3 extends JApplet
{
        public ce3(double x1,double x2,int n,String s)
        {
        Plot pp=new Plot(new f2D(),x1,x2,n);
        pp.setPlabel("f(x)="+s);
        pp.plot();
        }

        public ce3(double x1,double x2,int n)
        {Plot pp=new Plot(new f2D(),x1,x2,n);
        pp.plot();
        }

        public ce3(double x1,double x2)
        {Plot pp=new Plot(new f2D(),x1,x2,50);
        pp.plot();
        }

        public static void main(String arg[])
        {double a=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun alt sinirini giriniz x_minimum
        : "));
double b=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun üst sinirini giriniz x_maksimum : "));
        ce3 xx=new ce3(a,b,400);}
}
```

Program 1.5-5 gerçek zaman grafik girdi sınıfı: input

```
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.*;
import javax.swing.event.*;
import java.io.*;
import java.util.*;
```

```
public class input extends Thread implements ActionListener
{
        Container c;
        String s=" ";
        String s1="";
        functiongenerator fg;
        JTextArea jt;
        JButton bb;
        boolean basildi;

        public input(String isim,String fonksiyon)
        {
        jt=new JTextArea(s,5,30);
        jt.setFont(new Font("SansSerif",Font.PLAIN,20));
        fg=new functiongenerator();
        bb=new JButton("push button after entering the function into to the window : x");
        JScrollPane jsp=new JScrollPane(jt);
JFrame cerceve=new JFrame("function input window example: Math.sin(x)-1/(x*x-1) ");
cerceve.addWindowListener(new BasicWindowMonitor());
c=cerceve.getContentPane();
c.setLayout(new BorderLayout());
c.add(jsp,BorderLayout.NORTH);
c.add(bb,BorderLayout.SOUTH);
bb.addActionListener(this);
cerceve.pack();
cerceve.setVisible(true);
basildi=false;
}

public String vericiktisi()
{
while(!basildi)
{
try {Thread.sleep(100);}
catch(InterruptedException e)  {System.err.println(e.toString());}
}
        return s1;
        }

protected void finalize() throws Throwable
{super.finalize();}

public void actionPerformed( ActionEvent e)
{
if(e.getSource()==bb)
{s=jt.getText();fg.addfunction(s);s1=fg.createfunction();
basildi=true;
try{fg.finalize();finalize();
} catch(Throwable e1) {System.err.println("girdi hatasi.");}
}}}
```

Program 1.5-6 gerçek zaman grafik girdi sınıfı: functiongenerator

```
mport java.io.*;
import javax.swing.*;
import javax.tools.*;
import java.util.*;
import java.lang.Math.*;
```

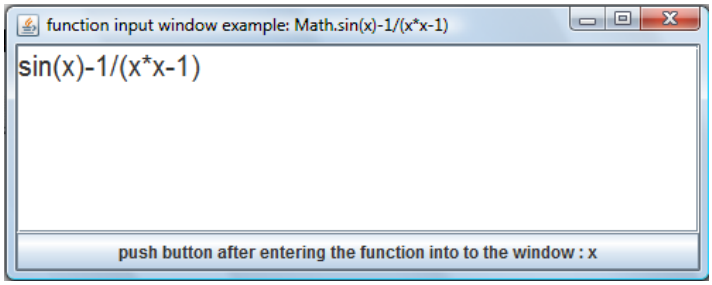
```
import java.lang.*;

class functiongenerator
{
String fonksiyon;
String dosyaismi;
String sinif;
PrintStream fout;
PrintStream ferr;

public functiongenerator(String dosyaismi,String fonksiyon)
{ this.fonksiyon=fonksiyon;
this.dosyaismi=dosyaismi;
try {
fout=new PrintStream(new FileOutputStream("print.txt"));
ferr=new PrintStream(new FileOutputStream("error.txt"));
}catch(FileNotFoundException e1) {System.err.println("dosya bulunamadi");}
}

public functiongenerator(String fonksiyon)
{this.fonksiyon=fonksiyon;
dosyaismi="f2D";
}

public void addfunction(String fonksiyon)
{this.fonksiyon=fonksiyon;}

public String toString()
{return fonksiyon;}

public functiongenerator()
{fonksiyon="x*x-4.0";
dosyaismi="f2D";
}

public  String createfunction()
{String isim=dosyaismi+".java";
String s="";
try {PrintWriter ffout=new PrintWriter(new BufferedWriter(new FileWriter(isim)));
s+="class "+dosyaismi+" extends f_x\n";
s+=" {\n";
s+=" public double func (double x)\n";
s+="{ double y="+fonksiyon+";\n";
s+=" return y;\n";
s+="}\n";
s+="}\n";
sinif=s;
ffout.println(s);
//System.out.println(s);
ffout.close();
}
catch(IOException e1) {System.err.println("girdi hatasi.");}
catch(NumberFormatException e2){};

try{
JavaCompiler javac = ToolProvider.getSystemJavaCompiler();
int rc = javac.run(null,null,null,isim);}
catch(NullPointerException e1) {System.err.println("NULL POINTER");}
```

```
return fonksiyon;
}

protected void finalize() throws Throwable
{super.finalize();}
}
```

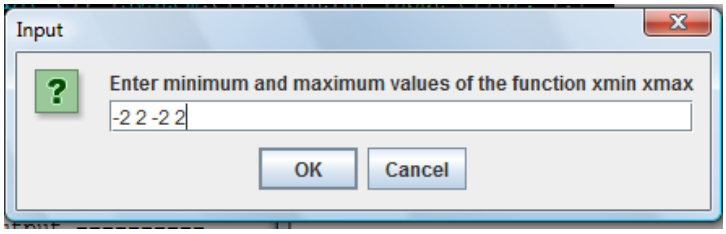
Gerçek zaman programı, ce1 tek başına java komutuyla çalıştırılacağı gibi küçük bir bat dosyası hazırlanarak ekrandan grafik sembolü üzerine çift tıklama ile de çalıştırılabilir.

Program 1.5-7 gerçek zaman program çalıştırma bat dosyası G2D.bat

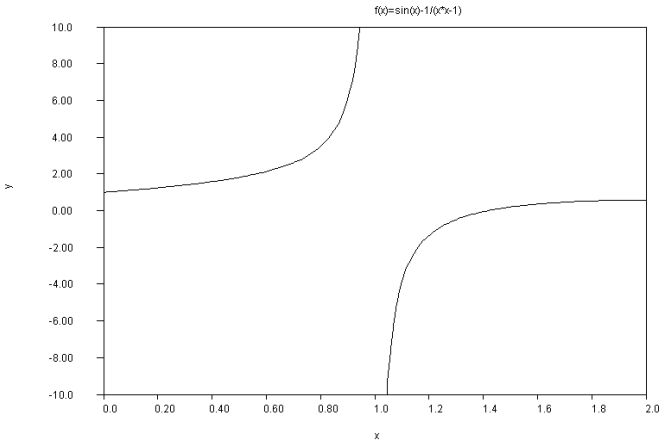
```
java ce1
```



(bolum1_016.jpg,Resimaltı: G2D girdi penceresi)



(bolum1_017.jpg,Resimaltı: G2D girdi penceresi)



(bolum1_018.jpg,Resimaltı: G2D grafik çıktısı)

Aynı tür bir gerçek zaman programlama ortamı 3 boyutlu progamlama için de sağlanabilir.

Program 1.5-8 gerçek zaman 3D Plot programı ce5

```
import java.io.*;
import javax.swing.*;
import javax.tools.*;
import java.util.*;
import java.lang.Integer;
import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;
```

```
import java.awt.geom.*;
import java.awt.image.*;
import javax.swing.*;
import java.util.Locale;
import java.text.*;
import java.util.*;
import javax.swing.table.*;
import visad.*;
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;

class ce5 extends JApplet
{
        public  ce5()
        {
        input3 g=new input3("f3D"," ");
        String s=g.vericiktisi();
        try{g.finalize();} catch(Throwable e1)
        {System.err.
println("input hatas1Į");}
        }

        public  ce5(String fonksiyon)
        {//Text.create3Dfunction(s1);
        functiongenerator3D g=new functiongenerator3D(fonksiyon);
        String s=g.createfunction();
        try{
g.finalize();
} catch(Throwable e1) {System.err.println("input hatas1Į");}
}

        public static void main(String arg[]) throws RemoteException,VisADException
        {
        ce5 xx=new ce5();
        double a[]=Text.readDoubleV("x_min x_max y_min y_max (enter 4 value)");
        ce6 xy=new ce6(a[0],a[1],a[2],a[3],200);
        }
}
```

Program 1.5.9 gerçek zaman 3D Plot sınıfı: ce6

```
import visad.*;
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class ce6
{    public ce6(double xmin,double xmax,double ymin,double ymax,int n)throws RemoteException,VisADException
        {f3D f=new f3D();
Plot3D.func(f,xmin,xmax,ymin,ymax);
}
public static void main(String arg[]) throws
s RemoteException,VisADException
        {
        double a[]=Text.readDoubleV("x_min x_max y_min y_max (4 değer giriniz)");
        ce6 xy=new ce6(a[0],a[1],a[2],a[3],200);
        }
}
```

Program 1.5.10 gerçek zaman 3D Plot sınıfı: functiongenerator3D

```
import java.io.*;
```

```
import javax.swing.*;
import javax.tools.*;
import java.util.*;

class functiongenerator3D extends Thread
{
String fonksiyon;
String dosyaismi;
String sinif;
public functiongenerator3D(String dosyaismi,String fonksiyon)
{ this.fonksiyon=fonksiyon;
this.dosyaismi=dosyaismi;
}
public functiongenerator3D(String fonksiyon)
{this.fonksiyon=fonksiyon;
dosyaismi="f3D";
}
public functiongenerator3D()
{fonksiyon="x[0]*x[0]+x[1]*x[1]-4.0";
dosyaismi="f3D";
}
public void addfunction(String fonksiyon)
{this.fonksiyon=fonksiyon;
}

public String toString()
{return fonksiyon;}

public  String createfunction()
{String isim=dosyaismi+".java";
String s="";
try {PrintWriter fout=new PrintWriter(new BufferedWriter(new FileWriter(isim)));
s+="class "+dosyaismi+" extends f_xj\n";
s+="{\n";
s+="public double func(double w[])\n";
s+="{//çözümü istenen fonksiyon\n";
s+="double x=w[0];\n";
s+="double y=w[1];\n";
s+="double ff;\n";
s+="ff="+fonksiyon+";\n";
s+="return ff;\n";
;
s+="}\n";
s+="}\n";
sinif=s;
fout.println(s);
//System.out.println(s);
fout.close();
}
catch(IOException e1) {System.err.println("girdi hatasi.");}
catch(NumberFormatException e2){};
JavaCompiler javac = ToolProvider.getSystemJavaCompiler();
int rc = javac.run(null, null, null,isim);
return fonksiyon;
}

protected void finalize() throws Throwable
{super.finalize();}

}
```

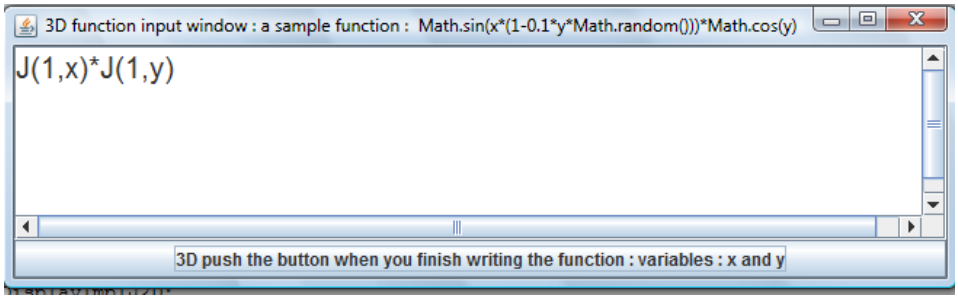
Program 1.5.11 gerçek zaman 3D Plot sınıfı: functiongenerator3D

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

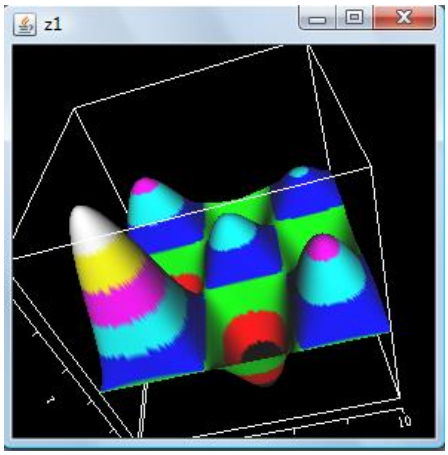
```
import javax.swing.table.*;
import javax.swing.event.*;
import java.io.*;
import java.util.*;

public class input3 extends Thread implements ActionListener
{
        Container c;
        String s=" ";
        String s1="";
        functiongenerator3D fg;
        JTextArea jt;
        JButton bb;
        boolean basildi;

        public input3(String isim,String fonksiyon)
        {
        jt=new JTextArea(s,5,40);
        jt.setFont(new Font("SansSerif",Font.PLAIN,20));
        fg=new functiongenerator3D();
        bb=new JButton("3D push the button when you finish writing the function : variables : x and y");
        JScrollPane jsp=new JScrollPane(jt);
JFrame cerceve=new JFrame("3D function input window : a sample function : Math.sin(x*(1-
0.1*y*Math.random()))*Math.cos(y) ");
cerceve.addWindowListener(new BasicWindowMonitor());
c=cerceve.getContentPane();
c.setLayout(new BorderLayout());
c.add(jsp,BorderLayout.NORTH);
c.add(bb,BorderLayout.SOUTH);
bb.addActionListener(this);
cerceve.pack();
cerceve.setVisible(true);
basildi=false;
}

public String vericiktisi()
{
while(!basildi)
{
try {Thread.sleep(100);}
catch(InterruptedException e)  {System.err.println(e.toString());}
}
        return s1;
        }

protected void finalize() throws Throwable
{super.finalize();}

public void actionPerformed( ActionEvent e)
{
if(e.getSource()==bb)
{s=jt.getText();fg.addfunction(s);s1=fg.createfunction();
basildi=true;
try{fg.finalize();finalize();
} catch(Throwable e1) {System.err.println("input error");}
}
}
}
```

Program 1.5.12 gerçek zaman 3D Plot bat dosyası: G3D.bat

```
javaw ce5
```



(bolum1_019.jpg,Resimaltı: G3D grafik fonksiyon girdi penceresi)



(bolum1_020.jpg,Resimaltı: G3D VisAD grafik çıktısı)

Önceki bölümde bahsettiğimiz ContourPlot programını kullanarak gerçek zaman fonksiyon grafikleri çizmek için de benzer bir program yazabiliriz.

Program 1.5.13 gerçek zaman 3D Contour Plot sınıfı: ce1DC.java

```
class ce1DC
{
        public static void main(String arg[])
        { ce5 x0=new ce5();
double a[]=Text.readDoubleV("enter minimum and maximum values of the function xmin xmax  ymin ymax");
ce3DC x1=new ce3DC(a,50,x0.s);
        }
}
```

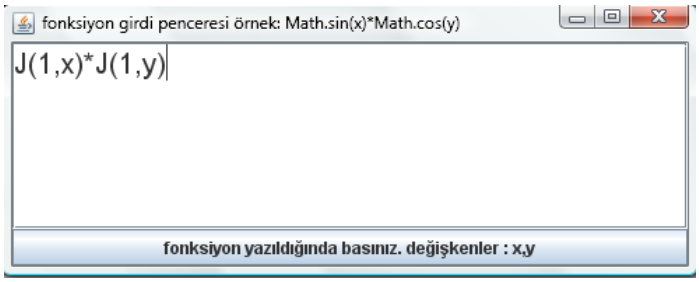
Program 1.5.14 gerçek zaman 3D Contour Plot sınıfı: ce3DC.java

```
import java.io.*;
import javax.swing.*;
import javax.tools.*;
import java.util.*;
import java.lang.Integer;
import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;
import java.awt.geom.*;
import java.awt.image.*;
import javax.swing.*;
import java.util.Locale;
import java.text.*;
import java.util.*;
import javax.swing.table.*;
import static java.lang.Math.*;

class ce3DC extends JApplet
{
        public ce3DC(double a[],int n,String s)
```

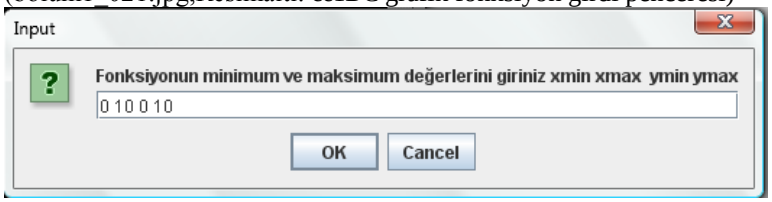
```
        { int n1=a.length;
        f3D fa=new f3D();  //
        if(n1==6) {ContourPlot pp = new ContourPlot(fa,a[0],a[1],(int)a[5],a[2],a[3],(int)a[6],n, false,s, "x", "y")
        pp.plot();}
        else {ContourPlot pp = new ContourPlot(fa,a[0],a[1],100,a[2],a[3],100,n, false,s, "x", "y");
                pp.plot();}
        }
}
```

Program 1.5-15 gerçek zaman 3D Contour Plot bat dosyası: G3D.bat

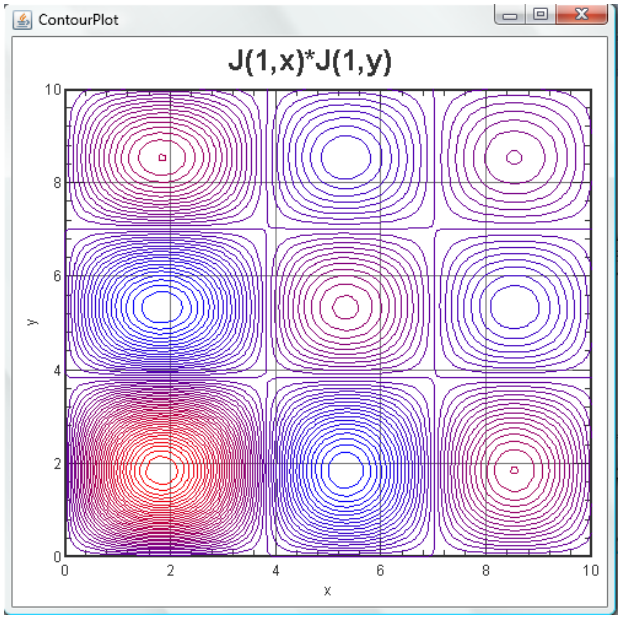
```
d:\java\bin\javaw ce1DC
```



(bolum1_021.jpg,Resimaltı: ce1DC grafik fonksiyon girdi penceresi)



(bolum1_022.jpg,Resimaltı: ce1DC grafik fonksiyon minimum maksimum değerleri girdi penceresi)



(bolum1_023.jpg,Resimaltı: ce1DC grafik çıktı penceresi)

Grafik fonksiyonuna fonksiyonu direk olarak yazarak grafikleri çizmek hiçbir programlama gerektirmediğinden yeni başlayanlar için başlangıç açısından yararlı olabilecek bir işlemdir. Gerçek zaman programlamayı Numeric.java sınıfındaki setX ve setY statik metodlarını kullanarak da yapabiliriz.

Program 1.5.16a Numeric.java sınıfını kullanarak gerçek zaman veri hazırlayıp grafiğini çizme

```
import java.io.*;
public class Numerictest
```

```
{
public static void main(String arg[]) throws IOException
{
double x[]=Numeric.setX(0.0,0.1,10.0);
double y[]=Numeric.setY("sin(x)",x);
Plot.data(x,y);
}}
```

Program 1.5-14b Numeric.java sınıfını kullanarak gerçek zaman veri hazırlayıp grafiğini çizme

```
import java.io.*;
public class Numerictest3
{
public static void main(String arg[]) throws IOException
{ Numeric.plot("sin(x)",0.0,0.1,10.0);}}
```

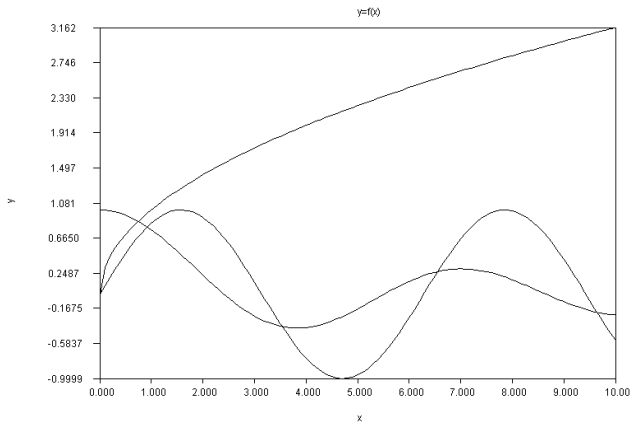
Numeric sınıfını kullanarak birden fazla fonksiyonu aynı anda çizmek de mümkündür.

Program 1.5-14d Numeric.java sınıfını kullanarak gerçek zaman veri hazırlayıp grafiğini çizme

```
import java.io.*;
public class Numerictest4
{
public static void main(String arg[]) throws IOException
{   double x[]=Numeric.setX(0.0,0.1,10.0);
String s[]={"sin(x)","J(0,x)","sqrt(x)"};
Numeric.plot(s,x);}}
```

Program 1.5-14e Numeric.java sınıfını kullanarak gerçek zaman veri hazırlayıp grafiğini çizme

```
import java.io.*;
public class Numerictest5
{
public static void main(String arg[]) throws IOException
{   String s[]={"sin(x)","J(0,x)","sqrt(x)"};
Numeric.plot(s,0.0,0.1,10.0);}}
```



(bolum1_024.jpg,Resimaltı: NumericTest5 grafik çıktısı)

Aynı kavram kullanılarak 3 boyutlu Contour plotlarda direk olarak programınızın içinde ayrı bir fonksiyon tanımlanmadan çizilebilir.

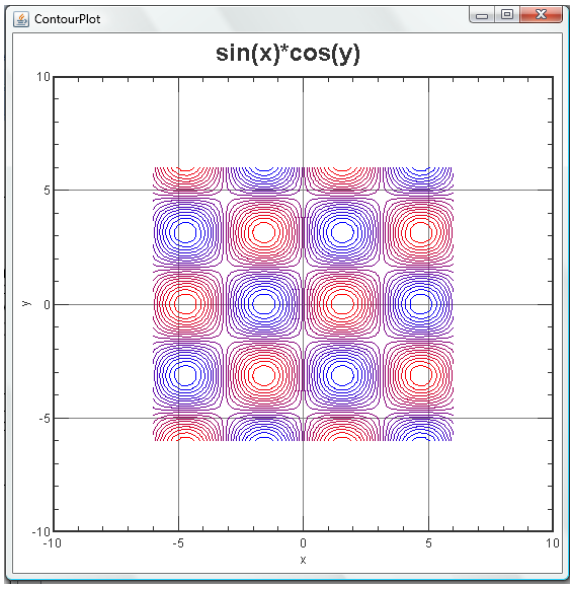
Program 1.5.15 gerçek zaman contour plot çizim programı Plot3DCS.java

```
//=====================================================
// Numerical Analysis package in java
// PlotS class to plot 3D function
```

```
// with real time function generation (function input as
// String variable uses ContourPlot class
// Dr. Turhan Coban
// EGE University, Engineering School, Mechanical Eng. Div.
// turhan.coban@ege.edu.tr
// ================================================
import java.io.*;
import java.awt.Color;
import static java.lang.Math.*;

public class Plot3DCS
{ ce5 x0;  // write function f3D compile it as a java file
ce3DC x1;  // plot function f3D by using ContourPlot class
String ss;
double a[];
int N,ipt,ir,ig,ib;

public Plot3DCS(String s,double ai[],int N)
{
        ce5 x0=new ce5(s);
        ss=s;
        a=new double[4];
        for(int i=0;i<4;i++) {a[i]=ai[i];}
this.N=N;
}

public void plot()
{ x1=new ce3DC(a,N,ss);}

public static void func(String s,double a[],int N)
{Plot3DCS ps=new Plot3DCS(s,a,N);
ps.plot();
}
public static void func(String s,double a[])
{Plot3DCS ps=new Plot3DCS(s,a,20);
ps.plot();
}
}
```

Program 1.5-16 gerçek zaman direk grafik çizim örnek programı

```
import java.io.*;
class NA03DCS
{
public static void main (String args[]) throws IOException
{       double a[]={-6,6,-6,6};
         Plot3DCS.func("sin(x)*cos(y)",a);
}
}
```



(bolum1_025.jpg,Resimaltı: NA03DCS grafik çıktısı)

Benzer uygulamayı visAD paketini kullanan 3D grafiklere de uygulayabiliriz.

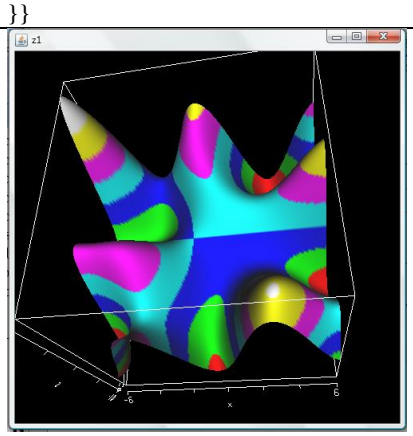
Program 1.5-17 visAD paketini kullanarak 3D grafik çizimi yapan gerçek zaman programı

```
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Plot3DS
{ ce5 x0;  // write function f3D compile it as a java file
ce6 x1;  // plot function f3D by using Plot3D class
String ss;
double a[];
int N,ipt,ir,ig,ib;

public Plot3DS(String s,double ai[],int N)
{   ce5 x0=new ce5(s);  ss=s;  a=new double[4];
for(int i=0;i<4;i++) {a[i]=ai[i];}
this.N=N;
}
public void plot() throws RemoteException,VisADException
{ x1=new ce6(a,N);}
public static void func(String s,double a[],int N) throws RemoteException,VisADException
{Plot3DS ps=new Plot3DS(s,a,N);
ps.plot();
}
public static void func(String s,double a[]) throws RemoteException,VisADException
{Plot3DS ps=new Plot3DS(s,a,20); ps.plot(); } }
```

Program 1.5.18 visAD paketini kullanarak 3D grafik çizimi yapan gerçek zaman programı test örneği

```
import javax.tools.*;
import java.util.*;
import java.lang.Integer;
import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;
import java.awt.geom.*;
import java.awt.image.*;
```

```
import javax.swing.*;
import java.util.Locale;
import java.text.*;
import java.util.*;
import javax.swing.table.*;
import visad.*;
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;
class NA03DS
{ public static void main (String args[]) throws RemoteException,VisADException
{double a[]={-6,6,-6,6};
Plot3DS.func("sin(x)*y+cos(y)*x",a,20);
}}
```



(bolum1_026.jpg,Resimaltı: NA03DS VisAD grafik çıktısı)

1.4 Grafik ortamından veri okuma

Sayısal analiz'de gerekli olabilecek alt yapıladan birisi de grafik ortamından veri okumaktır. Günümüzde grafikleri Scanner denilen aletlerle her zaman digitize edebilmemiz mümkün olmaktadır. Burada oluşturduğumuz program jpg formatında digitize edilmiş bir resmin üzerindeki grafiği fare kullanarak veriye dönüştürerek text dosyası haline çevirmeye yarayan bir programdır.

Program 1.6-1 scanP.java

```
import java.io.*;
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.event.*;
import java.awt.geom.*;
import java.net.URL;
import java.awt.image.*;
import java.util.*;

public class  scanP extends JPanel implements MouseListener,MouseMotionListener,MouseWheelListener
{
int    x1[],y1[];
public JLabel jp;
public double a[][];
int n;
boolean mousemoved;
Rectangle2D l;
Ellipse2D l1,l2;
Color c;
Image img;
public PrintWriter fout;
String dosya;
```

```
//BufferedWriter fout;
double xmin,xmax,ymin,ymax;
int    xcmin,xcmax,ycmin,ycmax;

public scanP(String p1, String idosya,double ixmin,double ixmax,double iymin,double iymax)
{super();
dosya=idosya;
try {fout=new PrintWriter(new BufferedWriter(new FileWriter(dosya))); }
catch(IOException e1) {System.err.println("input error.");}
catch(NumberFormatException e2){}
URL url = getClass().getResource(p1);
img = getToolkit().getImage(url);
xmin=ixmin;
xmax=ixmax;
ymin=iymin;
ymax=iymax;
jp=new JLabel();
xcmin=(int)xmin;
ycmin=(int)ymin;
xcmax=(int)xmax;
ycmax=(int)ymax;
x1=new int[1000];
y1=new int[1000];
a=new double[2][1000];
n=-2;
addMouseListener(this);
addMouseMotionListener(this);
addMouseWheelListener(this);
mousemoved=false;
}
public void setMinMax(double ixmin,double ixmax,double iymin,double iymax)
{  xmin=ixmin;
xmax=ixmax;
ymin=iymin;
ymax=iymax;
}
public JLabel getLabel()
{  return jp;
}

public void setDosyaismi(String Di)
{  dosya=Di;
try {fout=new PrintWriter(new BufferedWriter(new FileWriter(dosya))); }
catch(IOException e1) {System.err.println("input error.");}
catch(NumberFormatException e2){}}

public void setResimismi(String Ci)
{  URL url = getClass().getResource(Ci);
img = getToolkit().getImage(url);}

public void setCizgi(int x1i,int y1i)
{if(n==-2) {xcmin=x1i;ycmin=y1i;}
else if(n==-1) {xcmax=x1i;ycmax=y1i;}
else
{x1[n]=x1i;y1[n]=y1i;
a[0][n]=xmin+(xmax-xmin)/(xcmax-xcmin)*(x1i-xcmin);
a[1][n]=ymin+(ymax-ymin)/(ycmax-ycmin)*(y1i-ycmin);
System.out.println(a[0][n]+" "+a[1][n]);
fout.println(a[0][n]+" "+a[1][n]);
}}

public void paintComponent(Graphics g)
{
//super.paintComponent(g);
Graphics2D g2=(Graphics2D)g;
g2.setFont(new Font("Serif",Font.BOLD,24));
g2.setColor(Color.blue);
```

```
g2.setStroke(new BasicStroke(2.0f));
Dimension d=getSize();
g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,RenderingHints.VALUE_ANTIALIAS_ON);
int dx = d.width;
int dy = d.height;
g2.drawImage(img, 0, 0,dx,dy, this);
if(n>=-1)
{
l1=new Ellipse2D.Double(xcmin-2,ycmin-2,5,5);
l2=new Ellipse2D.Double(xcmax-2,ycmax-2,5,5);
g2.draw(l1);
g2.draw(l2);
}
for(int i=0;i<n;i++)
{l=new Rectangle2D.Double(x1[i]-1,y1[i]-1,3,3);
g2.draw(l);
}
}

//MouseListener (fare dinleyicisi)
public void mouseClicked(MouseEvent e)
{}

public void mouseclose()
{if(!mousemoved)
{fout.close();
Text.print(dosya,"digitiser output data : "+dosya+" file ");
mousemoved=true;
}
}

public void mousePressed(MouseEvent e)
{  if(e.isMetaDown()) {mouseclose();} //when right button is pushed
else if(e.isAltDown())            //when middle button is pushed
{}
else { setCizgi(e.getX(),e.getY());n++;repaint();}  //when left button is pushed
}

public void mouseReleased(MouseEvent e)
{ }

public void mouseEntered(MouseEvent e)
{ }

public void mouseExited(MouseEvent e)
{ }
//MouseMotionListener methods

public void mouseDragged(MouseEvent e)
{   }

public void mouseMoved(MouseEvent e)
{ double x1,y1;
x1=xmin+(xmax-xmin)/(xcmax-xcmin)*(e.getX()-xcmin);
y1=ymin+(ymax-ymin)/(ycmax-ycmin)*(e.getY()-ycmin);
jp.setText("x = "+x1+"y = "+y1);
}

public void mouseWheelMoved(MouseWheelEvent e)
{ }

}
```

Program 1.6-2 digitizer.java

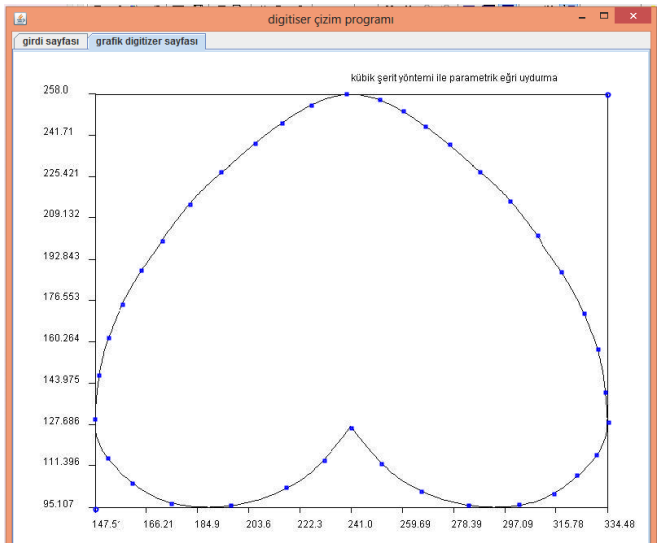
```
import javax.swing.*;
import java.awt.*;
```

```
import java.awt.event.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.filechooser.*;

public class digitizer extends JApplet implements ActionListener
{
private JButton B[];
private JLabel l[];
JLabel altbar;
private JTextField t[];
private JTextArea ta;
private String S[]={"input page","graphic digitizing page"};
private Container c;
private JPanel J[];
scanP fp;
double xmin,xmax,ymin,ymax;
String resim,dosya;
JTabbedPane jtp;

public void init()
{
        xmin=0.0;
        xmax=10.0;
        ymin=0.0;
        ymax=10.0;
        resim="kalp.JPG";
        dosya="a.txt";
c=getContentPane();
c.setLayout(new BorderLayout(5,5));
B=new JButton[2];
l=new JLabel[8];
t=new JTextField[6];
altbar=new JLabel();
ta=new JTextArea(toString());
jtp=new JTabbedPane();
J=new JPanel[3];
J[0]=new JPanel();
J[0].setLayout(new GridLayout(4,4));
J[1]=new JPanel();
J[2]=new JPanel();
B[0]=new JButton("JFileChoser reading picture");
B[1]=new JButton("JFileChoser output file");
l[0]=new JLabel("minimum x");
l[1]=new JLabel("minimum y");
l[2]=new JLabel("maximum x");
l[3]=new JLabel("maxsimum y");
l[4]=new JLabel("graphic  (input) file");
l[5]=new JLabel("graphic (output) file");
l[6]=new JLabel("graphic  (input) file");
l[7]=new JLabel("graphic (output) file");
t[0]=new JTextField(""+xmin);
t[1]=new JTextField(""+ymin);
t[2]=new JTextField(""+xmax);
t[3]=new JTextField(""+ymax);
t[4]=new JTextField(resim);
t[5]=new JTextField(dosya);
for(int i=0;i<5;i++) t[i].addActionListener(this);
for(int i=0;i<2;i++) B[i].addActionListener(this);
J[0].add(l[0]);
J[0].add(t[0]);
J[0].add(l[1]);
J[0].add(t[1]);
J[0].add(l[2]);
J[0].add(t[2]);
```

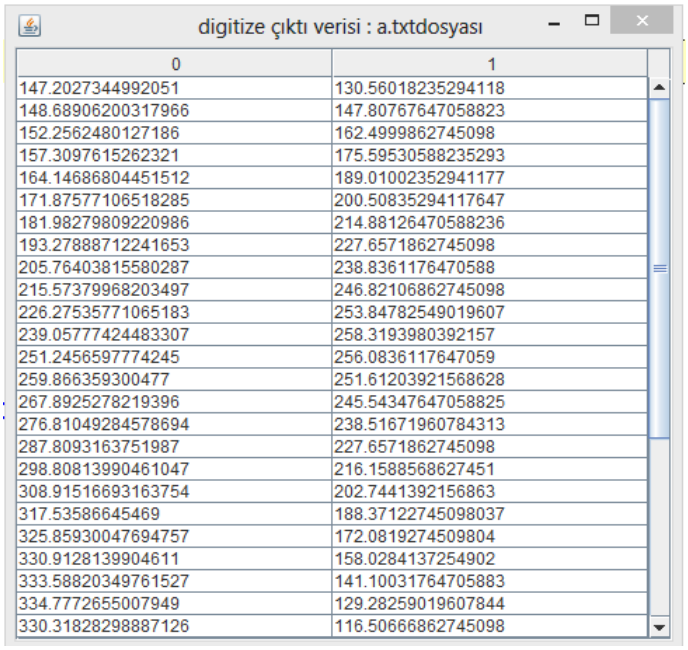
```
J[0].add(l[3]);
J[0].add(t[3]);
J[0].add(l[4]);
J[0].add(t[4]);
J[0].add(l[5]);
J[0].add(t[5]);
J[0].add(l[6]);
J[0].add(B[0]);
J[0].add(l[7]);
J[0].add(B[1]);
J[1].add(J[0],BorderLayout.NORTH);
J[1].add(ta,BorderLayout.CENTER);
jtp.addTab(S[0],J[1]);
fp=new scanP(resim,dosya,xmin,xmax,ymin,ymax);
//fp.setSize(1000,600);
//J[2].setLayout(new FlowLayout());
J[2].add(fp,BorderLayout.CENTER);
//J[2].add(fp.jp,BorderLayout.SOUTH);
jtp.addTab(S[1],fp);
c.add(jtp);
}

public String toString()
{String s;
s="              Turhan ÇOBAN EGE Üniversitesi, Mühendislik Fak. Makina Müh.\n";
s+="=============================================================================================\n"
;
s+="Command structure : first two tkics to mouse defines minimum and maximum data points\n";
s+="After this each left mouse click converted to x and y coordinates according to given maximum \n";
s+="and minimum values. A picture of the graphics placed on the screen as a picture\n";
s+="when right hand button of mouse is pushed process stops and all the values are recorded \n";
s+="into the given file.\n";
s+="=============================================================================================\n"
;
return s;
}

public void actionPerformed(ActionEvent e)
{
if(e.getSource()==t[0])
{      Double V0=new Double(t[0].getText());
xmin=V0.doubleValue();
}
else if(e.getSource()==t[1])
{      Double V0=new Double(t[1].getText());
ymin=V0.doubleValue();
}
else if(e.getSource()==t[2])
{      Double V0=new Double(t[2].getText());
xmax=V0.doubleValue();
}
else if(e.getSource()==t[3])
{      Double V0=new Double(t[3].getText());
ymax=V0.doubleValue();
}
else if(e.getSource()==t[4])
{      resim=t[4].getText();
fp.setResimismi(resim);}
else if(e.getSource()==t[5])
{      dosya=t[4].getText();
fp.setDosyaismi(dosya);
}
else if(e.getSource()==B[0])
{      File file=new File("kalp.jpg");
            JFileChooser fc=new JFileChooser();
if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {file = fc.getSelectedFile();resim=file.getName();}
t[4].setText(resim);
```

```
fp.setResimismi(resim);
}
else if(e.getSource()==B[1])
{       File file=new File("a.txt");
            JFileChooser fc=new JFileChooser();
if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {file = fc.getSelectedFile();dosya=file.getName();}
t[5].setText(dosya);
fp.setDosyaismi(dosya);
}
System.out.println("xmin = "+xmin+"xmax="+xmax+"ymin="+ymin+"ymax="+ymax);
fp.setMinMax(xmin,xmax,ymin,ymax);
repaint();
}


public static void main(String s[])
{
JFrame f = new JFrame("digitizer program");
JApplet applet = new digitizer();
applet.init();
f.add(applet);
f.pack();
f.setSize(800,1000);
f.setVisible(true);
}
}
```

(bolum1_027.jpg,Resimaltı: digitizer çizim programı)



(bolum1_028.jpg,Resimaltı: digitizer çizim programı grafik girdi penceresi)



0	1
147.2027344992051	130.56018235294118
148.68906200317966	147.80767647058823
152.2562480127186	162.4999862745098
157.3097615262321	175.59530588235293
164.14686804451512	189.01002352941177
171.87577106518285	200.50835294117647
181.98279809220986	214.88126470588236
193.27888712241653	227.6571862745098
205.76403815580287	238.8361176470588
215.57379968203497	246.82106862745098
226.27535771065183	253.84782549019607
239.05777424483307	258.3193980392157
251.2456597774245	256.0836117647059
259.866359300477	251.61203921568628
267.8925278219396	245.54347647058825
276.81049284578694	238.51671960784313
287.8093163751987	227.6571862745098
298.80813990461047	216.1588568627451
308.91516693163754	202.7441392156863
317.53586645469	188.37122745098037
325.85930047694757	172.0819274509804
330.9128139904611	158.0284137254902
333.58820349761527	141.10031764705883
334.7772655007949	129.28259019607844
330.31828298887126	116.50666862745098

(bolum1_029.jpg,Resimaltı: digitizer çizim programı dosya çıktısı)

Program başladığında, JPG resminin minimum ve maksimum noktaları seçilmeli ve değerleri verilen pencerelere girilmelidir. Bundan sonra fare ile çizginin üzerindeki noktalar işaretlendiğinde (sol klik) bu noktalar digitize edilecek ve farenin sağ tarafı kliklendiğinde ismi tanımlanmış olan dosyaya kaydedilecektir.

1.5 Taylor serileri ve hata terimi

Taylor serileri bir çok sayısal analiz metodunun ve bir yerde sayısal analizin temelini teşkil eder. Taylor serisi bir fonksiyonun a noktası civarında seriye açılımını verir. Sonsuz sayıdaki seri terimleri fonksiyonu ve fonksiyonun türevlerini kullanır.

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + ..$$

Veya seri formunda yazarsak:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Eğer seri a=0 noktası civarında açılırsa McLaurin serisi adını alır. Bu seri adını İskoçlu matematikçi Colin MacLaurin'den almıştır.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!}x^n$$

Eğer Taylor serisindeki x, x^{k+1} ve a x^k ile değiştirilirse, seri

$f(x^{k+1}) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x^k)}{n!}(x^{k+1} - x^k)^n$ formunu alır. Bilgisayarlar aslında fonksiyonları hesaplayamazlar, sadece toplama işlemini, ve bu işlem üzerinden çarpma işlemini yapabilirler (.çıkarma ve bölme de bu işlemlerin değişik tanımlamasıdır). Örneğin $f(x)=e^x$ fonksiyonunu MacLaurin serisine açarsak

$$f(x) = e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} .. = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} ..$$

Formunu alır. Bilgisayarda bu fonksiyonu kullanarak exponansiyeli hesaplayabiliriz. Elbette sonsuz terim kullanmamız mümkün değildir. Serinin pratikte sonlu bir kısmı kullanılır. Şimdi bu seride sonsuz terim yerine sonlu terim kullanmanın oluşturacağı hatayı irdelersek, örneğin seriyi ilk 4 terimine açtığımızda

$$f(x) = e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24}$$

Serinin hatası geriye kalan ihmal ettiğimiz terimlere eşit olacaktır.

$$\text{Hata} = = \sum_{n=5}^{\infty} \frac{f^{(n)}(x^k)}{n!} x^{k+1}$$

Xin 0 a yakın değerleri için hatayı yaklaşık olarak

$Hata \cong \frac{x^5}{120}$ şeklinde yazılabilir. Bu form bir çok fonksiyon için hata analizinde kullanılabilir.

Taylor formülünün sayısal analiz metodlarında kullanmak için bir örnek daha verelim. Taylor formülünü ilk iki terim için açacak olursak ve fonksiyon değerini sıfıra eşitlersek

$$f(x^{k+1}) = 0 = f(x^k) + f'(x^k)(x^{k+1} - x^k)$$

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$$

Newton-Raphson kök bulma formülü dediğimiz formu elde ederiz.

1.6 Çeşitli fonksiyonların seri açılımları

Bir önceki alt bölümümüzde bilgisayarların fonksiyonları sadece seri çözümlerini kullanarak hesapladıklarını söylemiştik. Bu yüzden yaygın olarak kullanılan bazı fonksiyonların seri çözümlerini giriş bölümümüzün son alt bölümünde vermeyi uygun görüyoruz.

1. $\ln(x) = 2\sum_{k=1}^{\infty} \frac{1}{2k-1}\left(\frac{x-1}{x+1}\right)^{2k-1} \quad [0 < x]$
2. $\exp(x) = \sum_{k=1}^{\infty} \frac{x^k}{k!}$
3. $\exp(-x) = \sum_{k=1}^{\infty} (-1)^k \frac{x^k}{k!}$
4. $\exp(-x^2) = \sum_{k=1}^{\infty} (-1)^k \frac{x^{2k}}{k!}$
5. $\exp(x)(1+x) = \sum_{k=1}^{\infty} \frac{x^k(k+1)}{k!}$
6. $a^x = \sum_{k=1}^{\infty} \frac{[x\ln(a)]^k}{k!}$
7. $\sum_{k=0}^{\infty} a^{kx} = \frac{1}{1-a^x} \quad [a > 1 \ and \ \ x < 0 \ \ or \ 0 < a < 1 \ \ x > 0$
8. $\sin(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$
9. $\sinh(x) = \sum_{k=0}^{\infty} \frac{x^{2k+1}}{(2k+1)!}$
10. $\cos(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$
11. $\cosh(x) = \sum_{k=0}^{\infty} \frac{x^{2k}}{(2k)!}$

12. $\tan(x) = \dfrac{\sin(x)}{\cos(x)}$

13. $\tanh(x) = \dfrac{\sinh(x)}{\cosh(x)}$

14. $\arcsin(x) = \sum_{k=0}^{\infty} \dfrac{(2k)!}{2^{2k}(k!)^2(2k+1)} x^{2k+1} \quad x^2 \le 1$

15. $\arcsin h(x) = \sum_{k=0}^{\infty} (-1)^k \dfrac{(2k)!}{2^{2k}(k!)^2(2k+1)} x^{2k+1} \quad x^2 \le 1$ 16. $\arctan(x) = \sum_{k=0}^{\infty} (-1)^k \dfrac{x^{2k+1}}{(2k+1)} \quad x^2 \le 1$

17. $\arctan h(x) = \sum_{k=0}^{\infty} \dfrac{x^{2k+1}}{(2k+1)} \quad x^2 \le 1$

Bölüm 2 Fonksiyonların kökleri

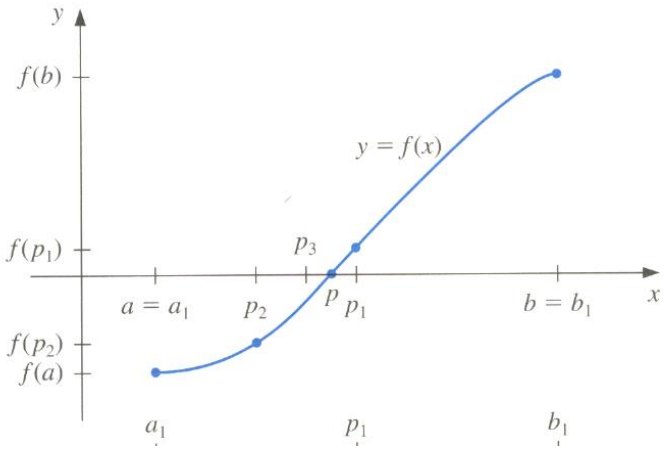
Kök bulma işlemi mühendislikte oldukça yoğun olarak kullanılan yöntemlerden biridir. Matematiksel olarak bir bilinmiyenli fonksiyonlar için f(x)= 0 denklemini sağlayan x değerinin bulunması olarak özetlenebilir.
Eğer denklem çok bilnmiyenli ise $f_i(x_1,x_2,x_3,\ldots,x_n)=0$, i=1..n denklemlerini sağlayan $x_1,x_2,x_3,\ldots,x_n$ değerlerini bulmak işlemidir. Bu bölümde kök bulma işlemlerinin bazılarını detaylı olarak inceleyeceğiz.

2.19 İkiye Bölme Yöntemi (Bisection)

İkiye bölme yönteminde f(x) fonksiyonunun a <= x <= b bölgesinde kökü aranır. Eğer bu bölgede bir kök mevcutsa
f(a)*f(b) < 0
olacaktır. Bu şart gerçekleşmiyor ise bu bölgede bir kök olmadığını varsayabiliriz. Eğer Bu bölgede bir kök mevcut ise bölgeyi iki eşit parçaya böleriz a-b bölgesinin orta noktası p=(a+b)/2 ise fonksiyonu p noktasında değerlendiririz. Eğer f(a)*f(p) < 0 ise kökün a-p aralığında olduğunu söyleyebiliriz. Eğer f(a)*f(p) > 0 ise kök p-b bölgesindedir. Tesadüfen f(a)*f(p)=0 oluşabilir, bu kök değerinin p ye tam olarak eşit olduğu anlamına gelir ama bu çok küçük bir olasılıktır. Bulduğumuz yeni kök bölgesini bir sonraki iteasyon için kullanırız. İteratif prosesi durdurmak için

$$\left|\frac{(b-a)}{(b+a)}\right| < \varepsilon$$

gibi bir durdurma kriterinden yararlanabiliriz.



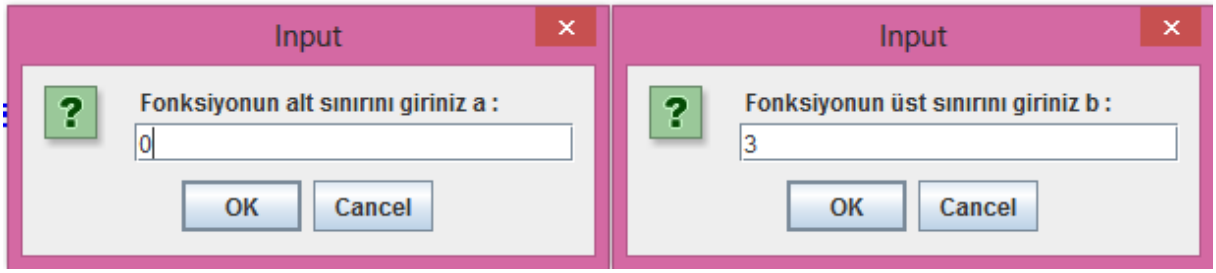
(bolum2_000.jpg,Resimaltı: **ikiye bölme yönteminin uygulanmasının grafik gösterimi**)

Altta ikiye bölme yöntemini hesaplayan SCO1 sınıfı verilmiştir. Döngüyü durdurmak için iteratif yöntemde durdurma kriterinin yanında maksimum iterasyon sayısı kriteri uygulanmıştır.

Program 2.1.1 İkiye bölme yöntemi

```
 ///Bilgisayar programlama II ve Sayısal Çözümleme
//ÖDEV 1
// f(x)=ln(x)-0.7 fonksiyonunun a=0.5 ile b=3 arasındaki
// kökünü bulmak için yarıya bölme yönteminikullanarak bir
// program geliştiriniz, kök değerini hesaplatınız

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============== Tanımı gereken fonksiyon ================

class fb extends f_x
{
  public double func (double x)
  { return x*x-2.0;}
//============================================================
}
public class FSCO1
{

public static double ikiyebolme(f_x f,double a,double b)
{
//ikiye bölme yöntemi a ile b arsında kök değerini
// her seferinde sayıyı ikiye bölerek arar.
double test;
double p=0;
double es,ea;
double fa,fp;
int maxit=500,iter=0;
es=0.0000001;
ea=1.1*es;
while((ea>es)&&(iter<maxit))
{
   p=(a+b)/2.0;
   iter++;
   if((a+b)!=0)
      { ea=Math.abs((b-a)/(a+b))*100;}
   fa= f.func(a);
   fp= f.func(p);
   test= fa*fp;
   if(test==0.0)      ea=0;
   else if(test<0.0)  b=p;
   else               {a=p;}
}
if(iter>=maxit) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return p;
}

    public static void main (String args[])
    {
            fb f=new fb();
```

```
    double a,b;
    a=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun alt sınırını giriniz a : "));
    b=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun üst sınırını giriniz b : "));
    double r;
    r= ikiyebolme(f,a,b);
    System.out.println(" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r));
    //JOptionPane.showMessageDialog(null," kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r),
    //"ikiye bölme yöntemiyle kök hesabı : ",JOptionPane.PLAIN_MESSAGE);
    System.exit(0);
    }
}
```



(bolum2_001.jpg,Resimaltı:)

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO1
 kök değeri  : 1.4142135614529252
Fonksiyon değeri : -2.6026334420947705E-9
> Terminated with exit code 0.
```

Aşağıdaki tabloda aynı ikiyebölme yöteminin excel formunda hesaplanmasını görmekteyiz.

Program 2.1.2 excel ortamında ikiyebölme programı

xl	xu	xr	fxl	fxu	fxr	fxr*fxl
1	2	1.5	-1	2	0.25	-0.25
1	1.5	1.25	-1	0.25	-0.4375	0.4375
1.25	1.5	1.375	-0.4375	0.25	-0.10938	0.047852
1.375	1.5	1.4375	-0.10938	0.25	0.066406	-0.00726
1.375	1.4375	1.40625	-0.10938	0.066406	-0.02246	0.002457
1.40625	1.4375	1.421875	-0.02246	0.066406	0.021729	-0.00049
1.40625	1.421875	1.4140625	-0.02246	0.021729	-0.00043	9.6E-06
1.4140625	1.421875	1.41796875	-0.00043	0.021729	0.010635	-4.5E-06
1.4140625	1.41796875	1.416015625	-0.00043	0.010635	0.0051	-2.2E-06
1.4140625	1.416015625	1.415039063	-0.00043	0.0051	0.002336	-1E-06
1.4140625	1.415039063	1.414550781	-0.00043	0.002336	0.000954	-4.1E-07
1.4140625	1.414550781	1.414306641	-0.00043	0.000954	0.000263	-1.1E-07
1.4140625	1.414306641	1.41418457	-0.00043	0.000263	-8.2E-05	3.5E-08
1.41418457	1.414306641	1.414245605	-8.2E-05	0.000263	9.06E-05	-7.4E-09

Aynı kodu Matlab(Octave) ortamında yazacak olursak :

Program 2.1.3 Matlab(Octave) ortamında ikiyebölme programı

```
function xr=bisection(f,xl,xu)
% ikiyebölme metodu (bisection)
maxit=100;
iter=0;
es=0.0000001;
```

```
ea=1.1*es;
while((ea>es)&&(iter<maxit))
   xr=(xl+xu)/2.0;
   iter=iter+1;
   if xr~=0 ea=abs((xu-xl)/(xu+xl))*100;end
   fxl= f(xl);
   fxr= f(xr);
   test= fxl*fxr;
   if test == 0.0      ea=0;
   elseif test < 0.0  xu=xr;
   else               xl=xr;
   end
end
if(iter>=maxit) fprintf('maksimum döngü sayısı aşıldı , sonuç geçerli olmayabilir');
end
end
```

Matlab editöründe bu fonksiyonu kullanırsak

```
>> f1=@(x) x*x-2;
>> y=bisection(f1,0,2)
y =1.414213561452925
```

Sonucunu alırız. Son olarak programımızın C++ dilindeki karşılığını da verelim. Başlangıç programlarında programların görece kısa olmasından da yararlanarak eşdeğer kodların bazılarını değişik dillerde vermeye çalışacağız, bunun temel gayesi diller arasında çok küçük farklar olduğunun farkındalığını oluşturmaktır. Yalnız göreceli karışık olan programlarda diğer dil örneklerine girilmeyecek, bu programlar sadece göreceli basit programlar için verilecektir.

Program 2.1.4 C++ ortamında ikiyebölme programı

```
#include <iostream>
#include <math.h>

double func(double x)
{ return x*x-2.0;}

double bisection(double xl,double xu)
{
//ikiye bölme metodu
double test;
double xr=0;
double es,ea;
double fxl,fxr;
int maxit=100;
int iter=0;
es=0.00000001;
ea=1.1*es;
while((ea>es)&&(iter<maxit))
{
   xr=(xl+xu)/2.0;
   iter++;
   if(xr!=0)
      { ea=fabs((xu-xl)/(xu+xl))*100.0;}
   fxl= func(xl);
   fxr= func(xr);
   test= fxl*fxr;
   if(test==0.0)      ea=0;
   else if(test<0.0)  xu=xr;
   else               {xl=xr;}
}
if(iter>=maxit) std::cout <<("Maksimum döngü sayısı aşılı sonuç hatalı olabilir \n");
return xr;
}

int main()
{  double a,b,r;
  std::cout << " alt siniri giriniz a : ";
  std::cin >>a;
  std::cout << " ust siniri giriniz b : ";
  std::cin >>b;
```

```
   r=bisection(a,b);
   std::cout<<"r="<<r;
   return 0;
}
```

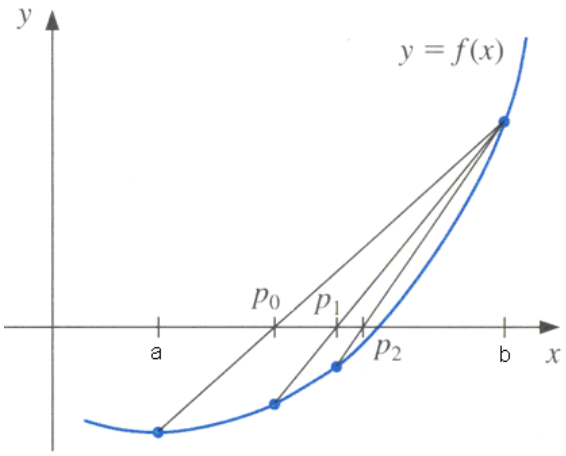
```
C:\Users\asus-pc\Desktop\Say□sal €"z□mleme - M. Turhan €oban\SCO1>bisection
bisection
 alt siniri giriniz a : 0
 ust siniri giriniz b : 2
r=1.41421
C:\Users\asus-pc\Desktop\Say□sal €"z□mleme - M. Turhan €oban\SCO1>
```

2.20 Yer değiştirme yöntemi (false position)

İkiye bölme yönteminde yaklaşma göreceli olarak yavaş ilerler. Yaklaşma hızını arttırmanın bir yolu a ve b noktalarını bir doğru ile birleştirmek ve doğrunun x eksenini kesim noktasını bir sonraki kök olarak kullanmaktır. p noktasını

$$p = b - f(b)\frac{(a-b)}{(f(a)-f(b)}$$

formülüyle bulabiliriz. Kök testi için ikiye bölme yönteminde kullandığımız aynı yöntemi kullanırız.

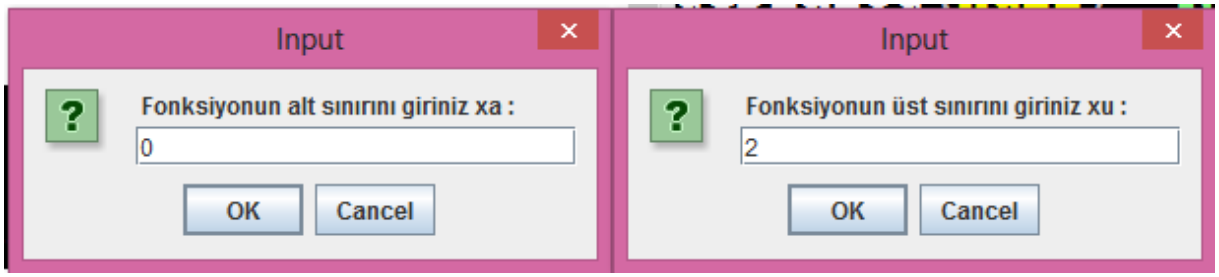


(bolum2_003.jpg,Resimaltı: yer değiştirme yöntemi)

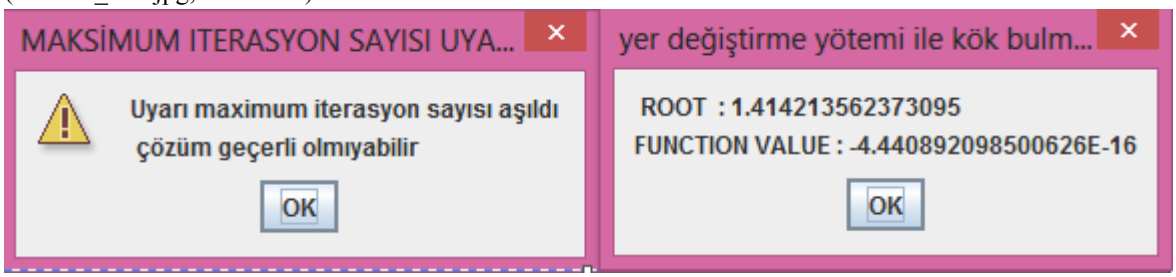
Program 2.2.1 Yer değiştirme yöntemi ile kök bulma

```
 import javax.swing.*;

//============= Function to be define ================

class fb extends f_x
{
   public double func (double x)
   { return x*x-2.0;}
}
//============================================================
public class NA2
{
public static double yerdegistirme(f_x f,double xl,double xu)
{
//false position root finding method
double test;
double xr=0;
double es,ea;
double fxl,fxu,fxr;
int maxit=100,iter=0;
es=0.001;
ea=1.1*es;
fxl= f.func(xl);
fxu= f.func(xu);
while((ea>es)&&(iter<maxit))
{
                xr=xu-fxu*(xl-xu)/(fxl-fxu);
                fxr= f.func(xr);
```

```
    iter++;
    if((xl+xu)!=0)
       { ea=Math.abs((xu-xl)/(xu+xl))*100;}
    test= fxl*fxr;
    if(test==0.0)      ea=0;
    else if(test<0.0)  {xu=xr;fxu=fxr;}
    else               {xl=xr;fxl=fxr;}
}
if(iter>=maxit) {JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n çözüm geçerli
olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE); }
return xr;
}

    public static void main (String args[])
    {
    double a,b;
    a=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun alt sınırını giriniz xa : "));
    b=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun üst sınırını giriniz xu : "));
    double r;
    fb f=new fb();
    r= yerdegistirme(f,a,b);
    JOptionPane.showMessageDialog(null," ROOT  : "+r+"\nFUNCTION VALUE : "+f.func(r),
    "yer değiştirme yötemi ile kök bulma : ",JOptionPane.PLAIN_MESSAGE);
    System.exit(0);
    }
}
```



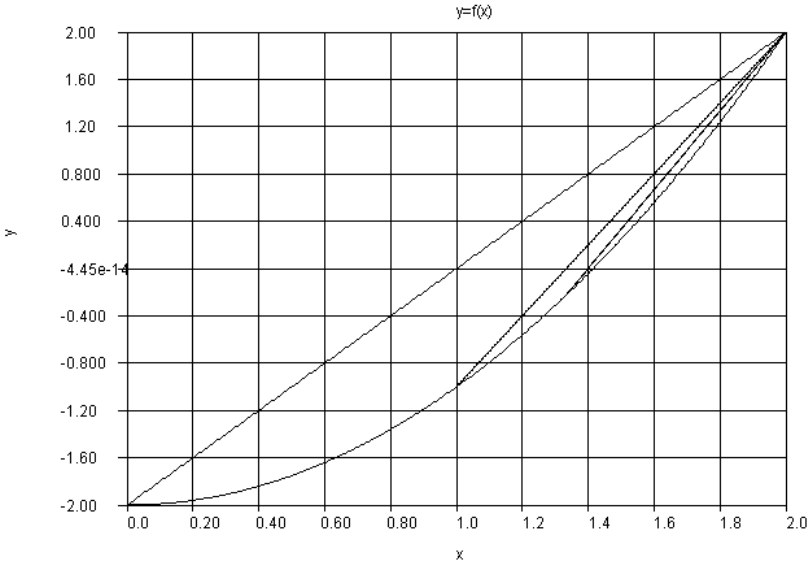
(bolum2_004.jpg,Resimaltı:)



(bolum2_005.jpg,Resimaltı:)

Burada birşeylerin yanlış gittiği görülüyor. Çok basit bir fonksiyon için sistemimiz uyarı mesajı verecek kadar çok iterasyon yaptı, u durumda sonuç doğru olarak bulundu, ama tabi ki bundan emin olamayız. Yanlış ve hatalı giden

neydi? Temel olarak bir grafik gösterimde fonksiyon değerlerinin nasıl değiştiğine bakacak olursak, hep



(bolum2_006.jpg,Resimaltı:f(x)=x^2-2 fonksiyonunun kökünün yer değiştirme yöntemiyle elde edilen ilk 3 dönüşümü)

tek yönlü bir değişim olduğunu gözlüyoruz. Bu yüzden de döngünün dönüşüm hızı yavaş olmakta. Yapılması mümkün olan basit bir değişimle bunu önleyebiliriz. Örneğin ikiye bölme yöntemiyle yer değiştirme yöntemi birlikte kullanılabiliriz. Bu durumda değiştirilmiş yer değiştirme yöntemini elde ederiz. İlk programımızda java dilindeki örnek kodumuz yer almaktadır. Aynı kodun Matlab ve c++ versiyonları da verilmiştir.

Program 2.2.2 Dönüştürülmüş yer değiştirme yöntemi ile kök bulma

```
 import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============== Tanımı gereken fonksiyon =================

class fb extends f_x
{
  public double func (double x)
  { return x*x-2;}
}
//============================================================
public class FSCO3
{

public static double yerdegistirme(f_x f,double xa,double xu)
{
//Düzeltmeli yer değiştirme (Modified False position) yöntemi ile kök bulma
// xa ve xu bölgesinde kök bulur
double test;
double p=0;
double es,ea;
double fa,fp,fu;
int maxit=100,iter=0;
es=0.000001;
ea=1.1*es;
double xold;
int ia=0,iu=0;
```

```
fa=f.func(xa);
fu=f.func(xu);
p=xu;
double xpold;
while(iter<maxit && ea>es)
{
xpold=p;
p=xu-fu*(xa-xu)/(fa-fu);
fp=f.func(p);
iter++;
if(p!=0) ea=Math.abs((p-xpold)/p)*100.0;
test=fa*fp;
if(test<0) {xu=p;fu=f.func(xu);iu=0;ia++;if(ia>=2) fa/=2.0;}
else if(test>0) {xa=p;fa=f.func(xa);ia=0;iu+=1;if(iu>=2)fu/=2.0;}
else ea=0;
}
if(iter>=maxit) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return p;
}

    public static void main (String args[])
    {
            fb f=new fb();
    double xa,xu;
    xa=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun alt sınırını giriniz xa : "));
    xu=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun üst sınırını giriniz xu : "));
    double r= yerdegistirme(f,xa,xu);
    System.out.println(" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r));
    //JOptionPane.showMessageDialog(null," kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r),
    //"yer değiştirme : ",JOptionPane.PLAIN_MESSAGE);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO3
 kök değeri  : 1.4142135623730951
Fonksiyon değeri : 4.440892098500626E-16
> Terminated with exit code 0.
```

Program 2.2.3 Dönüştürülmüş yer değiştirme yöntemi ile kök bulma, Matlab (Ocatave) ortamı

```
function p=yerdegistirme(f,xa,xu)
% dönüştürülmüş yer değiştirme (Illinois) metodu
maxit=100;
iter=0;
es=0.000001;
ea=1.1*es;
ia=0;
iu=0;
fa=f(xa);
fu=f(xu);
p=xu;
double xpold;
while(iter<maxit && ea>es)
xpold=p;
p=xu-fu*(xa-xu)/(fa-fu);
fp=f(p);
iter=iter+1;
if p~=0 ea=abs((p-xpold)/p)*100.0;
test=fa*fp;
if test<0
   xu=p;
```

```
    fu=f(xu);
    iu=0;ia=ia+1;
    if ia >= 2 fa=fa/2.0;end
elseif test>0  xa=p;fa=f(xa);ia=0;iu=iu+1;
       if iu>=2 fu=fu/2.0; end
elseif test==0 ea = 0; end
end
if iter>=maxit fprintf('Maksimum döngü sayısı aşıldı, sonuç geçerli olmayabilir'); end
end
```

```
>> f1=@(x) x*x-2;
>> yerdegistirme(f1,0,2)
ans =
   1.414213562373095
>>
```

Program 2.243 Dönüştürülmüş yer değiştirme yöntemi ile kök bulma, c++ ortamı

```
#include <iostream>
#include <math.h>

double func(double x)
{ return x*x-2.0;}

double yerdegistirme(double xa,double xu)
{
//Modified False position root finding
double test;
double p=0;
double es,ea;
double fa,fp,fu;
int maxit=100,iter=0;
es=0.000001;
ea=1.1*es;
int ia=0,iu=0;
fa=func(xa);
fu=func(xu);
p=xu;
double xpold;
while(iter<maxit && ea>es)
{
xpold=p;
p=xu-fu*(xa-xu)/(fa-fu);
fp=func(p);
iter++;
if(p!=0) ea=fabs((p-xpold)/p)*100.0;
test=fa*fp;
if(test<0) {xu=p;fu=func(xu);iu=0;ia++;if(ia>=2) fa/=2.0;}
else if(test>0) {xa=p;fa=func(xa);ia=0;iu+=1;if(iu>=2)fu/=2.0;}
else ea=0;
}
if(iter>=maxit) std::cout <<("Maximum number of iteration is exceeded \n result might not be valid");
return p;
}

int main()
{  double a,b,r;
std::cout << " alt siniri giriniz a : ";
   std::cin >>a;
   std::cout << " ust siniri giriniz b : ";
   std::cin >>b;
   r=yerdegistirme(a,b);
   std::cout<<"r="<<r;
   return 0;
}
```

```
C:\Users\asus-pc\Desktop\Sayisal çözümleme - M. Turhan Çoban\SCO1>yerdegistirme
yerdegistirme
 alt siniri giriniz a : 0
 ust siniri giriniz b : 2
r=1.41421
C:\Users\asus-pc\Desktop\Say□sal €”z□mleme - M. Turhan €oban\SCO1>
```

2.21 Müller Yöntemi

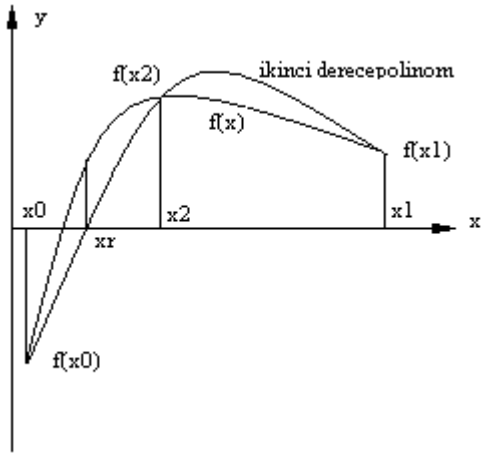
Yer değiştirme yönteminde iki noktadan bir doğru geçmesi prensibinden yararlanarak yeni kök değeri tahmini geliştirmiştik. Aynı kavramı ikinci dereceden polinom kullanarakta yapabiliriz, ancak ikinci dereceden polinom tanımlamak için üç noktaya ihtiyacımız olacaktır. x_0, x_1, x_2 noktalarının giriş değerleri olarak verildiğini varsayalım (x_3 değeri x_0 ve x_1 değeri arasında olsun). Fonksiyonumuz f(x) olsun. Bu üç noktanın y değerleri olarak $f(x_0)$, $f(x_1)$ ve $f(x_2)$ değerlerini alalım bu üç noktadan ikinci derece bir polinom geçirelim. Bu polinomun kökünün lineer döngüye göre gerçek köke daha fala yaklaşacağını varsayabiliriz.

$f(x_0)-f(x_2)=a(x_0-x_2)^2+b(x_0-x_2)$
$f(x_1)-f(x_2)=a(x_1-x_2)^2+b(x_1-x_2)$

$h_0= x_1 - x_0$
$h_1= x_2 - x_1$

$$d_0 = \frac{f(x_1) - f(x_0)}{h_0}$$

$$d_1 = \frac{f(x_2) - f(x_1)}{h_1}$$



(bolum2_07.jpg,Resimaltı: Müller yönteminin grafik olarak gösterilmesi)

bu ifadeler ana denklemlerde yerine konulursa
$(h_0+h_1)b - (h_0+h_1)^2a = h_0d_0 + h_1d_1$
$h_1 b - h_1^2 a = h_1d_1$
elde edilir buradan da a ve b çözülebilir.

$$a = \frac{d_1 - d_0}{h_1 + h_0}$$

$b = a*h_1 + d_1$

$c=f(x_2)$
kökü bulmak için :

$$\Delta = \sqrt{b^2 - 4ac}$$

eğer

$$|b + \Delta| > |b - \Delta| \quad e = b + \Delta$$

$$|b+\Delta|\ |b-\Delta|\ e = b-\Delta$$

$$x_r = x_2 - \frac{2c}{e}$$

x_2 değerinin yerine x_r değerini koyarak yeni iterasyonumuza başlayabiliriz. Program 2.3.1 de müller yöntemi gerçek köklerin değerini bulma yöntemi olarak verilmiştir. Bu yöntem kompleks köklerin bulunması için de rahatlıkla kullanılabilir.

Program 2.3.1 Müller yöntemi ile kök bulma programı

```
 ///Bilgisayar programlama II ve Sayısal Çözümleme
// Muller yöntemi ile kök bulma
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============== Tanımı gereken fonksiyon ================

class fb extends f_x
{
  public double func (double x)
  { return x*x-2;}
}
//============================================================


public class FSCO2
{
public static double muller(f_x f,double x0,double x1,double x2)
{
//Müller yöntemi ile kök bulma (gerçek kökler)
//orta xr noktasından +/- h kadar giderek diğer 2 noktaya ulaşıyoruz.
int maxit=5;
//double x2=xr;
//double x1=xr*(1+h);
//double x0=xr*(1-h);
double xr=x0;
int iter=0;
double es1=0.001;
double es=es1;
double ea=1.1*es;
double h0,h1;
double d0,d1;
double a,b,c;
double fx0,fx1,fx2;
double determinant;
double den;
double dxr;
while(iter<maxit && ea>es)
{
   fx0=f.func(x0);
   fx1=f.func(x1);
   fx2=f.func(x2);
   h0=x1-x0;
   h1=x2-x1;
   d0=(fx1-fx0)/h0;
   d1=(fx2-fx1)/h1;
```

```
    a=(d1-d0)/(h1+h0);
    b=a*h1+d1;
    c=fx2;
    determinant=b*b-4.0*a*c;
    if(determinant<=0) determinant=0;//we do not want to calculate complex roots here
    else determinant=Math.sqrt(determinant);
    if(Math.abs(b+determinant)>Math.abs(b-determinant)) {den=b+determinant;;}
    else {den=b-determinant;}
    dxr=-2*c/den;
    xr=x2+dxr;
    System.out.println("a="+a+"b="+b+"c="+c+"disc="+determinant+"xr="+xr+"iter="+iter);
    ea=Math.abs((xr-x2)/xr)*100;
    iter++;
    x0=x1;
    x1=x2;
    x2=xr;
}
if(iter>=maxit) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return xr;
}

    public static void main (String args[])
    {
            fb f=new fb();
    double x0,x1,x2;
    x0=Double.parseDouble(JOptionPane.showInputDialog(" kök tahmini x0 : "));
    x1=Double.parseDouble(JOptionPane.showInputDialog(" kök tahmini x1 : "));
    x2=Double.parseDouble(JOptionPane.showInputDialog(" kök tahmini x2 : "));
    //h=Double.parseDouble(JOptionPane.showInputDialog(" h=(küçük bir rakam örneğin 0.1"));
    double r;
    r= muller(f,x0,x1,x2);
    System.out.println(" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r));
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO3
 kök değeri  : 1.4142135623730951
Fonksiyon değeri : 4.440892098500626E-16
> Terminated with exit code 0.
```

Program 2.3.2 Müller yöntemi ile kök bulma programı (Matlab-Octave versiyonu)

```
function xr=muller(f,x0,x1,x2)
% Muller kök bulma metodu
% ikinci derce polinom eğri uydurmaya dayanır
iter=0;
es1=0.001;
es=es1;
ea=1.1*es;
maxit = 100;
xr=x0;
while (iter<maxit) && (ea>es)
   fx0=f(x0);
   fx1=f(x1);
   fx2=f(x2);
   h0=x1-x0;
   h1=x2-x1;
   d0=(fx1-fx0)/h0;
   d1=(fx2-fx1)/h1;
   a=(d1-d0)/(h1+h0);
   b=a*h1+d1;
```

```
    c=fx2;
    determinant=b*b-4.0*a*c;
    if determinant<=0  determinant=0;
    else determinant=sqrt(determinant);end
    if abs(b+determinant)>  abs(b-determinant) den=b+determinant;
    else den=b-determinant;end
    dxr=-2*c/den;
    xr=x2+dxr;
    ea=abs((xr-x2)/xr)*100;
    iter=iter+1;
    x0=x1;
    x1=x2;
    x2=xr;
end
if iter>=maxit fprintf('Uyarı:Maksimum döngü sayısı aşıldı, sonuç geçerli olmayabilir');
end
```

```
>> f1=@(x) x*x-2;
>> muller(f1,0,1,2)
ans =
   1.414213562373095
>>
```

Program 2.3.3 Müller yöntemi ile kök bulma programı (C++ versiyonu)

```
#include <iostream>
#include <math.h>

double func(double x)
{ return x*x-2.0;}

double muller(double xr,double h)
{
//Müller yöntemi ile kök bulma (gerçek kökler)
//orta xr noktasından +/- h kadar giderek diğer 2 noktaya ulaşıyoruz.
int maxit=50;
double x2=xr;
double x1=xr*(1+h);
double x0=xr*(1-h);
int iter=0;
double es1=0.001;
double es=es1;
double ea=1.1*es;
double h0,h1;
double d0,d1;
double a,b,c;
double fx0,fx1,fx2;
double determinant;
double den;
double dxr;
while(iter<maxit && ea>es)
{
   fx0=func(x0);
   fx1=func(x1);
   fx2=func(x2);
   h0=x1-x0;
   h1=x2-x1;
   d0=(fx1-fx0)/h0;
   d1=(fx2-fx1)/h1;
   a=(d1-d0)/(h1+h0);
   b=a*h1+d1;
   c=fx2;
   determinant=b*b-4.0*a*c;
   if(determinant<=0) determinant=0;//we do not want to calculate complex roots here
   else determinant=sqrt(determinant);
   if(fabs(b+determinant)>fabs(b-determinant)) {den=b+determinant;;}
   else {den=b-determinant;}
   dxr=-2*c/den;
   xr=x2+dxr;
   ea=fabs((xr-x2)/xr)*100;
   iter++;
   x0=x1;
   x1=x2;
   x2=xr;
```

```
}
if(iter>=maxit) std::cout <<("Maksimum döngü sayısı aşılı sonuç hatalı olabilir \n");
return xr;
}

int main()
{  double xr,h,r;
   std::cout << "Muller yontemi kok tahmin degerini giriniz\n x0 : ";
   std::cin >>xr;
   std::cout << "diğer iki kökün ilk kökten mesafesini giriniz  h : ";
   std::cin >>h;
   r=muller(xr,h);
   std::cout<<"r="<<r;
   return 0;
}
```

```
C:\co>muller
muller
Muller yontemi kok tahmin degerini giriniz
 x0 : 1
diğer iki kökün ilk kökten mesafesini giriniz  h : 1
r=1.41421
C:\co>
```

2.22Newton-Raphson Yöntemi

Taylor formülü

$$f(x_{n+1}) = f(x_n) + f'(x_n)(x_{n+1} - x_n) + f''(x_n)\frac{(x_{n+1} - x_n)^2}{2!} + f^{(3)}(x_n)\frac{(x_{n+1} - x_n)^3}{3!} +$$

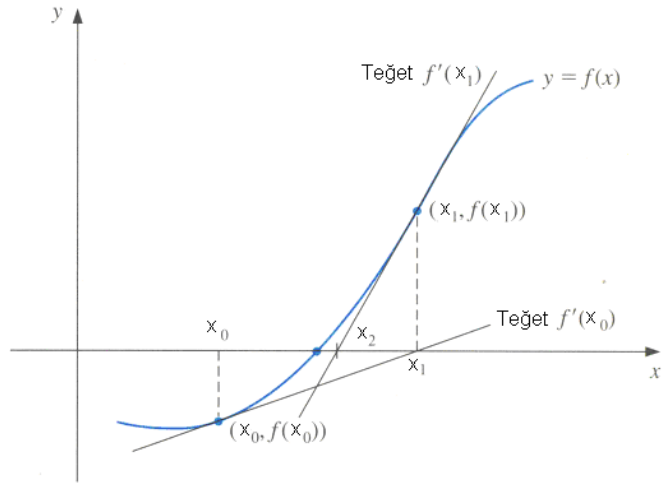
$$f^{(4)}(x_n)\frac{(x_{n+1} - x_n)^4}{4!} + f^{(5)}(x_n)\frac{(x_{n+1} - x_n)^5}{5!} + ...$$

şeklindedir. $f(x_{n+1})=0$ olmasını istediğimizden bu değeri denkleme koyar ve birinci dereceden yüksek terimlerin 0 a gittiğini kabul eder isek

$0 = f(x_n) + f'(x_n)(x_{n+1} - x_n)$ olur bu denklemden x_{n+1} çekilirse

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

denklemi elde edilir Bu denkleme Newton-Raphson kök bulma denklemi ismi verilir. x_0 ilk tahmininden (tek değer) başlayarak kök değerine iteratif olarak ulaşabiliriz. Newton formülü ilk tahmin gerçek köke göreceli olarak yakınsadığında çok hızlı yaklaşım verir, ancak ilk tahmin kökten uzak ise köke ulaşması uzun sürebilir veya hiç ulaşamayabilir. Newton raphson denkleminin tek zorluğu türevleri bilme zorunluluğudur.



(bolum2_08.jpg,Resimaltı: Newton yönteminin grafik olarak gösterilmesi)

Program 2.4.1 Newton-Raphson kök bulma programı

```
 //
//
// Newton-Raphson yöntemi ile kök bulmak
//

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============== Tanımı gereken fonksiyon ================

class fb extends f_x
{
   public double func (double x)
   { return x*x-2;}
//============================================================
}

class dfb extends f_x
{
   public double func (double x)
   { return 2.0*x;}
}
//============================================================
public class FSCO4A
{
public static double newton(f_x f,f_x df,double x)
{
int nmax=100;
double tolerance=1.0e-10;
double fx,dfx;
for(int i=0;i<nmax;i++)
{
  fx=f.func(x);
  dfx=df.func(x);
  x-=fx/dfx;
  System.out.println("i="+i+"x="+x+"fx="+fx+"dfx = "+dfx);
  if(Math.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
    public static void main (String args[])
    {
            fb f=new fb();
            dfb df=new dfb();
    double x0;
    x0=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyon değeri için ilk tahmin : "));
    double r= newton(f,df,x0);
    System.out.println(" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r));
    System.exit(0);
    }
}
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO4A
i=0x=1.5fx=-1.0dfx = 2.0
i=1x=1.4166666666666667fx=0.25dfx = 3.0
i=2x=1.4142156862745099fx=0.006944444444444642dfx = 2.8333333333333335
i=3x=1.4142135623746899fx=6.007304882871267E-6dfx = 2.8284313725490198
i=4x=1.4142135623730951fx=4.510614104447086E-12dfx = 2.8284271247493797
 kök değeri  : 1.4142135623730951
Fonksiyon değeri : 4.440892098500626E-16
> Terminated with exit code 0.
```

Program 2.4.2 Newton-Raphson kök bulma programı (Matlab-Octav versiyonu)

```
function p=newton_raphson(f,df,x,tolerance,nmax)
% secant method for root finding
if nargin<5 nmax=100;end
if nargin<4 tolerance=1e-10;end
i=0;
fx=f(x);
while (abs(fx) > tolerance) && i<nmax
   fx=f(x);
   dfx=df(x);
   x=x-fx/dfx;
   i=i+1;
end
p=x;
if i>=nmax fprintf('Maksimum döngü sayısı aşıldı sonuç geçersiz olabilir');end
end
```

```
>> f1=@(x) x*x-2;
>> df1=@(x)2*x;
>> newton_raphson(f1,df1,1)
ans =
   1.414213562373095
>>
```

Program 2.4.3 Newton-Raphson kök bulma programı (C++ versiyonu)

```
 #include <iostream>
#include <math.h>

double func(double x)
{               double y=x*x-2.0;
                return y;
}

double dfunc(double x)
{ double y=2.0*x;
                return y;
}

double newton(double x)
{
int nmax=100;
double tolerance=1.0e-8;
double fx,dfx;
for(int i=0;i<nmax;i++)
{
  fx=func(x);
  dfx=dfunc(x);
  x-=fx/dfx;
  std::cout <<"i="<<i<<"x="<<x<<"fx="<<fx<<"dfx="<<dfx<<"\n";
  if(fabs(fx)<tolerance) { return x;}
}
std::cout <<("Maksimum döngü sayısı aşılı sonuç hatalı olabilir \n");return x;
return x;
}

int main()
{  double xa,r;
   std::cout << " kök ilk tahmin değerini giriniz : ";
   std::cin >>xa;

   r=newton(xa);
```

```
  std::cout<<"kök değeri : ="<<r;
  return 0;
}
```

```
C:\Users\asus-pc\Desktop\Say□sal €"z□mleme - M. Turhan €oban\SCO1>newton
newton
 kök ilk tahmin değerini giriniz : 1
i=0x=1.5fx=-1dfx=2
i=1x=1.41667fx=0.25dfx=3
i=2x=1.41422fx=0.00694444dfx=2.83333
i=3x=1.41421fx=6.0073e-006dfx=2.82843
i=4x=1.41421fx=4.51061e-012dfx=2.82843
kök değeri : =1.41421
C:\Users\asus-pc\Desktop\Say□sal €"z□mleme - M. Turhan €oban\SCO1>
```

Program 4.4-4 Newton-Raphson kök bulma programı (excell versiyonu)

Newton-Raphson kök bulma metodu $f(x)=x^2$-a fonksiyonu

a	2	
x	f(x)=x*x-a	fx/dx=2*x
1.00000000000000000	-1	2
1.50000000000000000	0.25	3
1.41666666666667000	0.006944444	2.833333333
1.41421568627451000	6.0073E-06	2.828431373
1.41421356237469000	4.51061E-12	2.828427125
1.41421356237310000	0	2.828427125
1.41421356237310000	0	2.828427125
1.41421356237310000	0	2.828427125

Newton- Raphson yöntemi aynı zamanda kompleks köklerin bulunmasında da kullanılabilir. Kompleks Newton-raphson metodu program 4.4-2 de verilmiştir. Bu programda kullanılan kompleks değişken hesabını yapan complex sınıfı da program 4.4-3 de verilmiştir.

Program 2.4.4 Newton-Raphson metoduyla kompleks kök bulma programı

```
 //
//
// kompleks Newton-Raphson yöntemi ile kök bulmak
//
//

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

abstract class cf_cx
{
 abstract complex func(complex x);
 complex dfunc(complex x)
 { // complex sayısal türev fonksiyonu
  double hh=0.001;
  complex h=new complex(hh,hh);
  complex he=new complex(-hh,-hh);
  complex h2=new complex(2.0*hh,2.0*hh);
  complex he2=new complex(-2.0*hh,-2.0*hh);
  complex xah=complex.add(x,h);
  complex xa2h=complex.add(x,h2);
  complex xeh=complex.add(x,he);
  complex xe2h=complex.add(x,he2);
  complex fe2h=complex.multiply(func(xe2h),(1.0/12.0));
  complex feh=complex.multiply(func(xeh),(-8.0/12.0));
  complex fah=complex.multiply(func(xah),(8.0/12.0));
  complex fa2h=complex.multiply(func(xa2h),(-1.0/12.0));
  complex df=complex.add(fe2h,complex.add(feh,complex.add(fah,fa2h)));
  df=complex.divide(df,h);
  return df;
 }
```

```
}

class fa extends cf_cx
{  //tek fonksiyon, tek bağımsız değişken
  // örnek  f=x*x+1
complex func(complex x)
{ complex f=complex.add(complex.multiply(x,x),1.0);
  return f;
}
}


public class SCO4G
{
public static complex newton(cf_cx f,complex x)
{
int nmax=50;
double tolerance=1.0e-8;
complex fx,dfx;
for(int i=0;i<nmax;i++)
{
  fx=f.func(x);
  dfx=f.dfunc(x);
  //x=complex.add(x,complex.multiply(complex.divide(fx,dfx),-1.0));
  x=complex.substract(x,complex.divide(fx,dfx));
  if(complex.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
    public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }

    public static void main (String args[])
    {
                fa f=new fa();
    double x0[]=verigir(" Fonksiyon değeri için ilk tahmin (iki değer giriniz) z=(a+bi) : ");
    complex x=new complex(x0[0],x0[1]);
    complex r= newton(f,x);
    complex ff=f.func(r);
    System.out.println("Newton-Raphson yöntemiyle kök hesabı : \n kök değeri  : "+r.toString()+"\nFonksiyon değeri : "+ff.toString());
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------> "C:\java\bin\javaw.exe" SCO4G
Newton-Raphson yöntemiyle kök hesabı :
 kök değeri  : (   0.0000000000 + i )
Fonksiyon değeri : (   0.0000000000 +    0.0000000000i )

> Terminated with exit code 0.
```

Kompleks kök bulma prosesinde gerekli olan kompleks değişken yapısını kendi geliştirdiğimiz bir java sınıfı olan complex sınıfı üzerinden oluşturduk. Bu sınıfın listesi aşağıda verilmiştir. Bu sınıf genel kullanım için hazırlandığından kod oldukça detaylı olarak hazırlanmıştır.

Program 2.4.5 kompleks Newton-Raphson programında kullanılan complex sınıfı

```
import java.io.*;
import java.text.*;
import java.util.Locale;
import java.util.*;

import javax.swing.*; // giriş çıkış

// class complex
//   complex number abstraction
//
class complex {
   // constructors
public complex()
{
nreal=0;
nimag=0;
}

public complex(double nre,double nim)
{
nreal=nre;
nimag=nim;
}

public complex(int nre,int nim)
{
nreal=(double)nre;
nimag=(double)nim;
}

public complex(double numer)
{
nreal=numer;
nimag=0;
}

public complex(complex value )
{
nreal=value.real();
nimag=value.imaginary();
}

// accessor functions

public double real()
{
//return real part of the complex number
return nreal;
}

public double imaginary()
{
//return imaginary part of the complex number
return nimag;
}

public double R()
{
```

```
// Return radius of polar coordinate equivalent of complex number
return Math.sqrt(nreal*nreal+nimag*nimag);
}

public double theta()
{
// Return angle of polar coordinate equivalent of complex number in radian
return Math.atan2(nimag,nreal);
}

public double dtheta()
{
// Return angle of polar coordinate equivalent of complex number in degree
return Math.atan2(nimag,nreal)*180.0/Math.PI;
}

    // assignments
public void assign(complex right)
{
nreal=right.real();
nimag=right.imaginary();
}

public void add(complex  right)
{
nimag = nimag + right.imaginary();
nreal = nreal + right.real();
}

public void substract(complex right)
{
nimag = nimag - right.imaginary();
nreal = nreal - right.real();
}

public void multiply(complex right )
{
nreal = nreal*right.real() - nimag*right.imaginary();
nimag = nreal*right.imaginary() +  nimag*right.real();
}

public void divide(complex right )
{
double a=nreal*nreal+nimag*nimag;
nreal = ( nreal*right.real() +  nimag*right.imaginary())/a;
nimag = (-nreal*right.imaginary() +  nimag*right.real())/a;
}

public static complex add(complex left, complex right)
{ // return sum of two complex numbers
 double r1=(left.real() + right.real());
 double i1=(left.imaginary() + right.imaginary());
 complex result;
 result=new complex(r1,i1);
 return result;
}
public static complex add(complex left, double right)
{ // return sum of two complex numbers
 double r1=(left.real() + right);
 double i1=left.imaginary();
 complex result;
 result=new complex(r1,i1);
 return result;
}

public static complex conjugate(complex z)
```

```
{complex z1=new complex(z.real(),-z.imaginary());
return z1;
}

public static complex substract(complex left, complex right)
{ // return substraction of two complex numbers
 complex result;
 result=new complex((left.real() - right.real()),
            (left.imaginary() - right.imaginary()));
 return result;
}

public static complex multiply(complex left, complex right)
{ // return multiplication of two complex numbers
 complex result;
 result=new complex
 ((left.real()*right.real() - left.imaginary()*right.imaginary()),
  (left.real()*right.imaginary() + left.imaginary()*right.real()));
return result;
}

public static complex multiply(complex left, double  right)
{complex result;
 result=new complex((left.real()*right),(left.imaginary()*right));
 return result;
}

public static complex divide(complex left, complex  right)
{ // return division of two complex numbers
 double a=right.real()*right.real()+right.imaginary()*right.imaginary();
 complex result;
 result=new complex
 ((left.real()*right.real()  + left.imaginary()*right.imaginary())/a,
  (-left.real()*right.imaginary() + left.imaginary()*right.real())/a);
 return result;
}
public static complex divide(complex left, double  right)
{ // return division of a complex number to a real number
 complex result;
 result=new complex((left.real()/right),(left.imaginary()/right));
 return result;
}

public static complex pow(complex left, double right)
{ // return sum of two complex numbers
 double Rad,th;
 Rad=Math.pow(left.R(),right);
 th=right*left.theta();
 complex result;
 result =new complex((Rad*Math.cos(th) ),
            (Rad*Math.sin(th) )  );
 return result;
}
public static complex exp(complex left)
{ // exp(x+i*y)
 complex result;
 result =new complex((Math.exp(left.real())*Math.cos(left.imaginary()) ),
                (Math.exp(left.real())*Math.sin(left.imaginary()) ));
 return result;
}

public static complex exp(double left)
{ // exp(i*left) imaginary exponent
 complex result;
 result =new complex(Math.cos(left),
                Math.sin(left) );
```

```
 return result;
}

public static complex sqrt(complex left)
{
return pow(left,0.5);
}

public static double abs(complex left)
{
return left.R();
}

public boolean smaller(complex left,complex right)
{
// less then comparison of two complex numbers
return (left.R() < right.R());
}

public boolean smaller_equal(complex left,complex right)
{
// less then and equal comparison of two complex numbers
return (left.R() <= right.R());
}

public boolean greater(complex left,complex right)
{
// greater then comparison of two complex numbers
return left.R() > right.R();
}

public boolean greater_equal(complex left,complex  right)
{
// greater then and equal comparison of two complex numbers
return left.R() >= right.R();
}


public boolean equal(complex left,complex right)
{
// equal comparison of two complex numbers
return left.R() == right.R();
}

public boolean not_equal(complex left,complex  right)
{
// not equal comparison of two complex numbers
return left.R() != right.R();
}

public static String toString(double left, int w, int d)
// converts a double to a string with given width and decimals.
{
                NumberFormat df=NumberFormat.getInstance(Locale.US);
                df.setMaximumFractionDigits(d);
   df.setMinimumFractionDigits(d);
   df.setGroupingUsed(false);
   String s = df.format(left);
   while (s.length() < w)
     s = " " + s;
   if (s.length() > w)
   {
      s = "";
      for (int i=0; i<w; i++)
        s = s + "-";
   }
```

```
    return s;
}

public static String toString(double left)
{// converts a double to a string with a constant width and constant decimals.
  return toString(left,15,10);}

public static String toString(complex  value)
{
String b="";
if(Math.abs(value.imaginary())!=1)
  {
  if(value.imaginary() >= 0)
    b=b+"("+toString(value.real())+" + "+toString(value.imaginary())+"i )";
  else
    b=b+"("+toString(value.real())+" - "+toString(-value.imaginary())+"i )";
  }
else
  {
  if(value.imaginary() >= 0)
    b=b+"("+toString(value.real())+" + i )";
  else
    b=b+"("+toString(value.real())+" - i )";
}
return b;
}

public String toString()
{
String b="";
if(Math.abs(imaginary())!=1)
  {
  if(imaginary() >= 0)
    b=b+"("+toString(real())+" + "+toString(imaginary())+"i )";
  else
    b=b+"("+toString(real())+" - "+toString(-imaginary())+"i )";
  }
else
  {
  if(imaginary() >= 0)
    b=b+"("+toString(real())+" + i )";
  else
    b=b+"("+toString(real())+" - i )";
}
return b;
}
    public static complex toComplex(String s)
     {
                //bu metod compleks sayıyı ekrandan okur.
                //StringTokanizer kütüphane sınıfı bir stringi cümlelere ayırır
     String s1=JOptionPane.showInputDialog(s);
     StringTokenizer token=new StringTokenizer(s1);
     int n=token.countTokens()-1;
     int m=n+1;
     double a[]=new double[m];
     int j=0;
     while(token.hasMoreTokens())
     {
     Double ax=new Double(token.nextToken());
     a[j++]=ax.doubleValue();
     }
     complex b=new complex(a[0],a[1]);
     return b;
     }
    // data areas
     public double   nreal;
```

```
    public double   nimag;
        };
```

2.23Kiriş Yöntemi

Newton yönteminin bir problemi türev fonksiyonu kulanma gereğidir. Bu türev fonksiyonunu fark denklemiyle değiştirirsek (türevdeki teğet yerine teğete yakın bir kiriş kullanırsak) kiriş yöntemi oluşur

Eğer fark denklemi olarak $f'(x_n) \cong \frac{f(x_n) - f(x_{n-1})}{(x_n - x_{n-1})}$ formülü kullanılırsa

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$$

Bu yöntemde iterasyona başlamak için iki noktanın verilmiş olması gerekmektedir. İlk tahmin olarak $f(x_0)$ ve $f(x_{-1})$ olmak üzere iki başlangıç değerinin verilmesi gerekir.

Bunun yerine tek bir başlangıç noktası için de decant yöntemi denklemini oluşturabiliriz. Çok küçük bir Δx aralığı üzerinden fark denklemi

$$f'(x_n) \cong \frac{f(x_n + \Delta x) - f(x_n - \Delta x)}{2\Delta x}$$

şeklinde tanımlanacak olursa (merkezi fark denklemini kullanacak olursak), kiriş denklemi

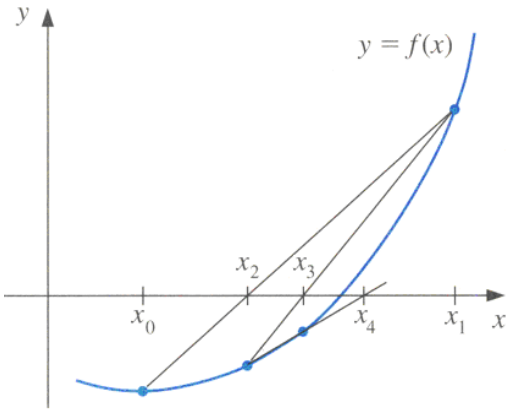
$$x_{n+1} = x_n - \frac{2f(x_n)\Delta x}{f(x_n + \Delta x) - f(x_n - \Delta x)}$$

şeklini alır. Hatanın az olması için Δx değerinin göreceli olarak küçük olması gereklidir. Hata $(\Delta x)^2$ ile orantılı olacaktır. Daha az sayıda döngü kullanarak kökleri bulmak istersek daha yüksek dereceden türev(fark denklemi) yaklaşımları kullanabiliriz.

$$f'(x_n) \cong \frac{-f(x_n + 2\Delta x) + 8f(x_n + \Delta x) - 8f(x_n - \Delta x) + f(x_n - 2\Delta x)}{12\Delta x}$$

$$f'(x_n) \cong \frac{f(x_n + 3\Delta x) - 9f(x_n + 2\Delta x) + 45f(x_n + \Delta x) - 45f(x_n - \Delta x) + 9f(x_n - 2\Delta x) - f(x_n - 3\Delta x)}{60\Delta x}$$

$$f'(x_n) \cong \frac{-3f(x_n + 4\Delta x) + 32f(x_n + 3\Delta x) - 168f(x_n + 2\Delta x) + 672f(x_n + \Delta x) - 672f(x_n - \Delta x) + 168f(x_n - 2\Delta x) - 32f(x_n - 3\Delta x) + 3f(x_n - 4\Delta x)}{840\Delta x}$$



(bolum2_09.jpg,Resimaltı: Kiriş yöntemi yakınsamasının grafiksel gösterimi)

Program 2.5.1 Kiriş yöntemi(iki nokta verilmiş)

```
// Bilgisayar programlama II ve Sayısal Çözümleme
```

```
// Sekant yöntemi ile kök bulmak
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

//============== Tanımı gereken fonksiyon =================

class fb extends f_x
{
   public double func (double x)
   { return Math.sin(Math.sqrt(x))-x;}
}
public class FSCO4D
{
public static double sekant(f_x f,double x0,double x1)
{
int nmax=500;
double tolerance=1.0e-10;
double fx0,fx1;
double x;
fx0=f.func(x0);
for(int i=0;i<nmax;i++)
{
  fx1=f.func(x1);
  x=x1-fx1*(x1-x0)/(fx1-fx0);
  x0=x1;
  fx0=fx1;
  x1=x;
  if(Math.abs(fx1)<tolerance) { return x1;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x1;
}

    public static void main (String args[])
    {
            fb f=new fb();
    double x0,x1;
    x0=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyon değeri için ilk tahmin : "));
    x1=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyon değeri için ikinci tahmin : "));
    double r= sekant(f,x0,x1);
    String s1="Sekant yöntemiyle kök hesabı : \n kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r);
    System.out.println(s1);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO4D
```

```
Sekant yöntemiyle kök hesabı :
 kök değeri  : 1.4142135623730951
Fonksiyon değeri : 4.440892098500626E-16
> Terminated with exit code 0.
```

İki nokta verildiğinde kiriş metoduyla hesap yapan bir de excel çıktısı oluşturduk. Fonksiyonumuz $f(x)=x^2-2$, başlangıç değerleri $x_{-1}=0$, $x_0=1$

Program 2.5.2 Kiriş yöntemi(iki nokta verilmiş) excel versiyonu

Kiriş $f(x)=x^2-a$

a	2	
x	f(x)=x*x-a	$fx/dx=(f(x_n)-f(x_n-1))/(x_n-x_{n-1})$
0.00000000000000000	-2	
1.00000000000000000	-1	1
2.00000000000000000	2	3
1.33333333333333000	-0.222222222	3.333333333
1.40000000000000000	-0.04	2.733333333
1.41463414634146000	0.001189768	2.814634146
1.41421143847487000	-6.00729E-06	2.828845585
1.41421356205732000	-8.93146E-10	2.828425001
1.41421356237310000	0	2.828424244
1.41421356237310000	0	

İkinci programımız kiriş denklemini tek girdi ve $f'(x_n) \cong \frac{f(x_n+\Delta x)-f(x_n-\Delta x)}{2\Delta x}$ fark denklemini kullanarak hesaplayacak şekilde oluşturuldu.

Program 2.5.3 Kiriş yöntemi(tek nokta verilmiş, birinci dereceden merkezi fark denklemi ile)

```
 ///Bilgisayar programlama II ve Sayısal Çözümleme
//
//
// Sekant yöntemi ile kök bulmak
//
//

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

//============= Tanımı gereken fonksiyon ================

class fb extends f_x
{
   public double func (double x)
   { return x*x-2;}
}
public class FSCO4
{
public static double sekant(f_x f,double x)
{
int nmax=100;
double tolerance=1.0e-10;
double fx,dfx;
double dx=0.01;
for(int i=0;i<nmax;i++)
{
  fx=f.func(x);
  dfx=(f.func(x+dx)-f.func(x-dx))/(2.0*dx);
  x-=fx/dfx;
  if(Math.abs(fx)<tolerance) { return x;}
```

```
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}

    public static void main (String args[])
    {
              fb f=new fb();
    double x0;
    x0=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyon değeri için ilk tahmin : "));
    double r= sekant(f,x0);
    String s1="tek noktalı kiriş yöntemiyle kök hesabı : \n kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r);
    System.out.println(s1);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO4
tek noktalı kiriş yöntemiyle kök hesabı :
 kök değeri  : 1.4142135623730951
Fonksiyon değeri : 4.440892098500626E-16
> Terminated with exit code 0.
```

Türev denklemi yerine fark denklemi kullanmayı bir adım daha ileri götürebiliriz. Gerçek türeve yaklaşımı çok daha fazla olacak bir fark denklemini kullanabiliriz. Örneğin fark denklemi olarak

$$f'(x_n) \cong \frac{-f(x_n+2\Delta x)+8f(x_n+\Delta x)-8f(x_n-\Delta x)+f(x_n-2\Delta x)}{12\Delta x}$$

formülünü kullanarak hata miktarını $(\Delta x)^2$ den $(\Delta x)^4$ seviyesine düşürebiliriz.

Program 2.5.4 Kiriş yöntemi(tek nokta verilmiş, ikinci dereceden merkezi fark denklemi ile)

```
 /
//
// Kiriş yöntemi ile kök bulmak
//
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

//============= Tanımı gereken fonksiyon ================

class fb extends f_x
{
  public double func (double x)
  { return x*x-2;}
}
//=========================================================

public class FSCO4A1
{

public static double df(f_x f,double x,double h)
{ //Türev hesabı
  double hh=1.0/h;
  return (-f.func(x+2.0*h)+8.0*f.func(x+h)-8.0*f.func(x-h)+f.func(x-2.0*h))/12.0*hh;
}

public static double newton(f_x f,double x)
{
int nmax=500;
```

```
double tolerance=1.0e-15;
double fx,dfx;
for(int i=0;i<nmax;i++)
{
  fx=f.func(x);
  dfx=df(f,x,0.00001);
  x-=fx/dfx;
  if(Math.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}

    public static void main (String args[])
    {
            fb f=new fb();
    double x0;
    x0=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyon değeri için ilk tahmin : "));
    double r= newton(f,x0);
    String s1="tek noktalı ikinci dereceden fark denklemli kiriş yöntemiyle kök hesabı : \n";
    s1+= "kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r);
    System.out.println(s1);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO4A1
tek noktalı ikinci dereceden fark denklemli kiriş yöntemiyle kök hesabı :
kök değeri  : 1.414213562373095
Fonksiyon değeri : -4.440892098500626E-16
> Terminated with exit code 0.
```

PROBLEM 2.5-.5 Kiriş yöntemi(tek nokta verilmiş, ikinci dereceden merkezi fark denklemi ile) excel çıktısı

Secant 3 $f(x)=x^2 - a$

a	2									
Δx	0.1									
x	f(x)=x*x-a	x+2Δx	x+Δx	x-Δx	x-2Δx	f(x+2Δx)	f(x+Δx)	f(x-Δx)	f(x-2Δx)	f'(x)
1	-1	1.2	1.1	0.9	0.8	-0.56	-0.79	-1.19	-1.36	2.99167
1.334261838	-0.21975	1.53426	1.43426	1.23426	1.13426	0.35396	0.05711	-0.4766	-0.71345	2.08121
1.439846995	0.07316	1.63985	1.53985	1.33985	1.23985	0.6891	0.37113	-0.20481	-0.46278	1.71622
1.397218859	-0.04778	1.59722	1.49722	1.29722	1.19722	0.55111	0.24166	-0.31722	-0.56667	1.86805
1.422795983	0.02435	1.6228	1.5228	1.3228	1.2228	0.63347	0.31891	-0.25021	-0.50477	1.77768
1.409099259	-0.01444	1.6091	1.5091	1.3091	1.2091	0.5892	0.27738	-0.28626	-0.53808	1.82635
1.417005353	0.0079	1.61701	1.51701	1.31701	1.21701	0.61471	0.30131	-0.2655	-0.5189	1.79833
1.412610076	-0.00453	1.61261	1.51261	1.31261	1.21261	0.60051	0.28799	-0.27705	-0.52958	1.81393
1.415108941	0.00253	1.61511	1.51511	1.31511	1.21511	0.60858	0.29556	-0.27049	-0.52351	1.80507
1.413705498	-0.00144	1.61371	1.51371	1.31371	1.21371	0.60405	0.2913	-0.27418	-0.52692	1.81005
1.414499268	0.00081	1.6145	1.5145	1.3145	1.2145	0.60661	0.29371	-0.27209	-0.52499	1.80724
1.414052077	-0.00046	1.61405	1.51405	1.31405	1.21405	0.60516	0.29235	-0.27327	-0.52608	1.80882
1.414304575	0.00026	1.6143	1.5143	1.3143	1.2143	0.60598	0.29312	-0.2726	-0.52546	1.80793
1.414162185	-0.00015	1.61416	1.51416	1.31416	1.21416	0.60552	0.29269	-0.27298	-0.52581	1.80843
1.414242539	8.2E-05	1.61424	1.51424	1.31424	1.21424	0.60578	0.29293	-0.27277	-0.52562	1.80815
1.414197211	-4.6E-05	1.6142	1.5142	1.3142	1.2142	0.60563	0.29279	-0.27289	-0.52573	1.80831
1.414222786	2.6E-05	1.61422	1.51422	1.31422	1.21422	0.60572	0.29287	-0.27282	-0.52566	1.80822
1.414208358	-1.5E-05	1.61421	1.51421	1.31421	1.21421	0.60567	0.29283	-0.27286	-0.5257	1.80827
1.414216498	8.3E-06	1.61422	1.51422	1.31422	1.21422	0.60569	0.29285	-0.27283	-0.52568	1.80824

Aynı problem Matlab (Octave) ortamında çözülürse:

PROBLEM 2.5-6 Kiriş yöntemi(tek nokta verilmiş, ikinci dereceden merkezi fark denklemi ile) Matlam m fonksiyon dosyaları

```
function p=df(f,x,dx)
```

```
% derivative of a function
if nargin<3
   dx=0.001;
end
p = ( -3*f(x+4*dx)+32*f(x+3*dx)-168*f(x+2*dx)+672*f(x+dx)-672*f(x-dx) + 168*f(x-2*dx)-32*f(x-3*dx)+3*f(x-4*dx) )/(dx*840);
end
```

```
  function kiris(f,x,tolerance,nmax)
% secant method for root finding
if nargin<4 nmax=100;end
if nargin<3 tolerance=1e-10;end
i=0;
fx=f(x);
while (abs(fx) > tolerance) && i<nmax
   fx=f(x);
   dfx=df(f,x);
   x=x-fx/dfx;
   i=i+1;
end
p=x;
if i>=nmax fprintf('maksimum döngü sayısı aşıldı sonuç hatalı olabilird');end
end
```

```
f=@(x) x*x-2;
>> kiris(f,1)
ans =1.414213562373095
```

PROBLEM 2.5-7 Kiriş yöntemi(tek nokta verilmiş, ikinci dereceden merkezi fark denklemi ile) c++ versiyonu

```
#include <iostream>
#include <math.h>

double func(double x)
{               double y=x*x-2.0;
                return y;
}

double dfunc(double x,double h)
{ double hh=1.0/h;
  return (-func(x+2.0*h)+8.0*func(x+h)-8.0*func(x-h)+func(x-2.0*h))/12.0*hh;
}

double kiris(double x)
{
int nmax=100;
double tolerance=1.0e-8;
double fx,dfx;
double h=0.0001;
for(int i=0;i<nmax;i++)
{
  fx=func(x);
  dfx=dfunc(x,h);
  x-=fx/dfx;
  std::cout <<"i="<<i<<"x="<<x<<"fx="<<fx<<"dfx="<<dfx<<"\n";
  if(fabs(fx)<tolerance) { return x;}
}
std::cout <<("Maksimum döngü sayısı aşılı sonuç hatalı olabilir \n");return x;
return x;
}

int main()
{  double xa,r;
   std::cout << " kök ilk tahmin değerini giriniz : ";
   std::cin >>xa;

   r=kiris(xa);
   std::cout<<"kök değeri : ="<<r;
   return 0;
}
```

```
C:\Users\asus-pc\Desktop\Say□sal €”z□mleme - M. Turhan €oban\SCO1>kiris
kiris
 kök ilk tahmin değerini giriniz : 1
```

```
i=0x=1.5fx=-1dfx=2
i=1x=1.41667fx=0.25dfx=3
i=2x=1.41422fx=0.00694444dfx=2.83333
i=3x=1.41421fx=6.0073e-006dfx=2.82843
i=4x=1.41421fx=4.51061e-012dfx=2.82843
kök değeri : =1.41421
C:\Users\asus-pc\Desktop\Say□sal €"z□mleme - M. Turhan €oban\SCO1>
```

2.24 Aitken Extrapolasyon Düzeltmeli Newton-Raphson Yöntemi

Aitken extrapolasyonu Newton raphson hesaplamasındaki hataları azaltarak kök bulma olasılığını artıran bir hata azaltma yöntemidir. Temel olarak k çok büyük bir değer olduğunda

$$\frac{x_{k+2}-a}{x_{k+1}-a} \approx \frac{x_{k+1}-a}{x_k-a} \qquad k >> 1$$

bu bağıntıdan dördüncü bir noktanın değerini bulmak istersek

$$x_e = \frac{x_k x_{k+2} - x_{k+1}^2}{x_k - 2x_{k+1} + x_{k+2}}$$

3 önceki nokta bilindiğinde 4üncü noktayı tahmin edebiliriz.

$\Delta x_k = x_{k+1} - x_k$

$\Delta^2 x_k = x_{k+2} - 2x_{k+1} + x_k$

$$x_e = x_k - \frac{(\Delta x_k)^2}{\Delta^2 x_k}$$

Şimdi bu formülü Newton metodundan elde ettiğimiz ilk yaklaşımları daha iyileştirmek için kullanabiliriz.

Program 2.6-1 Aitken ekstrapolasyonlu Newton yöntemi

```
 // Newton-Raphson_Aitken yöntemi ile kök bulmak

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

//============= Tanımı gereken fonksiyon ================

class fb extends f_x
{
  public double func (double x)
  { return Math.log(x*x)-0.7;}
}

class dfb extends f_x
{
  public double func (double x)
  { return 2.0/x;}
}
//============================================================
public class FSCO4B
{
public static double newton_aitken(f_x f,f_x df,double x)
{
// newton metodu aitken extrapolasyonuyla birlikte
int nmax=500;
double tolerance=1.0e-10;
double fx,dfx;
double xm1,xm2;
```

```
xm1=x;
xm2=x;
int j=0;;
for(int i=0;i<3;i++)
{  if(i==0) xm2=x;
   else if(i==1) xm1=x;
   fx=f.func(x);
   dfx=df.func(x);
   x-=fx/dfx;
   if(Math.abs(fx)<tolerance) { return x;}
}
for(int i=3;i<nmax;i++)
{   x=(xm2*x-xm1*xm1)/(xm2-2.0*xm1+x); //extrapolasyon
    fx=f.func(x);
    if(Math.abs(fx)<tolerance) { return x;}
    xm2=xm1;
    xm1=x;
    dfx=df.func(x);
    x-=fx/dfx;
    fx=f.func(x);
    if(Math.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
    public static void main (String args[])
    {
              fb f=new fb();
              dfb df=new dfb();
    double x0;
    x0=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyon değeri için ilk tahmin : "));
    double r= newton_aitken(f,df,x0);
    String s1= "Newton-Raphson-Aitken ektrapolasyon yöntemiyle kök hesabı : \n";
    String s2=" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r);
    System.out.println(s1+s2);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO4B
Newton-Raphson-Aitken ektrapolasyon yöntemiyle kök hesabı :
 kök değeri  : 1.419067548587509
Fonksiyon değeri : -8.101186388387305E-12
> Terminated with exit code 0.
```

Program 4.6.2 Aitken ekstrapolasyonlu Newton yöntemi (C++ versiyonu)

```
#include <iostream>
#include <math.h>

double func(double x)
{           double y=x*x-2.0;
            return y;
}

double dfunc(double x)
{ double y=2.0*x;
            return y;
}

double newton_aitken(double x)
{
// newton metodu aitken extrapolasyonuyla birlikte
int nmax=500;
double tolerance=1.0e-10;
double fx,dfx;
```

```
double xm1,xm2;
xm1=x;
xm2=x;
//Newton stebi
for(int i=0;i<3;i++)
{  if(i==0) xm2=x;
   else if(i==1) xm1=x;
   fx=func(x);
   dfx=dfunc(x);
   x-=fx/dfx;
   if(fabs(fx)<tolerance) { return x;}
}
//Aitken stebi
for(int i=3;i<nmax;i++)
{   x=(xm2*x-xm1*xm1)/(xm2-2.0*xm1+x); //extrapolasyon
    fx=func(x);
    if(fabs(fx)<tolerance) { return x;}
    xm2=xm1;
    xm1=x;
    dfx=dfunc(x);
    x-=fx/dfx;
    fx=func(x);
    if(fabs(fx)<tolerance) { return x;}
}
std::cout <<("Maksimum döngü sayısı aşılı sonuç hatalı olabilir \n");
return x;
}
int main()
{  double xa,r;
   std::cout << " kök ilk tahmin değerini giriniz : ";
   std::cin >>xa;
   r=newton_aitken(xa);
   std::cout<<"kök değeri : ="<<r;
   return 0;
}
```

```
C:\Users\asus-pc\Desktop\Say□sal €”z□mleme - M. Turhan €oban\SCO1>newton_aitken
newton_aitken
 kök ilk tahmin değerini giriniz : 1
kök değeri : =1.41421
C:\Users\asus-pc\Desktop\Say□sal €”z□mleme - M. Turhan €oban\SCO1>
```

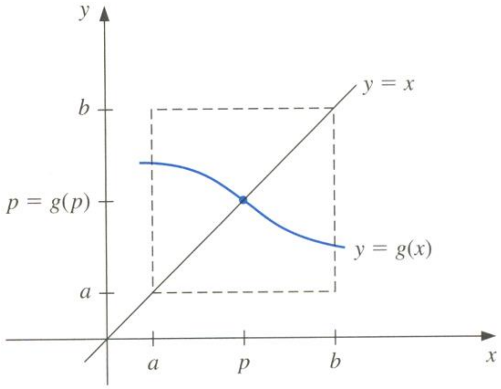
Program 2.6.3 Aitken ekstrapolasyonlu Newton yöntemi, türev yerine fark denklemi kullanılmış hali

```
// Newton-Raphson_Aitken yöntemi ile kök bulmak
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============= Tanımı gereken fonksiyon ================

class fb extends f_x
{
   public double func (double x)
   { return Math.log(x*x)-0.7;}
}
public class FSCO4B1
{
public static double df(f_x f,double x,double h)
{ //türev
double hh=1.0/h;
  return (-f.func(x+2.0*h)+8.0*f.func(x+h)-8.0*f.func(x-h)+f.func(x-2.0*h))/12.0*hh;
}

public static double newton_aitken(f_x f,double x)
{
// newton metodu aitken extrapolasyonuyla birlikte
int nmax=500;
double tolerance=1.0e-10;
```

```
double fx,dfx;
double xm1,xm2;
double dx=0.00001;
xm1=x;
xm2=x;
int j=0;;
for(int i=0;i<3;i++)
{  if(i==0) xm2=x;
   else if(i==1) xm1=x;
   fx=f.func(x);
   dfx=df(f,x,dx);
   x-=fx/dfx;
   if(Math.abs(fx)<tolerance) { return x;}
}
for(int i=3;i<nmax;i++)
{   x=(xm2*x-xm1*xm1)/(xm2-2.0*xm1+x); //extrapolasyon
    fx=f.func(x);
    if(Math.abs(fx)<tolerance) { return x;}
    xm2=xm1;
    xm1=x;
    dfx=df(f,x,dx);
    x-=fx/dfx;
    fx=f.func(x);
    if(Math.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}

    public static void main (String args[])
    {
            fb f=new fb();
    double x0;
    x0=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyon değeri için ilk tahmin : "));
    double r= newton_aitken(f,x0);
    JOptionPane.showMessageDialog(null," kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r),
    "Newton-Raphson-Aitken ektrapolasyon yöntemiyle kök hesabı : ",JOptionPane.PLAIN_MESSAGE);
    System.exit(0);
    }
}
```

2.25 Aitken Ekstrapolasyon Düzeltmeli Direk Yerine Koyma : Steffensen metodu

Kök bulmada yakınsamada zorluklar olabilmesi sebebiyle çok fazla kullanılmayan bir metod da direk yerine koyma metodudur. Bu metodda f(x) fonksiyonundan bir x değeri çekilerek f(x)=0 fonksiyonu, x=g(x) fonksiyonuna dönüştürülür. Buna sabit nokta iterasyonu veya direk yerine koyma iterasyonu ismi verilir.



(bolum2_010.jpg,Resimaltı: direk yerine koyma metodunun grafik gösterimi)

Aitken interpolasyon prosesinden yararlanarak direk yerine koyma metodu iyileştirilebilir. Metod ilk olarak Steffensen tarafından önerilmiştir. $p_0^{(0)}$ ilk tahmin değeri verildiğinde, $p_1^{(0)}=g(p_0^{(0)})$ ve , $p_2^{(0)}=g(p_1^{(0)})$ normal direk yerine koyma adımı olarak hesaplanır, sonra bir Aitken interpolasyon stepi kullanılır. Bu metod şu şekilde tanımlanmıştır : Temel olarak k çok büyük bir değer olduğunda

$$\frac{x_{k+2}-a}{x_{k+1}-a} \approx \frac{x_{k+1}-a}{x_k-a} \qquad k >> 1$$

bu bağıntıdan dördüncü bir noktanın değerini bulmak istersek

$$x_e = \frac{x_k x_{k+2} - x_{k+1}^2}{x_k - 2x_{k+1} + x_{k+2}}$$

3 önceki nokta bilindiğinde 4üncü noktayı tahmin edebiliriz.

$\Delta x_k = x_{k+1} - x_k$

$\Delta^2 x_k = x_{k+2} - 2x_{k+1} + x_k$

$$x_e = x_k - \frac{(\Delta x_k)^2}{\Delta^2 x_k}$$

Stephensen metodunda aitken düzeltme stebi aşağıdaki formülle verilir.
$p_0^{(1)} = p_0^{(0)} - (p_1^{(0)} - p_0^{(0)})^2/(p_2^{(0)} - 2\, p_1^{(0)} + p_0^{(0)})$ bundan sonra yine
$p_1^{(1)}=g(p_0^{(1)})$ ve , $p_2^{(1)}=g(p_1^{(1)})$ stepleriyle devam edilir.

Program 2.7.1 Stephensen kök bulma metodu (Aitken ekstrapolasyonlu direk yerine koyma metodu)

```
 ///Bilgisayar programlama II ve Sayısal Çözümleme
// Stephensen metodu x=g(x) verildiğinde x değerini bulur.

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============== Tanımı gereken fonksiyon ================

class fb extends f_x
{  //x = g(x) fonksiyonu
   public double func (double x)
   { return Math.sqrt(10.0/(x+4.0));}
}

//============================================================
public class FSCO3A
{

public static double stephensen(f_x f,double x0)
{
//Stephensen kök bulma metodu
double es,ea;
int maxit=100,iter=0;
es=0.000001;
ea=1.1*es;
double x,x1,x2;
```

```
x=0;
while(iter<maxit && ea>es)
{
x1=f.func(x0);
x2=f.func(x1);
x=x0-(x1-x0)*(x1-x0)/(x2-2*x1+x0);
ea=Math.abs(x-x0)*100.0;
x0=x;
iter++;
}
if(iter>=maxit) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
    public static void main (String args[])
    {
            fb g=new fb();
    double xa,xu;
    xa=Double.parseDouble(JOptionPane.showInputDialog(" ilk tahmin değerini giriniz : "));
    double r= stephensen(g,xa);
    double fr=g.func(r);
    String s=" kök değeri  : "+r+"\ng(x) = "+fr+"\nf(x)=g(x)-x = "+(fr-r);
        String s1= "Stephensen kök bulma metodu :  \n";
    System.out.println(s1+s);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO3A
Stephensen kök bulma metodu :
 kök değeri  : 1.3652300134140969
g(x) = 1.3652300134140969
f(x)=g(x)-x = 0.0
> Terminated with exit code 0.
```

Stephensen Metodunu excel ortamında oluşturalım. Problemimiz:

$$f(x) = x - \sqrt{\frac{5}{x+1}} = x - g(x)$$

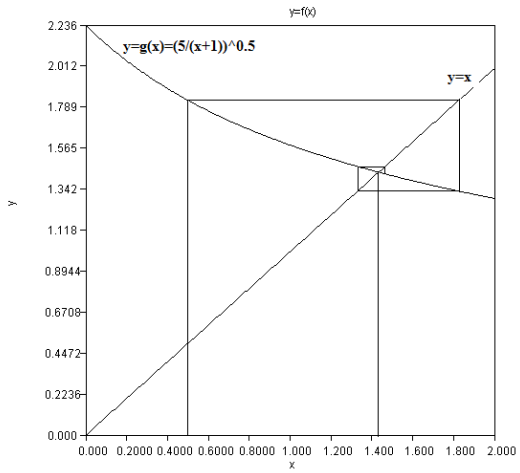
$g(x) = \sqrt{\frac{5}{x+1}}$ olsun.

Program 2.7.2 : Stephensen methodu excel örneği:

f(x)=x-(5/(x+1)^0.5

Aitken		dxk	d2xk	xe=xk-(dxk)^2/d2xk	g(x)=(5/(x+1))^0.5
	x				
0	0.5				1.825741858
0	1.825741858	1.325741858			1.330205563
0	1.330205563	-0.495536296	-1.821278154	1.465031877	1.464832223
1	1.465031877	0.134826314	0.63036261	1.436194292	1.424209021
0	1.424209021	-0.040822856	-0.17564917	1.433696715	1.436150556
0	1.436150556	0.011941535	0.052764391	1.433447971	1.432626367
0	1.432626367	-0.003524189	-0.015465724	1.433429427	1.433663727
1	1.433429427	0.00080306	0.004327249	1.433280394	1.433427145
0	1.433427145	-2.28264E-06	-0.000805343	1.433427151	1.433427817
0	1.433427817	6.72303E-07	2.95494E-06	1.433427664	1.433427619
0	1.433427619	-1.98012E-07	-8.70315E-07	1.433427664	1.433427677
1	1.433427664	4.50514E-08	2.43064E-07	1.433427656	1.433427664

0	1.433427664	-7.32747E-15	-4.50514E-08	1.433427664	1.433427664
0	1.433427664	1.9984E-15	9.32587E-15	1.433427664	1.433427664
0	1.433427664	0	-2.44249E-15	1.433427664	1.433427664



(bolum2_011.jpg,Resimaltı:Örnek problemin döngü adımlarının görünümü)

Aynı problemi matlab ortamında tanımlarsak:

Program 2.7.3 : Direk yerine koyma (Stephensen metodu) Matlab(Octave) programı:

```
  function x = stephensen(f,x0)
% Fixed point iteration with Aitken extrapolasyonlu
% direk yerine koyma
% Stephensen metodu
maxit=100;
iter=0;
es=0.000001;
ea=1.1*es;
x=0;
while iter<maxit && ea>es
x1=f(x0);
x2=f(x1);
x=x0-(x1-x0)*(x1-x0)/(x2-2*x1+x0);
ea=abs(x-x0)*100.0;
x0=x;
iter=iter+1;
end
if iter>=maxit  fprintf('Maksimum döngü sayısı aşıldı sonuç hatalı olabilir');end
end
```

```
>> g=@(x) sqrt(5/(x+1));
>> stephensen(g,1)
ans =
   1.433427663863820
```

Aynı problemin C++ cdilindeki çözümü:

Program 2.7.4 : Stephensen metodu c++ dili

```
 #include <iostream>
#include <math.h>

double func(double x)
{ return sqrt(5.0/(x+1.0));}

double stephensen(double x0)
{
//Stephensen root finding method
double es,ea;0

int maxit=100,iter=0;
es=0.000001;
```

```
ea=1.1*es;
double x,x1,x2;
x=0;
while(iter<maxit && ea>es)
{
x1=func(x0);
x2=func(x1);
x=x0-(x1-x0)*(x1-x0)/(x2-2*x1+x0);
ea=fabs(x-x0)*100.0;
x0=x;
iter++;
}
if(iter>=maxit) std::cout <<("Maksimum döngü sayısı aşıldı sonuç hatalı olabilir");
return x;
}
int main()
{  double xa,r;
  std::cout << " first guess for the root value : ";
  std::cin >>xa;

  r=stephensen(xa);
  std::cout<<"root value : ="<<r<<"fr="<<func(r)-r;
  return 0;
}
```

```
C:\Users\asus-pc\Desktop\Say□sal €"z□mleme - M. Turhan €oban\SCO1>stephensen
stephensen
 first guess for the root value : 1
root value : =1.43343fr=0
C:\Users\asus-pc\Desktop\Say□sal €"z□mleme - M. Turhan €oban\SCO1>
```

2.26İkinci ve daha üst türev dereceli Yöntemler

Taylor denklemini tekrar ele alalım, ancak bu sefer bir yerine iki terimin denklemde yerini aldığını varsayalım.

$$f(x_{n+1}) = 0 = f(x_n) + \frac{f'(x_n)}{1!}(x_{n+!} - x_n) + \frac{f''(x_n)}{2!}(x_{n+!} - x_n)^2 + \frac{f'''(x_n)}{3!}(x_{n+!} - x_n)^3 + ..$$

Kök bulma formülü aşağıdaki formu alacaktır.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{1}{2}\frac{f''(x_n)}{f'(x_n)} = x_n - \frac{2f(x_n) * f'(x_n)}{2f'(x_n) * f'(x_n) - f(x_n) * f''(x_n)}$$

Bu metod **Halley** metodu adını alır. Bu ve bunun benzeri bir çok birinci dereceden daha fazla derecelerde türevleri içeren kök bulma formülleri literatürde görülebilmektedir. Bu bölümde bunlardan bazılarına yer vereceğiz. Olver metodu üçüncü dereceden hata payı oluşturan ikinci derece bir formüldür.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{1}{2}\frac{f(x_n)f(x_n)f''(x_n)}{f'(x_n)f'(x_n)f'(x_n)}$$

Abbasbandy[48] diğer bir iteratif formül geliştirmiştir

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f(x_n) * f(x_n) * f''(x_n)}{2f'(x_n) * f'(x_n) * f'(x_n)} - \frac{f(x_n) * f(x_n) * f(x_n) * f'''(x_n)}{6 * f'(x_n) * f'(x_n) * f'(x_n) * f'(x_n)}$$

Li ve Wang[49] dördüncü dereceden kök formülleri geliştirmiş ve bu metodlara Thiele 1 ve Thiele 2 metodları ismini vermişlerdir. Bu ismin temel nedeni formülasyonun THiele'nin sürekli bölümler

metodundan yararlanılarak elde edilmiş olmasıdır. Denklemlerin genel formları altta verilmiştir

$$x_{n+1} = x_n - \frac{P_1(x_n)}{Q_1(x_n)}$$

$$P_1(x) = 4f(x)f'(x)[3f'(x)f'(x)f''(x) - 3f(x)f''(x)f''(x) + f(x)f'(x)f'''(x)]$$

$$Q_1(x) = 12f'(x)f'(x)f'(x)f'(x)f''(x) - 18f(x)f'(x)f'(x)f''(x)f''(x) + 3f(x)f(x)f''(x)f''(x)f''(x) + 4f(x)f'(x)f'(x)f'(x)f'''(x)$$

$$x_{n+1} = x_n - \frac{P_2(x_n)}{Q_2(x_n)}$$

$$P_2(x) = f(x)[6f'(x)f'(x) - 3f(x)f''(x)f''(x) + 2f(x)f'(x)f'''(x)]$$

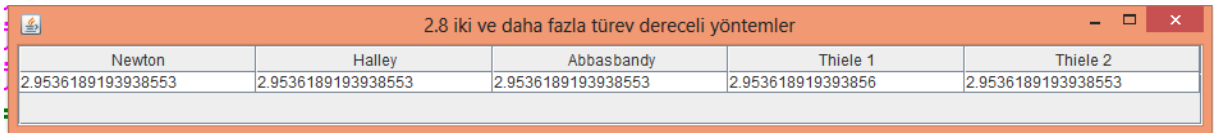
$$Q_2(x) = 2f'(x)[3f'(x)f'(x)f''(x) - 3f(x)f''(x)f''(x) + f(x)f'(x)f'''(x)]$$

Program 2.8.1: Yüksek türev dereceli kök bulma formülleri

```
//  kök bulma metodları Halley, Abbasbandy Thiele1 and Thiele2
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============== function to be defined ================

class fb extends f_x
{
   public double func (double x)
   { return x*x*x+3.6*x-36.4;}
}
//============================================================
public class NA5B
{
public static double newton(f_x f,double x)
{
int nmax=500;
double tolerance=1.0e-8;
double fx,dfx;
for(int i=0;i<nmax;i++)
{  fx=f.func(x);
   dfx=f.dfunc(x);
   x-=fx/dfx;
  if(Math.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
public static double halley(f_x f,double x)
{
int nmax=500;
double tolerance=1.0e-8;
double fx,dfx,fxm,dfxm,d2fx;
for(int i=0;i<nmax;i++)
{  fx=f.func(x);
   dfx=f.dfunc(x);
   d2fx=f.dfunc(x,2);
   x-=2.0*fx*dfx/(2.0*dfx*dfx-fx*d2fx);
  if(Math.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);

return x;
```

```
}
public static double abbasbandy(f_x f,double x)
{
int nmax=500;
double tolerance=1.0e-8;
double fx,dfx,d2fx,d3fx;
for(int i=0;i<nmax;i++)
{  fx=f.func(x);
   dfx=f.dfunc(x);
   d2fx=f.dfunc(x,2);
   d3fx=f.dfunc(x,3);
   x-=(fx/dfx+fx*fx*d2fx/(2.0*dfx*dfx*dfx)+fx*fx*fx*d3fx/(6.0*dfx*dfx*dfx*dfx));
  if(Math.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);

   return x;
}

public static double TM1(f_x f,double x)
{ //Thiele 1 method
int nmax=500;
double tolerance=1.0e-8;
double fx,dfx,d2fx,d3fx;
double P1,Q1;
for(int i=0;i<nmax;i++)
{  fx=f.func(x);
   dfx=f.dfunc(x);
   d2fx=f.dfunc(x,2);
   d3fx=f.dfunc(x,3);
   P1=4.0*fx*dfx*(3.0*dfx*dfx*d2fx-3.0*fx*d2fx*d2fx+fx*dfx*d3fx);
   Q1=12.0*dfx*dfx*dfx*dfx*d2fx-18.0*fx*dfx*dfx*d2fx*d2fx+3.0*fx*fx*d2fx*d2fx*d2fx+4.0*fx*dfx*dfx*dfx*d3fx;
   x-=P1/Q1;
  if(Math.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}

public static double TM2(f_x f,double x)
{ //Thiele 2 method
int nmax=500;
double tolerance=1.0e-8;
double fx,dfx,d2fx,d3fx;
double P2,Q2;
for(int i=0;i<nmax;i++)
{  fx=f.func(x);
   dfx=f.dfunc(x);
   d2fx=f.dfunc(x,2);
   d3fx=f.dfunc(x,3);
   P2=fx*(6.0*dfx*dfx*d2fx-3.0*fx*d2fx*d2fx+2*fx*dfx*d3fx);
   Q2=2.0*dfx*(3.0*dfx*dfx*d2fx-3.0*fx*d2fx*d2fx+fx*dfx*d3fx);
   x-=P2/Q2;
  if(Math.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);

return x;
}
    public static void main (String args[])
    {
            fb f=new fb();
            dfb df=new dfb();
    double x0;
```

```
    x0=Double.parseDouble(JOptionPane.showInputDialog(" first guess for x : "));
    double r[]=new double[5];
    String s[]={"Newton","Halley","Abbasbandy","Thiele 1","Thiele 2"};
    r[0]=newton(f,x0);
    r[1]= halley(f,x0);
    r[2]= abbasbandy(f,x0);
    r[3]= TM1(f,x0);
    r[4]= TM2(f,x0);
    Text.print(r,s,"2.8 iki ve daha fazla türev dereceli yöntemler");
    }
}
```



(bolum2_012.jpg,Resimaltı:)

Program 2.8.2: türev formülleri, matlab

```
function df=dnf(f,x,n,dx)
% degree nth derivative of a function f
if nargin<4
   dx=0.001;
end
if nargin<3
   dx=0.001;n=1;
end
dxh=1.0/dx;
if n==0 df=f(x);
elseif n==1
df=(3.0*f(x-4.0*dx)-32.0*f(x-3.0*dx)+168.0*f(x-2.0*dx)-672.0*f(x-dx)+672.0*f(x+dx)-
168.0*f(x+2.0*dx)+32.0*f(x+3.0*dx)-3.0*f(x+4.0*dx))/840.0*dxh;
elseif n==2
df=(-14350.0*f(x)-9.0*f(x-4.0*dx)+128*f(x-3.0*dx)-1008*f(x-2.0*dx)+8064*f(x-dx)+8064.0*f(x+dx)-
1008.0*f(x+2.0*dx)+128.0*f(x+3.0*dx)-9.0*f(x+4.0*dx))/5040.0*dxh*dxh;
elseif n==3
df=(-7.0*f(x-4.0*dx)+72.0*f(x-3.0*dx)-338.0*f(x-2.0*dx)+488.0*f(x-dx)-488.0*f(x+dx)+338.0*f(x+2.0*dx)-
72.0*f(x+3.0*dx)+7.0*f(x+4.0*dx))/240.0*dxh*dxh*dxh;
elseif n==4
df=(2730.0*f(x)+7.0*f(x-4.0*dx)-96.0*f(x-3.0*dx)+676.0*f(x-2*dx)-1952.0*f(x-dx)-1952.0*f(x+dx)+676.0*f(x+2.0*dx)-
96.0*f(x+3.0*dx)+7.0*f(x+4.0*dx))/240.0*dxh*dxh*dxh*dxh;
elseif n==5
df=(f(x-4.0*dx)-9.0*f(x-3.0*dx)+26.0*f(x-2.0*dx)-29.0*f(x-dx)+29.0*f(x+dx)-26.0*f(x+2.0*dx)+9.0*f(x+3.0*dx)-
f(x+4.0*dx))/6.0*dxh*dxh*dxh*dxh*dxh;
elseif n==6
df=(-150.0*f(x)-f(x-4.0*dx)+12.0*f(x-3.0*dx)-52.0*f(x-2.0*dx)+116.0*f(x-dx)+116.0*f(x+dx)-
52.0*f(x+2.0*dx)+12.0*f(x+3.0*dx)-f(x+4.0*dx))/4.0*dxh*dxh*dxh*dxh*dxh*dxh;
elseif n==7
df=(-f(x-4.0*dx)+6.0*f(x-3.0*dx)-14.0*f(x-2.0*dx)+14.0*f(x-dx)-14.0*f(x+dx)+14.0*f(x+2.0*dx)-
6.0*f(x+3.0*dx)+f(x+4.0*dx))/2.0*dxh*dxh*dxh*dxh*dxh*dxh*dxh;
elseif n==8
df=(70.0*f(x)+f(x-4.0*dx)-8.0*f(x-3.0*dx)+28.0*f(x-2.0*dx)-56.0*f(x-dx)-56.0*f(x+dx)+28.0*f(x+2.0*dx)-
8.0*f(x+3.0*dx)+f(x+4.0*dx))*dxh*dxh*dxh*dxh*dxh*dxh*dxh*dxh;

else df=0;end
end
```

Program 2.8.3: Thiele1 kök bulma

```
function xx = TM2(f,x)
%Thiele 2 Metodu
nmax=500;
tolerance=1.0e-8;
i=0;
f_x=1;
```

```
while   abs(f_x)>=tolerance && i<=nmax
   f_x=f(x);
   df_x=dnf(f,x,1);
   d2f_x=dnf(f,x,2);
   d3f_x=dnf(f,x,3);
   P2=f_x*(6.0*df_x*df_x*d2f_x-3.0*f_x*d2f_x*d2f_x+2*f_x*df_x*d3f_x);
   Q2=2.0*df_x*(3.0*df_x*df_x*d2f_x-3.0*f_x*d2f_x*d2f_x+f_x*df_x*d3f_x);
   x=x-P2/Q2;
   i=i+1;
  end
if(i>=nmax) fprintf('Maksimum döngü sayısı aşıldı sonuç geçersiz olabilir');end
xx=x;
end
```

```
>> f1=@(x)x*x*x+3.6*x-36.4;
>> TM2(f1,1)
ans =
   2.953618919393856
```

2.27 Newton-Raphson ve ikiye bölme metodlarının birlikte kullanılması

İkiye bölme metodunun bizim için en önemli avantajı, eğer tanımlanan bölgede bir kök mevcutsa ona muhakkak ulaşabilmesidir, ancak kök değerine ulaşması çok fazla döngü stebi gerektirir. Öte yandan Newton-raphson metodu eğer köke yakın olduğunda az bir iterasyon sayısıyla köke ulaşırken, kökten uzak bir ilk tahmin verildiğinde kökü hiç bulamayabilir veya çok uzun step sayısıyla köke doğru hareket eder. Bu iki metodu birleştirerek her metodun avantajlı kısmını bir arada kullanabiliriz, İkiye bölme stepleriyle iterasyona başlayıp, kök değerine yaklaştığımızda Newton-Raphson adımlarına geçebiliriz. Alttaki örnek programda Newton metodundaki türev değerleri de sayısal şekle dönüştürülerek sadece fonksiyonun girdi olarak alındığı bir formata dönüştürülmüştür. Bu şekilde oldukça etkin çalışabilen bir kök bulma sistemi oluşmuştur.

Program 2.9.1 Newton-Raphson-İkiye bölme kombine formülü, birinci türev sayısal olarak hesaplanmış

```
 //
//
// Newton-Raphson-ikiye bölme kombine yöntemi ile kök bulmak
//
//

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

//============= Tanımı gereken fonksiyon ================

class fb extends f_x
{
   public double func (double x)
   { return Math.log(x*x)-0.7;}
}
//============================================================

public class FSCO4A2
{

public static double dfx(f_x f,double x,double h)
{ //Türev hesabı
  double hh=1.0/h;
  return (-f.func(x+2.0*h)+8.0*f.func(x+h)-8.0*f.func(x-h)+f.func(x-2.0*h))/12.0*hh;
```

```
}

public static double newton_bisection(f_x fx, double x1, double x2)
{
double xacc=1.0e-10; //iterasyonlar arasındaki fark, bu rakamdan
// küçük olduğunda çözüme ulaştığımızı kabul edeceğiz
double h=0.0001; //türev step büyüklüğü
int MAXIT=100;  //Maksimum iterasyon sayısı
int j;
double dfr,dx,dxold,fr,fh,fl;
double temp,b,a,r;
fl=fx.func(x1);
fh=fx.func(x2);
//Bölgemizde kök değeri varmı kontrol et
if ((fl > 0.0 && fh > 0.0) || (fl < 0.0 && fh < 0.0))
System.out.println("verilen sınırlar içinde kök değeri yok");
if (fl == 0.0) return x1;
if (fh == 0.0) return x2;
if (fl < 0.0)   { a=x1;b=x2;}
else           { b=x1;a=x2;}
r=0.5*(x1+x2); //orta nokta değeri
dxold=Math.abs(x2-x1);
dx=dxold;
fr=fx.func(r);   //fonksiyonun r noktasındaki değeri
dfr=dfx(fx,r,h); //fonksiyonun türevinin r noktasındaki değeri
for (j=1;j<=MAXIT;j++)
{
if ((((r-b)*dfr-fr)*((r-a)*dfr-fr) > 0.0) || (Math.abs(2.0*fr) > Math.abs(dxold*dfr)))
   { //ikiye bölme
                dxold=dx;
     dx=0.5*(b-a);
     r=a+dx;
     if (a == r) return r; //çözüm!!!!
   }
else{ //Newton-Raphson
                dxold=dx;
     dx=fr/dfr;
     temp=r;
     r -= dx;
     if (temp == r) return r;  //çözüm!!!!!
   }
if (Math.abs(dx) < xacc) return r; //çözüm!!!!!
fr=fx.func(r);
dfr=dfx(fx,r,h);
if (fr < 0.0)
a=r;
else
b=r;
}
System.out.println("Maksimum iterasyon sayısı aşıldı");
return 0.0; //program buraya gelmemeli
}

   public static void main (String args[])
   {
            fb f=new fb();
   double x0,x1;
   x0=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyon değeri için alt tahmin sınırı : "));
   x1=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyon değeri için üst tahmin sınırı : "));
   double r= newton_bisection(f,x0,x1);
   String s1=" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r);
   String s2="Newton-Raphson - ikiye bölme kombine yöntemiyle kök hesabı : \n";
   System.out.println(s2+s1);
   System.exit(0);
   }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO4A2
Newton-Raphson - ikiye bölme kombine yöntemiyle kök hesabı :
 kök değeri  : 1.4190675485932571
Fonksiyon değeri : -1.1102230246251565E-16
> Terminated with exit code 0.
```

İkinci bir örnek problemi ısı transferinden tanımlayalım. Düz bir duvarın sıcaklığı

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial T^2}$$

denklemiyle verilmektedir. Bu denklem zamana bağlı bir boyutlu taşınım ısı transferini tanımlar. Buradaki T sıcaklık(derece Kelvin), x mesafe, t zaman (saniye) değişkenidir.

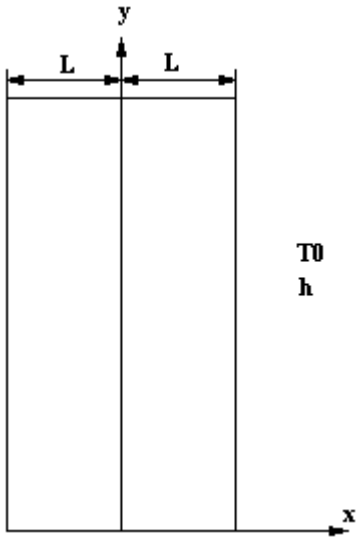
$$\alpha = \frac{k}{\rho C_p}$$

ısıl yayılım olarak tanımlanır ve formüldeki k ısıl iletim katsayısı (W/mK), ρ yoğunluk (kg/m^3), C_p özgül ısıdır(W/kgK). h ısı taşınım katsayısı adını alır. (W/m^{2}K). Diferansiyel denklemin çözüm seti

$$T(x,0) = T_i$$

$$\frac{\partial T}{\partial x}\Big|_{x=0} = 0$$

$$-k\frac{\partial T}{\partial x}\Big|_{x=L} = h\left[T(L,t) - T_\infty\right]$$



(bolum2_013.jpg,Resimaltı:Duvar ısı transferi problemi)

şeklinde yazılabilir. Bu denklemi boyutsuz formda yazarsak :

$$\frac{\partial \bar{\theta}}{\partial Fo} = \frac{\partial^2 \bar{\theta}}{\partial \bar{x}^2}$$

sınır şartları

$$\bar{\theta}(\bar{x},0) = 1$$

$$\frac{\partial \bar{\theta}}{\partial \bar{x}^2}\Big|_{\bar{x}=0} = 0$$

$$\frac{\partial \bar{\theta}}{\partial \bar{x}^2}\Big|_{\bar{x}=L} = -Bi\bar{\theta}(1, Fo)$$

$$\bar{\theta} = \frac{\theta}{\theta_i} = \frac{T - T_\infty}{T_i - T_\infty} \qquad \bar{x} = \frac{x}{L} \qquad \bar{t} = \frac{\alpha t}{L^2} = Fo$$

bu diferansiyel denklemi çözersek :

$$T = T(x, t, T_i, T_\infty, L, K, h, \alpha)$$

$$\bar{\theta} = \bar{\theta}(\bar{x}, Fo, Bi)$$

formunda bir çözüme ulaşırız. Buradaki Bi Biot sayısıdır.

$$Bi = \frac{hL}{k}$$

olarak tanımlanır. Bu denklemin çözümü

$$\bar{\theta} = \sum_{n=1}^{\infty} C_n e^{-\zeta_n^2 Fo} \cos(\zeta_n \bar{x})$$

şeklindedir. Burada

$$C_n = \frac{4\sin(\zeta_n)}{2\zeta_n + \sin(2\zeta_n)} \quad \text{ve} \quad \zeta_n \tan(\zeta_n) = Bi$$

şeklinde verilebilir. Bu denklemin çözülebilmesi denklemin kökünü bulma anlamına gelir. Denklemi çözmek için önce kökü buluruz, sonra C_n katsayılarını çözdükten sonra denklem çözülür. Bu işlemleri yapan ve çözümü veren program altta verilmiştir.

Program 2.9.2 Newton-Raphson-İkiye bölme kombine formülü, ile zamana bağlı duvar probleminin çözümü

```
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

public class HT_duvar
{

public static double fx(double lambda,double Bi)
{return lambda*Math.tan(lambda)-Bi;}

public static double dfx(double x,double h,double Bi)
{ //Türev hesabı
  return (-fx((x+2.0*h),Bi)+8.0*fx((x+h),Bi)-8.0*fx((x-h),Bi)+fx((x-2.0*h),Bi))/(12.0*h);
}

public static double Cn(double ln,double Bi)
{
 return 4.0*Math.sin(ln)/(2.0*ln+Math.sin(2.0*ln));
}

public static double lambda(double n,double Bi)
{
 //lineer olmayan denklemin kökü : ikiyebölme-Newton Raphson metodu
 //lambda*tan(lambda)-Bi=0 denkleminin köklerini bulur
 double x1=(n-1)*Math.PI;
 double x2=              x1+Math.PI/2-0.000001;
double xacc=1.0e-10; //iterasyonlar arasındaki fark, bu rakamdan
// küçük olduğunda çözüme ulaştığımızı kabul edeceğiz
```

```
double h=0.0001; //türev step büyüklüğü
int MAXIT=100;  //Maksimum iterasyon sayısı
int j;
double dfr,dx,dxold,fr,fh,fl;
double temp,b,a,r;
fl=fx(x1,Bi);
fh=fx(x2,Bi);
//System.out.println("x1 ="+x1+"fl="+fl+"x2="+x2+"fh="+fh);
//Bölgemizde kök değeri varmı kontrol et
if ((fl > 0.0 && fh > 0.0) || (fl < 0.0 && fh < 0.0))
System.out.println("n = "+n+"x1="+x1+"x2="+x2+"verilen sınırlar içinde kök değeri yok");
if (fl == 0.0) return x1;
if (fh == 0.0) return x2;
if (fl < 0.0)   { a=x1;b=x2;}
else           { b=x1;a=x2;}
r=0.5*(x1+x2); //orta nokta değeri
dxold=Math.abs(x2-x1);
dx=dxold;
fr=fx(r,Bi);   //fonksiyonun r noktasındaki değeri
dfr=dfx(r,h,Bi); //fonksiyonun türevinin r noktasındaki değeri
for (j=1;j<=MAXIT;j++)
{
if ((((r-b)*dfr-fr)*((r-a)*dfr-fr) > 0.0) || (Math.abs(2.0*fr) > Math.abs(dxold*dfr)))
    { //ikiye bölme
                dxold=dx;
     dx=0.5*(b-a);
     r=a+dx;
     if (a == r) return r; //çözüm!!!!
    }
else{ //Newton-Raphson
                dxold=dx;
     dx=fr/dfr;
     temp=r;
     r -= dx;
     if (temp == r) return r;  //çözüm!!!!!
    }
if (Math.abs(dx) < xacc) return r; //çözüm!!!!!
fr=fx(r,Bi);
dfr=dfx(r,h,Bi);
if (fr < 0.0)
a=r;
else
b=r;
}
System.out.println("Maksimum iterasyon sayısı aşıldı");
return 0.0; //program buraya gelmemeli
}

public static double teta(double Fo,double Bi,double x)
{double tt=0;double ln=0;
 for(int n=1;n<20;n++)
 {ln=lambda(n,Bi);tt+=Cn(ln,Bi)*Math.exp(-ln*ln*Fo)*Math.cos(ln*x);}
 return tt;
}

    public static void main (String args[])
    { fb f=new fb();
      double Bi,Fo,x;
      Fo=Double.parseDouble(JOptionPane.showInputDialog(" Forier sayısı Fo : "));
      Bi=Double.parseDouble(JOptionPane.showInputDialog(" Biot sayısı   Bi : "));
      x=Double.parseDouble(JOptionPane.showInputDialog("         x* = x/L : "));
      JOptionPane.showMessageDialog(null," teta = "+teta(Fo,Bi,x),
      "duvarda zamana bağlı ısı transferi  : ",JOptionPane.PLAIN_MESSAGE);
      System.exit(0);
    }
}
```

Bi	ζ_1(rad)	C1
0.01	0.0998	1.0017
0.02	0.141	1.0033
0.03	0.1732	1.0049
0.04	0.1987	1.0066
0.05	0.2217	1.0082
0.1	0.3111	1.016
0.5	0.6533	1.0701
1	0.8603	1.1191
5	1.3138	1.2402
10	1.4289	1.262
50	1.54	1.2727
100	1.5552	1.2731

silindir için benzer denklem silindirik koordinat sisteminde çözülürse sonuç

$$\bar{\theta} = \sum_{n=1}^{\infty} C_n e^{-\zeta_n^2 Fo} J_0(\zeta_n \bar{r})$$

$$C_n = \frac{2J_1(\zeta_n)}{\zeta_n J_0^2(\zeta_n) + J_1^2(\zeta_n)}$$

$$\zeta_n \frac{J_1(\zeta_n)}{J_0(\zeta_n)} = Bi$$

denklemleriyle verilebilir. Buradaki $J_0(\zeta_n)$, $J_1(\zeta_n)$ birinci tip Bessel fonksiyonlarıdır. Bu fonksiyonlar Mathd sınıfında tanımlanmıştır.

Program 2.9-3 Newton-Raphson-İkiye bölme kombine formülü, ile zamana bağlı silindir probleminin çözümü

```
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

public class HT_silindir
{
double x0,x1,x2;
int ix;

public HT_silindir()
{
ix=0;
x1=0.0;
}

public  static double fx(double lambda,double Bi)
{return lambda*Mathd.J(1,lambda)/Mathd.J(0,lambda)-Bi;}

public  static double dfx(double x,double h,double Bi)
{ //Türev hesabı
  return (-fx((x+2.0*h),Bi)+8.0*fx((x+h),Bi)-8.0*fx((x-h),Bi)+fx((x-2.0*h),Bi))/(12.0*h);
}

public  double Cn(double lambda)
{
 double J1=Mathd.J(1,lambda);
```

```
 double J0=Mathd.J(0,lambda);
 return 2.0/lambda*J1/(J0*J0+J1*J1);
}

public void genislet(double Bi)
{
        //kök olan bir bölge bulana kadar verilen sınırı genişletir
        x0=x1;
        x2=x1+0.00001;
        int NTRY=200;
        double a[]=new double[2];
        double FACTOR=1.01;
        int j;
        double f1,f2;
        if (x1 == x2) System.out.println("Giriş değerleri yanlış verildi");
        f1=fx(x1,Bi);
        f2=fx(x2,Bi);
        for (j=1;j<=NTRY;j++) {
                if (f1*f2 < 0.0) {ix++;break;}
                else
                {f2=fx(x2 += FACTOR*(x2-x1),Bi);}
        }
}
//  x1 ve x2 nin yeni değerleri geldi


public double lambda(double Bi)
{
//lineer olmayan denklemin kökü : ikiyebölme-Newton Raphson metodu
genislet(Bi);        //kök sınırlarını saptar
double xacc=1.0e-10; //iterasyonlar arasındaki fark, bu rakamdan
// küçük olduğunda çözüme ulaştığımızı kabul edeceğiz
double EPS=1.0e-10;
double h=0.0001; //türev step büyüklüğü
int MAXIT=100;  //Maksimum iterasyon sayısı
int j;
double dfr,dx,dxold,fr,fh,fl;
double temp,b,a,r;
fl=fx(x1,Bi);
fh=fx(x2,Bi);
//System.out.println("x1 ="+x1+"fl="+fl+"x2="+x2+"fh="+fh);
//Bölgemizde kök değeri varmı kontrol et
if ((fl > 0.0 && fh > 0.0) || (fl < 0.0 && fh < 0.0))
System.out.println("n = "+ix+"x1="+x0+"x2="+x2+"verilen sınırlar içinde kök değeri yok");
if (fl == 0.0) return x1;
if (fh == 0.0) return x2;
if (fl < 0.0)   { a=x1;b=x2;}
else          { b=x1;a=x2;}
r=0.5*(x1+x2); //orta nokta değeri
dxold=Math.abs(x2-x1);
dx=dxold;
fr=fx(r,Bi);   //fonksiyonun r noktasındaki değeri
dfr=dfx(r,h,Bi); //fonksiyonun türevinin r noktasındaki değeri
for (j=1;j<=MAXIT;j++)
{
if ((((r-b)*dfr-fr)*((r-a)*dfr-fr) > 0.0) || (Math.abs(2.0*fr) > Math.abs(dxold*dfr)))
   { //ikiye bölme
               dxold=dx;
    dx=0.5*(b-a);
    r=a+dx;
    if ( a == r && fr>EPS) {x1=r+0.001;return r;} //çözüm!!!!
   }
else{ //Newton-Raphson
               dxold=dx;
    dx=fr/dfr;
    temp=r;
```

```
      r -= dx;
      if ((temp == r) && fr<EPS) {x1=r+0.001;return r;}  //çözüm!!!!!
    }
if ((Math.abs(dx) < xacc)&& fr<EPS) {x1=r+0.001;return r;} //çözüm!!!!!
fr=fx(r,Bi);
dfr=dfx(r,h,Bi);
if (fr < 0.0)
a=r;
else
b=r;
}
System.out.println("Maksimum iterasyon sayısı aşıldı");
return 0.0; //program buraya gelmemeli
}

public  double teta(double Fo,double Bi,double x)
{double tt=0;double ln=0;
 for(int n=0;n<20;n++)
 {ln=lambda(Bi);
 double ffx=fx(ln,Bi);
 double cn;
  if(Math.abs(ffx)<1.0e-4)
  {
  cn=Cn(ln);
  System.out.println("ln ="+ln+"Cn = "+cn+"  ix="+ix+"  x1="+x0+"x2="+x2+"fx="+ffx);
  tt+=cn*Math.exp(-ln*ln*Fo)*Mathd.J(0,ln*x);
  }
 }
 return tt;
}
    public static void main (String args[])
    {
      HT_silindir hts=new  HT_silindir();
      double Bi,Fo,x;
      Fo=Double.parseDouble(JOptionPane.showInputDialog(" Forier sayısı Fo : "));
      Bi=Double.parseDouble(JOptionPane.showInputDialog(" Biot sayısı   Bi : "));
      x=Double.parseDouble(JOptionPane.showInputDialog("         x* = x/L : "));
      JOptionPane.showMessageDialog(null," teta = "+hts.teta(Fo,Bi,x),
      "silindirde zamana bağlı ısı transferi  : ",JOptionPane.PLAIN_MESSAGE);
    }
}
```

2.28 İkinci derece ters Lagrange interpolasyon yöntemi

Lagrange interpolasyonunu interpolasyon ve eğri uydurma bölümünde göreceğiz. İkinci dereceden Lagrange interpolasyonu x=f(y) şeklinde (ters interpolasyon olarak) yazılırsa

$$x=\frac{[y-f(x_1)][y-f(x_2)]x_3}{[f(x_3)-f(x_1)][f(x_3)-f(x_2)]}+\frac{[y-f(x_2)][y-f(x_3)]x_1}{[f(x_1)-f(x_2)][f(x_1)-f(x_3)]}+\frac{[y-f(x_3)][y-f(x_1)]x_2}{[f(x_2)-f(x_3)][f(x_2)-f(x_1)]}$$

bu intrepolasyon denkleminde y=0 alındığında x değeri bize y=f(x)=0 denkleminin köklerini verecektir.

$$x=\frac{f(x_1)f(x_2)x_3}{[f(x_3)-f(x_1)][f(x_3)-f(x_2)]}+\frac{f(x_2)f(x_3)x_1}{[f(x_1)-f(x_2)][f(x_1)-f(x_3)]}+\frac{f(x_3)f(x_1)x_2}{[f(x_2)-f(x_3)][f(x_2)-f(x_1)]}$$

$$x=\frac{[f(x_1)-f(x_2)]f(x_1)f(x_2)x_3+[f(x_3)-f(x_2)]f(x_2)f(x_3)x+[f(x_3)-f(x_1)]f(x_3)f(x_1)x_2}{[f(x_3)-f(x_1)][f(x_3)-f(x_2)][f(x_1)-f(x_3)]}$$

Denklemimiz hesap kolaylığı açısından

$$R = \frac{f(x_2)}{f(x_3)}$$

$$S = \frac{f(x_2)}{f(x_1)}$$

$$T = \frac{f(x_1)}{f(x_3)}$$

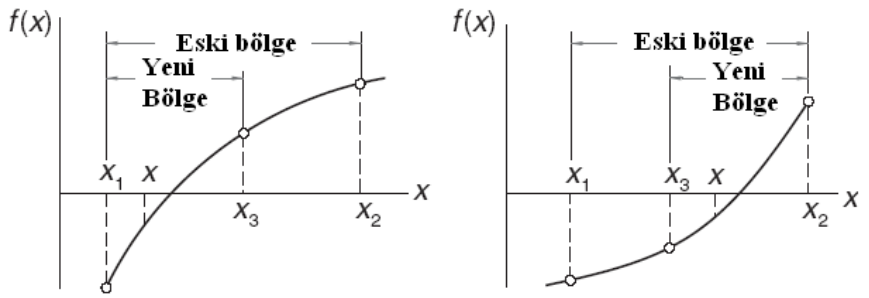
$P = S[T(R-T)(x_3 - x_2) - (1-R)(x_2 - x_1)]$

$Q = (T - 1)(R - 1)(S - 1)$

Formunda da yazılabilir. Kök değeri

$$x = x_2 + \frac{P}{Q}$$

olacaktır. Yeni kök bulunduktan sonra bulunan köke komşu olan bölgeler bırakılır ve bu bölgenin dışında olan bölge elimine edilir.



(bolum2_014.jpg,Resimaltı: İkinci derece ters lagrange interpolasyon yönteminde bölge seçimi)
Şekilde bu seçim işlemi görülmektedir. Bu metodun uygulaması Program 2.10.1 de verilmiştir.

Program 2.10.1 ikinci derece ters Lagrange interpolasyon formülü kullanarak kök bulma programı

```
 import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============= Tanımı gereken fonksiyon ================

class fc extends f_x
{
public double func(double x)
{return x*x-2;}
}
//============================================================

public class FSCO1a
{

public static double tersinter2Dkok(f_x f,double x1,double x2,double x3)
{
```

```
// brent metodu ile kök değerini bulur
double test;
double p=0;
double es,ea;
double f1,f2,f3,fp;
int maxit=500,iter=0;
double tol=1.0e-15;
es=0.0000001;
ea=1.1*es;
f1=f.func(x1);
f2=f.func(x2);
f3=f.func(x3);
if(f1==0) return x1;
else if(f2==0) return x2;
else if(f3==0) return x3;
if(f1*f3>0) System.out.println("verilen bölgede kök yok");
while((ea>es)&&(iter<maxit))
{               p=-(f2*f3*x1*(f2-f3)+f3*f1*x2*(f3-f1)+f1*f2*x3*(f1-f2))/((f1-f2)*(f2-f3)*(f3-f1));
   fp=f.func(p);
   ea=Math.abs(x3-p);
               if(Math.abs(f3)<tol) return x3;
   if(p>x3) {x1=x3;f1=f3;x3=p;f3=fp;}
   else if(p<x3) {x2=x3;f2=f3;x3=p;f3=fp;}
   iter++;
}
if(iter>=maxit) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return p;
}

    public static void main (String args[])
    {
                fc f=new fc();
    double x1,x2,x3;
    x1=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun alt sınırını giriniz x1 : "));
    x2=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun ortaki bir değerini giriniz x3 : "));
    x3=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun üst sınırını giriniz x2 : "));
    double r;
    r= tersinter2Dkok(f,x1,x2,x3);
    String s1=" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r);
    String s2="ikinci derece ters interpolasyon yöntemiyle kök hesabı : \n";
    System.out.println(s2+s1);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO1a
ikinci derece ters interpolasyon yöntemiyle kök hesabı :
 kök değeri  : 1.4142135623730954
Fonksiyon değeri : 8.881784197001252E-16
> Terminated with exit code 0.
```

Program 2.10.1 ikinci derece ters Lagrange interpolasyon formülü kullanarak kök bulma programı, matlab versiyonu

```
function p =tersLagrangeKok(f,x1,x2,x3)
%2. derece ters  Lagrange interpolasyon formülü ile kök bulma
p=1;
maxit=100;iter=0;
tol=1.0e-10;
es=0.00000001;
ea=0.001;
f1=f(x1);
f2=f(x2);
f3=f(x3);
```

```
if f1==0 p=x1;
elseif f2==0 p=x2;
elseif f3==0 p=x3;end
if f1*f3>0 fprintf('verilen bölgede kök mevcut değildir');end
while (ea>es) && (iter<maxit) && f1~=0 && f2~=0 && f3~=0
   p=-(f2*f3*x1*(f2-f3)+f3*f1*x2*(f3-f1)+f1*f2*x3*(f1-f2))/((f1-f2)*(f2-f3)*(f3-f1));
   fp=f(p);
   ea=abs(x3-p);
   if abs(f3)<tol  p=x3; end
   if p>x3  x1=x3;f1=f3;x3=p;f3=fp;
   elseif(p<x3) x2=x3;f2=f3;x3=p;f3=fp;end
   iter=iter+1;
   end
if(i>=maxit) fprintf('Maksimum döngü sayısı aşıldı sonuç geçersiz olabilir');end
end
```

```
>> f1=@(x) x*x-2;
>> tersLagrangeKok(f1,0,2,3)
ans =
  1.414213562373314
```

2.29 Brent kök bulma yöntemi

Yukarda Newton-Raphson metodu ile ikiye bölme metodunun integrasyonunun çok daha verimli ve emniyetli bir metod oluşturduğundan bahsetmiştik. Aynı şekilde ikinci dereceden ters Lagrange interpolasyon kök bulma yöntemi de ikiye bölme yöntemi ve kiriş yöntemi ile integre edilerek oldukça yararlı bir yöntem oluşturulabilir. Bu yöntem Brent yöntemi adını alır. Bu yöntemde ikinci dereceden ters Lagrange interpolasyon kök bulma yönteminden farklı olarak sadece iki sınır değeri tanımlanır. Birinci step olarak ikiye bölme formülü uygulanır ve ikinci dereceden interpolasyonun üçüncü noktası olarak bu noktadan yararlanılır. İkinci dereceden ters Lagrange interpolasyon kök formülü kullanılarak yeni bir çözüm noktası saptanır ve bu çözüm bölgesine göre bölge küçültülür. Eğer çözüm seçilen değer $x_1 - x_2$ bölgesine düşmüyorsa yeni bir ikiye bölme stebi uygulanır. Denklemin bölündüğü değerinin çok küçük olduğu sıfıra yaklaştığı değerlerde metod yine yanlış değer verebilir. Bu durumda yine ikiye bölme yöntemine dönülür. Bu bölümde Brent yönteminin iki versiyonu örnek program olarak düzenlenmiştir. Birinci program üstteki bölümde oluşturduğumuz programı değiştirerek oluşturulmuştur.

Program 2.11.1 Brent yöntemi ile kök bulma

```
 import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============== Tanımı gereken fonksiyon =================

class fc extends f_x
{
public double func(double x)
{return x*x-2;}

}
//============================================================

public class FSCO1B
{

public static double brent(f_x f,double a,double b)
{
```

```
// brent metodu ile kök değerini bulur
double test;
double p=0;
double es,ea;
double f1,f2,f3,fp;
int maxit=500,iter=0;
double tol=1.0e-15;
es=0.0000001;
double x1=a;f1=f.func(x1);
double x2=b;f2=f.func(x2);
double x3=(x1+x2)/2.0;f3=f.func(x3);
if(f1==0) return x1;
else if(f2==0) return x2;
else if(f3==0) return x3;
if(f1*f2>0) System.out.println("verilen bölgede kök yok");
p=-(f2*f3*x1*(f2-f3)+f3*f1*x2*(f3-f1)+f1*f2*x3*(f1-f2))/((f1-f2)*(f2-f3)*(f3-f1));
fp=f.func(p);
ea=Math.abs(x3-p);
while((ea>es)&&(iter<maxit))
{           if(Math.abs(f3)<tol) return x3;
   if((p<x1) && (p>x2))
   {p=(x1+x2)/2.0;
    if(p>x3) {x1=x3;f1=f3;x3=p;f3=fp;}
    else if(p<x3) {x2=x3;f2=f3;x3=p;f3=fp;}
   }
   else
   {
   if(p>x3) {x1=x3;f1=f3;x3=p;f3=fp;}
   else if(p<x3) {x2=x3;f2=f3;x3=p;f3=fp;}
   p=-(f2*f3*x1*(f2-f3)+f3*f1*x2*(f3-f1)+f1*f2*x3*(f1-f2))/((f1-f2)*(f2-f3)*(f3-f1));
   fp=f.func(p);
   ea=Math.abs(x3-p);
   }
   iter++;
}
if(iter>=maxit) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  "          çözüm          geçerli          olmıyabilir","MAKSİMUM          ITERASYON          SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return p;
}

    public static void main (String args[])
    {
            fc f=new fc();
    double x1,x2,x3;
    x1=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun alt sınırını giriniz x1 : "));
    x2=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun ortaki bir değerini giriniz x3 : "));
    double r;
    r= brent(f,x1,x2);
    String s1=" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r);
    String s2="Brent yöntemiyle kök hesabı : \n";
    System.out.println(s2+s1);
```

```
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO1B
Brent yöntemiyle kök hesabı :
 kök değeri  : 1.414213562373095
Fonksiyon değeri : -4.440892098500626E-16
> Terminated with exit code 0.
```

Program 2.11.2 Brent yöntemi ile kök bulma (ikinci versiyon)

```
 ///Bilgisayar programlama II ve Sayısal Çözümleme
//
//
// Brent yöntemi ile kök bulmak
//
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
class fc extends f_x
{
public double func(double x)
{return x*x-2;}

}

public class SCO7
{
public static double brent(f_x f,double x1,double x2)
{
   //lineer olmıyan denklemin kökü : brent metodu
            //referans : Numerical Recipes in C, second edition, William H. Press,
   //Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery
   //cambridge university press
            double ITMAX=100;
   double EPS=3.0e-8;
            int iter;
            double a=x1,b=x2,c,d,e,min1,min2;
            double tol=1.0e-5;
            c=0;
            d=0;
            e=0;
            double fa=f.func(a),fb=f.func(b),fc,p,q,r,s,tol1,xm;
            if (fb*fa > 0.0) JOptionPane.showMessageDialog(null,"Kök sınırları doğru olarak seçilmemiş \n sonuç hatalı olabilir"
            ,"KÖK SINIRLARI SEÇİM UYARISI",JOptionPane.WARNING_MESSAGE);
            fc=fb;
            for (iter=1;iter<=ITMAX;iter++) {
                        if (fb*fc > 0.0) {
                                    c=a;
                                    fc=fa;
                                    e=d=b-a;
                        }
                        if (Math.abs(fc) < Math.abs(fb)) {
                                    a=b;
                                    b=c;
                                    c=a;
                                    fa=fb;
                                    fb=fc;
                                    fc=fa;
                        }
                        tol1=2.0*EPS*Math.abs(b)+0.5*tol;
```

```
                        xm=0.5*(c-b);
                        if (Math.abs(xm) <= tol1 || fb == 0.0) return b;
                        if (Math.abs(e) >= tol1 && Math.abs(fa) > Math.abs(fb)) {
                                s=fb/fa;
                                if (a == c) {
                                        p=2.0*xm*s;
                                        q=1.0-s;
                                } else {
                                        q=fa/fc;
                                        r=fb/fc;
                                        p=s*(2.0*xm*q*(q-r)-(b-a)*(r-1.0));
                                        q=(q-1.0)*(r-1.0)*(s-1.0);
                                }
                                if (p > 0.0)  q = -q;
                                p=Math.abs(p);
                                min1=3.0*xm*q-Math.abs(tol1*q);
                                min2=Math.abs(e*q);
                                if (2.0*p < (min1 < min2 ? min1 : min2)) {
                                        e=d;
                                        d=p/q;
                                } else {
                                        d=xm;
                                        e=d;
                                }
                        } else {
                                d=xm;
                                e=d;
                        }
                        a=b;
                        fa=fb;
                        if (Math.abs(d) > tol1)
                                b += d;
                        else
                                b += (xm > 0.0 ? Math.abs(tol1) : -Math.abs(tol1));
                        fb=f.func(b);
                }
                JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return b;
}
    public static void main (String args[])
    {
    double a,b;
    fc f=new fc();
    a=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun alt sınırını giriniz a : "));
    b=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun üst sınırını giriniz b : "));
    double r;
    r= brent(f,a,b);
    String s1=" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r);
    String s2="Brent yöntemiyle kök hesabı : \n";
    System.out.println(s2+s1);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO7
Brent yöntemiyle kök hesabı :
 kök değeri  : 1.4142135624137586
Fonksiyon değeri : 1.1501377628064802E-10
> Terminated with exit code 0.
```

2.12 Kök sınırlarının saptanması

Bir değişkenli fonksiyonlar için (a,b) bölgesinde en az bir kök vardır diyebilmemiz için uç noktadaki fonksiyon değerlerinin ters işaretli olması gerekir. Bu kural üniversal olarak her tür fonksiyon için geçerli olmasa da sürekli fonsiyon ailesinin çoğu üyesi için geçerlidir. O zaman verilen bir alanı daha küçük steplerle tarayarak kök sınırlarının bulunduğu bölgeyi daraltabiliriz, veya belli bir noktadan veya birbirine yakın iki noktadan başlayarak ve her seferinde bölgeyi birden büyük bir α katsayısıyla kökün bulunduğunu tahmin ettiğimiz tarafa doğru genişleterek kökün olduğu bir bölge sapayabiliriz. Bu amaçla **genişlet** ve **koksınırınısapta** metodları geliştirilmiş ve alttaki programda brent metoduyla birlikte kullanılmıştır.

Program 2.12.1 Kök sınırlarını saptayan ve brent metoduyla kökü bulan FSCO6.java programı

```
 ///Bilgisayar programlama II ve Sayısal Çözümleme
//
// Brent yöntemi ile kök bulmak
// ve kökün sınırlarını saptamak
// Dr. M. Turhan Çoban
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
class fc extends f_x
{
public double func(double x)
{return x*x*3+2*x-3;}

}

public class SCO6
{

public static double[] genislet(f_x f,double x1)
{return genislet(f,x1,(x1+0.1));}

public static double[] genislet(f_x f,double x1,double x2)
{
	//kök olan bir bölge bulana kadar verilen sınırı genişletir
	int NTRY=50;
	double a[]=new double[2];
	double FACTOR=1.6;
	int j;
	double f1,f2;
	if (x1 == x2) System.out.println("Bad initial range in ZBRAC");
	f1=f.func(x1);
	f2=f.func(x2);
	for (j=1;j<=NTRY;j++) {
		if (f1*f2 < 0.0) {a[0]=x1;a[1]=x2;return a;}
		if (Math.abs(f1) < Math.abs(f2))
			f1=f.func(x1 += FACTOR*(x1-x2));
		else
			f2=f.func(x2 += FACTOR*(x2-x1));
	}
	{a[0]=x1;a[1]=x2;return a;}
}
public static double[][] koksinirisapta(f_x f,double x1,double x2,int n,int nbb)
{
	// koklerin yer aldığı alt bölgeleri saptar
	// n : verilen bölgeyi böldüğümüz alt bölge sayısı
	// x1,x2 : sınır değerleri
	// nbb = aranan bölgedeki köksayısı
	//
	int nb;
	int i;
```

```
		double x,fp,fc,dx;
		double xb[][]=new double[2][nbb];
		nb=0;
		dx=(x2-x1)/n;
		x=x1;
		fp=f.func(x1);
		for (i=1;i<=n;i++)
		{
			x+=dx;
			fc=f.func(x);
			// eğer kök olan bölge bulunduysa.....
			if (fc*fp < 0.0 || fp==0) {
				xb[0][nb]=x-dx;
				xb[1][nb]=x;
				nb++;
			}
			fp=fc;
			if (nbb == nb) return xb;
		}
if( nb == 0)
		System.out.println("arama tamamlandı kök olan bölge bulunamadı ");
else if(nb<nbb)
  {		System.out.println("arama tamamlandı sadece "+nb+" adet kök bulundu \n"+
    "siz "+nbb+" adet kök için arama yaptırdınız");
   double xc[][]=new double[2][nb];
   for (i=0;i<nb;i++) {xc[0][i]=xb[0][i];xc[1][i]=xb[1][i];}
   return xc;
  }
return xb;
}

public static double brent(f_x f, double x1,double x2)
{
    //lineer olmayan denklemin kökü : brent metodu
		//referans : Numerical Recipes in C, second edition, William H. Press,
    //Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery
    //cambridge university press
		double ITMAX=200;
    double EPS=3.0e-10;
		int iter;
		double a=x1,b=x2,c,d,e,min1,min2;
		double tol=1.0e-5;
		c=0;
		d=0;
		e=0;
		double fa=f.func(a),fb=f.func(b),fc,p,q,r,s,tol1,xm;
		if (fb*fa > 0.0) System.out.println("Kök sınırları doğru olarak seçilmemiş \n sonuç hatalı olabilir");
		fc=fb;
		for (iter=1;iter<=ITMAX;iter++) {
			if (fb*fc > 0.0) {
				c=a;
				fc=fa;
				e=d=b-a;
			}
			if (Math.abs(fc) < Math.abs(fb)) {
				a=b;
				b=c;
				c=a;
				fa=fb;
				fb=fc;
				fc=fa;
			}
			tol1=2.0*EPS*Math.abs(b)+0.5*tol;
			xm=0.5*(c-b);
			if (Math.abs(xm) <= tol1 || fb == 0.0) return b;
			if (Math.abs(e) >= tol1 && Math.abs(fa) > Math.abs(fb)) {
```

```
                                        s=fb/fa;
                                        if (a == c) {
                                                p=2.0*xm*s;
                                                q=1.0-s;
                                        } else {
                                                q=fa/fc;
                                                r=fb/fc;
                                                p=s*(2.0*xm*q*(q-r)-(b-a)*(r-1.0));
                                                q=(q-1.0)*(r-1.0)*(s-1.0);
                                        }
                                        if (p > 0.0)  q = -q;
                                        p=Math.abs(p);
                                        min1=3.0*xm*q-Math.abs(tol1*q);
                                        min2=Math.abs(e*q);
                                        if (2.0*p < (min1 < min2 ? min1 : min2)) {
                                                e=d;
                                                d=p/q;
                                        } else {
                                                d=xm;
                                                e=d;
                                        }
                                } else {
                                        d=xm;
                                        e=d;
                                }
                                a=b;
                                fa=fb;
                                if (Math.abs(d) > tol1)
                                        b += d;
                                else
                                        b += (xm > 0.0 ? Math.abs(tol1) : -Math.abs(tol1));
                                fb=f.func(b);
                        }
                        System.out.println("Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir");
return b;
}

    public static void main (String args[])
    {
    double a[],x0,r;
    double dx=0.1;
     fc f=new fc();
    a=genislet(f,Double.parseDouble(JOptionPane.showInputDialog(" ilk tahmin değeri x0 : ")));
    r= brent(f,a[0],a[1]);
    String s1=" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r)+"\nsınırlar: a="+a[0]+" b="+a[1];
    String s2="brent yöntemiyle kök hesabı : \n";
    System.out.println(s2+s1);
    double b[][]=koksinirisapta(f,0.0,10.0,10,4);
    System.out.println(Matrix.toString(b));
    b=koksinirisapta(f,b[0][0],b[1][0],10,4);
    System.out.println(Matrix.toString(b));
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO6
brent yöntemiyle kök hesabı :
 kök değeri  : 0.7207591392632257
Fonksiyon değeri : -5.109791505475414E-7
sınırlar: a=0.4239999999999995 b=1.1
arama tamamlandı sadece 1 adet kök bulundu
siz 4 adet kök için arama yaptırdınız
       0.000000000000000
       1.000000000000000
```

```
arama tamamlandı sadece 1 adet kök bulundu
siz 4 adet kök için arama yaptırdınız
      0.700000000000000
      0.800000000000000

> Terminated with exit code 0.
```

2.13 Ridder metodu

Daha önce Yer değiştirme (False position – Regula-falsi) metodunu incelemiştik. Ridder metodu bu metodun değiştirilmiş bir versiyonu olarak kabul edilebilir. x_1 ve x_2 noktaları arasında bir kök mevcut ise Ridder metodu önce orta noktada fonksiyonu değerlendirir.

$x_3=(x_1+x_2)/2$

daha sonra fonksiyonu lineer fonksiyona dönüştüren özel bir faktörü e^Q faktörünü çözer. Daha açık yazacak olur isek $f(x_1)-2f(x_3)\,e^Q + f(x_2)\,e^{2Q} = 0$ fonksiyonundan e^Q faktörünü çözer.

İkinci dereceden bir fonksiyon olan bu fonksiyonun kökleri

$$e^Q = \frac{f(x_3) + sign[f(x_2)]\sqrt{f(x_3)^2 - f(x_1)f(x_2)}}{f(x_2)}$$

şeklinde hesaplanır. Buradaki sign(x) fonksiyonu x in işaretini verecektir. Eğer x sıfırdan büyükse +1, eğer sıfırdan küçükse -1 değerini verecektir. Şimdi yer değiştirme metodu $f(x_1)$, $f(x_3)$ ve $f(x_2)$ fonksiyonlarını kullanarak değil $f(x_1)$, $f(x_3)\,e^Q$ ve $f(x_2)\,e^{2Q}$ fonksiyonlarına uygulanarak x_4 kök değeri bulunur.

$$x_4 = x_3 + (x_3 - x_1)\frac{sign[f(x_1) - f(x_2)]f(x_3)}{\sqrt{f(x_3)^2 - f(x_1)f(x_2)}}$$

Denklem ikinci dereceden olduğundan ikinci derece yaklaşım sağlar. Aynı zamanda bulunan kök her zaman x_1 ve x_2 arasında yer alacaktır.

Program 2.13.1 Ridder metoduyla kökü bulan FSCO7.java programı

```
 ///Bilgisayar programlama II ve Sayısal Çözümleme
//ÖDEV 8
// f(x)=ln(x^2)-0.7 fonksiyonunun x0=0 x1=4 ilktahminini kulanarak
// ridder  yöntemi ile kökünü bulmak için bir
// program geliştiriniz, kök değerini hesaplatınız

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============ Tanımı gereken fonksiyon ===============

class fb extends f_x
{
  public double func (double x)
  { return Math.log(x*x)-0.7;}
}

public class FSCO8
{
// Ridder Metodu ile lineer olmıyan denklemlerin köklerinin bulunması
//referans : Numerical Recipes in C, second edition, William H. Press,
//Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery
//cambridge university press

public static double SIGN(double a,double b)
{
return Math.abs(a)*b/Math.abs(b);
}
```

```
public static double ridder(f_x f, double x1, double x2)
{double xacc=1.0e-9;
 return ridder(f, x1,x2,xacc);
}

public static double ridder(f_x f, double x1, double x2, double xacc)
{
        int NRANSI;
        int MAXIT=64;
        double UNUSED=-1.11e30;
        int j;
        double ans,fh,fl,fm,fnew,s,xh,xl,xm,xnew;

        fl=f.func(x1);
        fh=f.func(x2);
        if ((fl > 0.0 && fh < 0.0) || (fl < 0.0 && fh > 0.0)) {
                xl=x1;
                xh=x2;
                ans=UNUSED;
                for (j=0;j<MAXIT;j++) {
                        xm=0.5*(xl+xh);
                        fm=f.func(xm);
                        s=Math.sqrt(fm*fm-fl*fh);
                        if (s == 0.0) return ans;
                        xnew=xm+(xm-xl)*((fl >= fh ? 1.0 : -1.0)*fm/s);
                        if (Math.abs(xnew-ans) <= xacc) return ans;
                        ans=xnew;
                        fnew=f.func(ans);
                        if (fnew == 0.0) return ans;
                        if (SIGN(fm,fnew) != fm) {
                                xl=xm;
                                fl=fm;
                                xh=ans;
                                fh=fnew;
                        } else if (SIGN(fl,fnew) != fl) {
                                xh=ans;
                                fh=fnew;
                        } else if (SIGN(fh,fnew) != fh) {
                                xl=ans;
                                fl=fnew;
                        } else System.out.println("bu noktaya asla ulaşamayız.");
                        if (Math.abs(xh-xl) <= xacc) return ans;
                }
                System.out.println("ridder metodu maksimum iterasyon sayısını aştı");
        }
        else {
                if (fl == 0.0) return x1;
                if (fh == 0.0) return x2;
                System.out.println("ridderde verilen sınırlar arasında kök mevcut değil");
        }
        return 0.0;
}
    public static void main (String args[])
    {
             fb f=new fb();
    double a,b;
    a=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun alt sınırını giriniz a : "));
    b=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun üst sınırını giriniz b : "));
    double r;
    r= ridder(f,a,b);
    String s1=" kök değeri  : "+r+"\nFonksiyon değeri : "+f.func(r);
    String s2="Ridder yöntemiyle kök hesabı : \n";
    System.out.println(s2+s1);
    System.exit(0);
    }
```

```
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO8
Ridder yöntemiyle kök hesabı :
 kök değeri  : 1.4190675485932573
Fonksiyon değeri : 1.1102230246251565E-16
> Terminated with exit code 0.
```

2.14 Üçüncü dereceden polinomun kökleri

İkinci dereceden polinomun köklerini bulma denklemini hepimiz biliriz. Üçüncü dereceden polinomun köklerini paralel, fakat biraz daha karışık olan bir denklem sistemi kullanarak bulabiliriz. Sınır değer problemi çözümünde ninci dereceden polinomun köklerini bulmayı da irdeleyeceğiz. Kübik polinomun köklerini temel olarak

$f(x) = d_3 x^3+d_2 x^2+d_1 x +d_0=0$

$$a=\frac{d_2}{d_3} \quad b=\frac{d_1}{d_3} \quad c=\frac{d_0}{d_3}$$

$f(x) = x^3+a x^2+b x +c = 0$

$$Q=\frac{a^2-3b}{9} \quad y_1=2a^3 \; y_2=-9ab \quad y_3=27c \quad y_4=y_1+y_2+y_3$$

$$R=\frac{y_4}{54} \quad \theta=\cos^{-1}\left(\frac{R}{\sqrt{Q^3}}\right)$$

Eğer $(R^2<Q^3)$ üç gerçek kök mevcut

$$x_0=(-2\sqrt{Q})\cos\left(\frac{\theta}{3}\right)-\frac{a}{3}$$

$$x_1=(-2\sqrt{Q})\cos\left(\frac{\theta-2\pi}{3}\right)-\frac{a}{3}$$

$$x_2=(-2\sqrt{Q})\cos\left(\frac{\theta+2\pi}{3}\right)-\frac{a}{3}$$

Eğer $(R^2\geq Q^3)$

$$A=\left(R+\sqrt{R^2-Q^3}\right)^{1/3}$$

Eğer A=0 $\Rightarrow$B=0

Eğer A< >0 $\Rightarrow B=\frac{Q}{A}$

$$x_0=(A+B)-\frac{a}{3}$$ gerçek kök

$$x_1=\left[-\left(\frac{A+B}{2}\right)-\frac{a}{3}\right]+\left[\frac{\sqrt{3}(A-B)}{2}\right]i$$ kompleks kök

$$x_2=\left[-\left(\frac{A+B}{2}\right)-\frac{a}{3}\right]-\left[\frac{\sqrt{3}(A-B)}{2}\right]i$$ kompleks kök

Şeklinde hesaplayabiliriz. Formüller biraz uzun görünse de bilgisayara uyarlarlama açısından oldukça basittir.

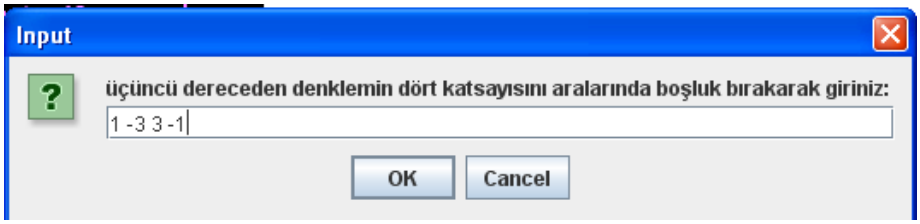
Program 2.14.1 Üçüncü dereceden polinomun köklerini bulan program

```
 import java.util.*;
import java.awt.*;
```

```
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.text.*;
import java.util.Locale;
//import Numeric;

class SCO13H
{
public static double[][] cubic_roots(double d[])
{
// roots of a degree 3 polynomial
// P(x)=d[3]*x^3+d[2]*x^2+d[1]*x+d[0]=0;
// assuming d is all real;
// if value of d will be entered bigger than d[3]
// remaining terms will be ignored
double x[][]=new double[2][3];
double a,b,c;
a=d[2]/d[3];
b=d[1]/d[3];
c=d[0]/d[3];
//System.out.println("a="+a+"b="+b+"c="+c);
double Q,R,theta;
Q=(a*a-3.0*b)/9.0;
double x1,x2,x3,x4;
x1=2.0*a*a*a;
x2=-9*a*b;
x3=27*c;
x4=x1+x2+x3;
R=x4/54.0;
//System.out.println("Q="+Q+"R="+R+"x1="+x1+"x2="+x2+"x3="+x3+"x4="+x4);
double Q3=Q*Q*Q;
double R2=R*R;
double qq;
//System.out.println("Q3="+Q3+"R2="+R2+(R2<Q3));
double A,B;
if(R2<Q3)
{
qq=-2.0*Math.sqrt(Q);
theta=Math.acos(R/Math.sqrt(Q3));
x[0][0]=qq*Math.cos(theta/3.0)-a/3.0;
x[0][1]=qq*Math.cos((theta-2.0*Math.PI)/3.0)-a/3.0;
x[0][2]=qq*Math.cos((theta+2.0*Math.PI)/3.0)-a/3.0;
}
else
{
A=-Math.pow((R+Math.sqrt(R2-Q3)),(1.0/3.0));
if(A==0) B=0;
else     B=Q/A;
x[0][0]=(A+B)-a/3.0;
x[0][1]=-0.5*(A+B)-a/3;
```

```
x[1][1]=Math.sqrt(3.0)/2.0*(A-B);
x[0][2]=-0.5*(A+B)-a/3;
x[1][2]=-Math.sqrt(3.0)/2.0*(A-B);
}
return x;
}


public static complex[] cubic_rootsC(double c[])
{
// roots of a degree 3 polynomial
//return 3 complex roots
double a[][]=new double[2][3];
a=cubic_roots(c);
complex e[]=new complex[3];
for(int i=0;i<3;i++)
  e[i]=new complex(a[0][i],a[1][i]);
return e;
}
     public static double[] verigir(String s)
     {
     String s1=JOptionPane.showInputDialog(s);
     StringTokenizer token=new StringTokenizer(s1);
     int n=token.countTokens()-1;
     int m=n+1;
     double a[]=new double[m];
     int j=0;
     while(token.hasMoreTokens())
     {
     Double ax=new Double(token.nextToken());
     a[j++]=ax.doubleValue();
     }
     return a;
     }

  public static void main(String args[]) throws IOException
  {
  String s="üçüncü dereceden denklemin dört katsayısını aralarında boşluk bırakarak giriniz: ";
  double [] x0=verigir(s);
  complex z[]=cubic_rootsC(x0);
  String s1="";
  for(int i=0;i<3;i++)
    {s1+=complex.toString(z[i])+"\n";}
  System.out.println("üçüncü dereceden denklemin kökleri : \n"+s1);
  }
}
```

Bu programda kullanılan kompleks değişken hesabını yapan complex sınıfı da program 2.4.5 de verilmiştir.



(bolum2_015.jpg,Resimaltı:)

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO13H
üçüncü dereceden denklemin kökleri :
(   1.0000000000 +    0.0000000000i )
(   1.0000000000 +   -0.0000000000i )
(   1.0000000000 +    0.0000000000i )

> Terminated with exit code 0.
```

2.15 Le Guerre metodu ile polinomların kompleks köklerini bulma

Leguerre iteratif metodu n inci dereceden polinomun köklerini bulmak için aşağıdaki denklemleri kullanır.

$$g(x) = [f'(x)]^2 - \frac{n f(x) f''(x)}{n-1}$$

$$x_{k+1} = x_k - \frac{n f(x_k)}{f'(x_k) \mp (n-1)\sqrt{g(x_k)}}$$

buradaki $\sqrt{g(x_k)}$ terimi kompleks olabilir. Dolayısıyla bu metodu kompleks olarak yazmak daha doğrudur.

Program 2.15-1 Leguerre yöntemi ile polinom köklerinin hesaplanması

```
 //
//
// kompleks Lequerre yöntemi ile polinomun kökünü bulmak
//
//

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

public class SCO4H
{
  static complex func(double a[],complex x)
  {  //complex polynomial
  // y(x)=a[n]*x^n+a[n-1]*x^(n-1)+..+a[0]
  int n=a.length;
  complex y=new complex(a[n-1]);
  for(int i=n-2;i>=0;i--)
  {y=complex.add(complex.multiply(y,x),a[i]);}
  return y;
```

```
 }

 static complex dfunc(double a[],complex x)
 {//complex numeric derivative of a polynomial
  double hh=0.001;
  complex h=new complex(hh,0.0);
  complex he=new complex(-hh,0.0);
  complex h2=new complex(2.0*hh,0.0);
  complex he2=new complex(-2.0*hh,0.0);
  complex xah=complex.add(x,h);
  complex xa2h=complex.add(x,h2);
  complex xeh=complex.add(x,he);
  complex xe2h=complex.add(x,he2);
  complex fe2h=complex.multiply(func(a,xe2h),(1.0/12.0));
  complex feh=complex.multiply(func(a,xeh),(-8.0/12.0));
  complex fah=complex.multiply(func(a,xah),(8.0/12.0));
  complex fa2h=complex.multiply(func(a,xa2h),(-1.0/12.0));
  complex df=complex.add(fe2h,complex.add(feh,complex.add(fah,fa2h)));
  df=complex.divide(df,h);
  return df;
 }

 static complex d2func(double a[],complex x)
 {//complex second numeric derivative of a polynomial
  double hh=0.001;
  complex h=new complex(hh,hh);
  complex hkare=complex.pow(h,2);
  complex he=new complex(-hh,-hh);
  complex h2=new complex(2.0*hh,2.0*hh);
  complex he2=new complex(-2.0*hh,-2.0*hh);
  complex xah=complex.add(x,h);
  complex xa2h=complex.add(x,h2);
  complex xeh=complex.add(x,he);
  complex xe2h=complex.add(x,he2);
  complex fe2h=complex.multiply(func(a,xe2h),(-1.0/12.0));
  complex feh=complex.multiply(func(a,xeh),(4.0/3.0));
  complex ff=complex.multiply(func(a,x),(-5.0/2.0));
  complex fah=complex.multiply(func(a,xah),(4.0/3.0));
  complex fa2h=complex.multiply(func(a,xa2h),(-1.0/12.0));
  complex d2f=complex.add(fe2h,complex.add(feh,complex.add(ff,complex.add(fah,fa2h))));
  d2f=complex.divide(d2f,hkare);
  return d2f;
 }


public static complex Leguerre(double a[],complex x)
{
int n=a.length-1;
double eps=1.0e-8;
int m=50;
int k=0;
complex f0=func(a,x);
```

```
complex f1=new complex();
complex f2=new complex();
complex g=new complex();
complex d1=new complex();
complex d2=new complex();
complex x0=x;
complex x1=new complex((-a[0]/a[1]),0.0);
do{
  f1=dfunc(a,x0);
  f2=d2func(a,x0);
  g=complex.add(complex.multiply(f1,f1),complex.multiply(complex.multiply(f0,f2),(-(double)n/(double)(n-
1))));
  d1=complex.add(f1,complex.multiply(complex.pow(g,0.5),(double)(n-1)));
  d2=complex.add(f1,complex.multiply(complex.pow(g,0.5),(double)(1-n)));
if(complex.abs(d1)>=complex.abs(d2))
x1=complex.add(x0,complex.multiply(complex.divide(f0,d1),(double)(-n)));
else                    x1=complex.add(x0,complex.multiply(complex.divide(f0,d2),(double)(-n)));
  x0=complex.add(x1,0.0);
  f0=func(a,x0);
  k++;
}while(complex.abs(f0)>eps && k<m);

if(complex.abs(f0)<eps) { return x1;}
JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x1;
}


    public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }

    public static void main (String args[])
    {
            //Polynom katsayılar olarak tanımlansı
            double a[]={-75,-5,4,-4,-1,1};
            int n=a.length-1;
    double x0[]=verigir(" Fonksiyon değeri için ilk tahmin : ");
```

```
    complex x=new complex(x0[0],x0[1]);
    complex r=Leguerre(a,x);
    complex ff=func(a,r);
    String s="polinom y(x)=a[n]*x^n+a[n-1]*x^(n-1)+..+a[0]\nkatsayılar : \n";
    s+=Matrix.toString(a);
    s+="kök = "+r.toString()+"\n"+"Fonksiyon = "+ff.toString();
    System.out.println("Leguerre yöntemiyle kök hesabı : \n"+s);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO4H
Leguerre yöntemiyle kök hesabı :
polinom y(x)=a[n]*x^n+a[n-1]*x^(n-1)+..+a[0]
katsayılar :
 -75.000000000000000000000000       -5.000000000000000000000000
   4.000000000000000000000000       -4.000000000000000000000000
  -1.000000000000000000000000        1.000000000000000000000000
kök = (  -2.0000000000 - i )
Fonksiyon = (   0.0000000000 +    0.0000000000i )

> Terminated with exit code 0.
```

Programda kullanılan complex sınıfı **Program 2.4.5** olarak tanımlanmıştır. Bu bölüme kadar lineer olmıyan bir bilinmiyenli f(x) denkleminin çeşitli çözüm yöntemlerini irdeledik. Eğer birden fazla bilinmiyen varsa, bu bilinmeyenlerin köklerini bulmak için kök sayısı kadar denkleme ihtiyacımız vardır. Bu durumda lineer olmıyan bir denklem sisteminin çözmekten veya köklerini bulmaktan bahsedebiliriz. Lineer olmıyan denklem sisteminin köklerini bulmak için lineer denklem sisteminin köklerini bulmayı öğrenmemiz gerektiğinden bu işlemi sonraki bölümlerimizde göreceğiz.

2.16 Horner metoduyla polinomların kompleks köklerinin bulunması

Horner metodu sentetik bölme olarak adlandırdığımız temel bir matematik işlemi kullanarak polinomların kökünü bulmaya yarayan bi methoddur. Sentetik bölme tekrarlı bir proses olarak tanımlanır. Genel tanım olarak eğer $P(x) = f_n(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$ polinomu verilmişse ve $(x - x_0)$ ile bölünmesi

$Q(x) = f_{n-1}(x) = b_1 + b_2x + b_3x^2 + ... + a_{n-1}x^{n-1}$ polinomunu veriyorsa

$f_n(x) = b_0 + (x - x_0) f_{n-1}(x)$ veya $P(x) = b_0 + (x - x_0)Q(x)$ olacaktır. $P = b_0$ bölüm sonunda kalan değerdir.

Eğer bölüm sonunda kalan sıfır ise bu x0 değeri polinomun köküdür. Sentetik bölme işlemi aşağıdaki şekilde tanımlanı:

$b_n = a_n$

$b_i = a_i + b_{i+1}x_0$

Örneğin eğer polinom

$f_n(x) = -4 + 3x - 3x^2 + 2x^4$

x_0	a_4	a_3	a_2	a_1	a_0
-2	2	0	-3	3	-4

	b_4	b_3	b_2	b_1	b_0
	2	-4	5	-7	10

$$Q(x) = f_{n-1}(x) = -7 + 5x - 4x^2 + 2x^3$$

$$f_n(x) = 10 + (x+2) f_{n-1}(x) \quad P(x) = 10 + (x+2)Q(x)$$

eğer $f_n(x)$ fonksiyonunun x e göre türevini alacak olursak

$f'_n(x) = f_{n-1}(x) + (x - x_0) f'_{n-1}(x)$ olur ve x=x_0 noktasında

$$f'_n(x_0) = f_{n-1}(x_0)$$

Newton - Raphson kök bulma metodu kök değerini bulmak için kullanırsak:

$$x_n = x_{n-1} - \frac{f_n(x_{n-1})}{f'_n(x_{n-1})} = x_{n-1} - \frac{b_0}{f_{n-1}(x_{n-1})}$$

$$x_n = x_{n-1} - \frac{b_0}{Q(x_{n-1})}$$

Polinomun kökleri kompleks sayılar olabileceğinden bu işlemi kompleks sayılarla yapmak gerekir. Sentetik bölme işlemi bir alt polinomun katsayılarını verdiğinden işlem seri olarak diğer kökleri bulmak içinde kullanılabilir. Polinom üöüncü veya daha düşük değerlere ulaştığında daha önce bahsettiğimiz standart kök bulma formüllerine geçilerek son kökler elde edilebilir. Aşağıda Horner metodunu kullanan örnek problemimiz verilmiştir.

Program 2.16-1 Kompleks Horner methodu ile polinomun köklerinin bulunması

```
public class horner
{
complex C[];
public static double f_poly(double c[],double x)
{
// this function calculates the value of
// least square curve fitting function
int n=c.length;
double ff;
if(n!=0.0)
  { ff=c[n-1];
  for(int i=n-2;i>=0;i--)
   { ff=ff*x+c[i]; }
  }
else
  ff=0;
return ff;
}

public static complex f_poly(complex c[],complex x)
{
// this function calculates the value of
// least square curve fitting function
int n=c.length;
complex ff;
if(n!=0.0)
  { ff=new complex(c[n-1]);
  for(int i=n-2;i>=0;i--)
   { ff=complex.add(complex.multiply(ff,x),c[i]); }
  }
else
  ff=new complex(0,0);
return ff;
}
public double[] toDouble()
{int n=C.length;
 double d[]=new double[n];
 for(int i=0;i<n;i++)
    {   d[i]=complex.toDouble(C[i]);}
    return d;
}
public complex horner(double c[], complex x)
{   int n=c.length;
    boolean b=true;
    int j=0;
              complex c1[]=new complex[n];
```

```
                int n1=n-1;
                complex cp[]=new complex[n1];
                c1[n-1]=new complex(c[n-1],0);
    for(int i=n-2;i>=0;i--)
    {   complex c2=new complex(c[i]);
                    c1[i]=complex.add(c2,complex.multiply(x,c1[i+1]));
    }
                complex P=c1[0];
                for(int i=n1-1;i>=0;i--)
                {cp[i]=c1[i+1];}
                C=cp;
                complex Q=f_poly(cp,x);
                x=complex.substract(x,complex.divide(P,Q));
                System.out.println("x="+x+"P="+P+"Q="+Q+"C="+toString());
                if(P.R()>0 && P.R()<1.0e-15) return x;
                else
                {
                while(b)
                {
                c1[n-1]=new complex(c[n-1],0);
    for(int i=n-2;i>=0;i--)
    {   complex c2=new complex(c[i]);
                    c1[i]=complex.add(c2,complex.multiply(x,c1[i+1]));
                }
                P=c1[0];
                for(int i=n1-1;i>=0;i--)
                {cp[i]=c1[i+1];}
                C=cp;
    Q=f_poly(cp,x);
                if(P.R()>0 && P.R()<1.0e-15) b=false;
                x=complex.substract(x,complex.divide(P,Q));
                System.out.println("x="+x+"P="+P+"Q="+Q+"C="+toString());
    }
}
   return x;

}
public complex horner(complex c[], complex x)
{   int n=c.length;
    boolean b=true;
    int j=0;
                complex c1[]=new complex[n];
                int n1=n-1;
                complex cp[]=new complex[n1];
                c1[n-1]=new complex(c[n-1]);
    for(int i=n-2;i>=0;i--)
    {   complex c2=new complex(c[i]);
                    c1[i]=complex.add(c2,complex.multiply(x,c1[i+1]));
    }
                complex P=c1[0];
                for(int i=n1-1;i>=0;i--)
                {cp[i]=c1[i+1];}
                C=cp;
                complex Q=f_poly(cp,x);
                x=complex.substract(x,complex.divide(P,Q));
                System.out.println("x="+x+"P="+P+"Q="+Q+"C="+toString());
                if(P.R()>0 && P.R()<1.0e-5) return x;
                else
                {
                while(b)
                {
                c1[n-1]=new complex(c[n-1]);
    for(int i=n-2;i>=0;i--)
    {   complex c2=new complex(c[i]);
                    c1[i]=complex.add(c2,complex.multiply(x,c1[i+1]));
                }
                P=c1[0];
                for(int i=n1-1;i>=0;i--)
                {cp[i]=c1[i+1];}
                C=cp;
    Q=f_poly(cp,x);
                if(P.R()>0 && P.R()<1.0e-5) b=false;
                x=complex.substract(x,complex.divide(P,Q));
                System.out.println("x="+x+"P="+P+"Q="+Q+"C="+toString());
    }
}
```

```
   return x;
}

public String toString()
{int n=C.length;
 String s="";
 for(int i=n-1;i>=0;i--) {s+=C[i]+" ";}
 return s;
}

public static void main(String arg[])
{
        double c[]={-4,3,-3,0,2};
        complex x=new complex(-2.0,0);
   horner h=new horner();
        complex x1=h.horner(c, x);
        System.out.println("Horner method x1="+x1);
        System.out.println("C="+h.toString());
        double d[]=h.toDouble();
        complex z[]=NA13D.cubic_rootsC(d);
   String s1="cubic root finding method: \n";
   for(int i=0;i<3;i++)
   {s1+=z[i]+"\n";}
   System.out.println(s1);
        complex x2=h.horner(h.C,x);
        System.out.println("Horner method x2="+x2);
        System.out.println("C="+h.toString());
        x=new complex(0.2,1.0);
        complex x3=h.horner(h.C,x);
        System.out.println("Horner method x3="+x3);
        System.out.println("C="+h.toString());
        //last root C1*x+C0=0 --> x=-C0/C1
        complex x4=complex.multiply(complex.divide(h.C[0],h.C[1]),-1.0);
        System.out.println("Horner method x4="+x4);
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" horner
x=(  -1.7959183673)P=(  10.0000000000)Q=( -49.0000000000)C=(   2.0000000000) (  -4.0000000000) (   5.0000000000) (  -7.0000000000)
x=(  -1.7424329168)P=(   1.7416906499)Q=( -32.5638211970)C=(   2.0000000000) (  -3.5918367347) (   3.4506455643) (  -3.1970777482)
x=(  -1.7389702353)P=(   0.0999559202)Q=( -28.8666233657)C=(   2.0000000000) (  -3.4848658335) (   3.0721449388) (  -2.3530064663)
x=(  -1.7389562567)P=(   0.0004002871)Q=( -28.6355896903)C=(   2.0000000000) (  -3.4779404707) (   3.0480349588) (  -2.3004420696)
x=(  -1.7389562565)P=(   0.0000000065)Q=( -28.6346590495)C=(   2.0000000000) (  -3.4779125134) (   3.0479377253) (  -2.3002303774)
x=(  -1.7389562565)P=(   0.0000000000)Q=( -28.6346590344)C=(   2.0000000000) (  -3.4779125129) (   3.0479377237) (  -2.3002303739)
Horner method x1=(  -1.7389562565)
C=(   2.0000000000) (  -3.4779125129) (   3.0479377237) (  -2.3002303739)
cubic root finding method:
(   1.2548818848)
(   0.2420371858 -    0.9262454873i )
(   0.2420371858 +    0.9262454873i )

x=(  -1.0647426414)P=( -38.3077558729)Q=(  40.9595877753)C=(   2.0000000000) (  -7.4779125129) (  18.0037627495)
x=(  -0.3749903930)P=( -11.9024769439)Q=(  17.2561625889)C=(   2.0000000000) (  -5.6073977957) (   9.0183732639)
x=(   0.2461922661)P=(  -4.0376947714)Q=(   6.5000120530)C=(   2.0000000000) (  -4.2278932989) (   4.6333570935)
x=(   1.2648342798)P=(  -1.7308064162)Q=(   1.6991311893)C=(   2.0000000000) (  -2.9855279807) (   2.3129238246)
x=(   1.2549871728)P=(   0.0378996078)Q=(   3.8488063189)C=(   2.0000000000) (  -0.9482439533) (   1.8485662660)
x=(   1.2548818968)P=(   0.0003967246)Q=(   3.7684233632)C=(   2.0000000000) (  -0.9679381674) (   1.8331877396)
x=(   1.2548818848)P=(   0.0000000449)Q=(   3.7675702706)C=(   2.0000000000) (  -0.9681487194) (   1.8330254224)
Horner method x2=(   1.2548818848)
C=(   2.0000000000) (  -0.9681487194) (   1.8330254224)
x=(   0.2390192744 +    0.9282086594i )P=(  -0.2806043215 -    0.1681487194i )Q=(  -0.1681487194 +    4.0000000000i )C=(
2.0000000000) (  -0.5681487194 +    2.0000000000i )
x=(   0.2420307878 +    0.9262426845i )P=(  -0.0072629856 -    0.0112049841i )Q=(  -0.0120716220 +    3.7128346375i )C=(
2.0000000000) (  -0.4901101707 +    1.8564173187i )
x=(   0.2420371799 +    0.9262454938i )P=(   0.0000104083 -    0.0000236822i )Q=(  -0.0000255681 +    3.7049707381i )C=(
2.0000000000) (  -0.4840871437 +    1.8524853691i )
x=(   0.2420371798 +    0.9262454938i )P=(   0.0000000001 +    0.0000000001i )Q=(   0.0000000001 +    3.7049819751i )C=(
2.0000000000) (  -0.4840743597 +    1.8524909875i )
Horner method x3=(   0.2420371798 +    0.9262454938i )
C=(   2.0000000000) (  -0.4840743597 +    1.8524909875i )
Horner method x4=(   0.2420371798 -    0.9262454938i )
```

> Terminated with exit code 0.

2.17 Bairstow metoduyla polinomların kompleks köklerinin bulunması

Bairstow's metodu iterativ bir yöntemdir. Horner metodu gibi sentetik bölmeye dayanır. Fakat temel farklılığı birinci derece yerine ikinci derece bir sentitik bölme işlemi kullanmasıdır. Bunun temel nedeni kompleks köklerdeki conjuge sanal kök açılımını daha effectif olarak değerlendirebilmektir. Bir önceki bölümde sentetik bölme işlemini incelemiştik. $f_n(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$ ve eğer polinom $(x - x_0)$ ile bölünürse

$f_{n-1}(x) = b_1 + b_2x + b_3x^2 + ... + b_{n-1}x^{n-1}$

burada $f_n(x) = b_0 + (x - x_0) f_{n-1}(x)$

$R = b_0$ kalan değeridir.Eğer kalan değeri 0 a eşitse kök bulunmuş olur. Sentetik bölme işleminin bir basamaklı denklemleri:

$b_n = a_n$

$b_i = a_i + b_{i+1}x_0$

Şeklinde idi. Şimdi ikinci dereceden bir polinoma bölme işlemine bakalım. Eğer bölünecek polinom parçamız $x^2 - rx - s$ ise geri kalan polinom $f_{n-2}(x) = b_2 + b_3x + b_3x^2 + ... + b_{n-1}x^{n-3} + b_nx^{n-2}$ olacaktır. Bu polinomun kalan değeri

$R = b_1(x - r) + b_0$ dir. Bu işlem için sentetik bölme denklemleri şu şekilde yazılabilir:

$b_n = a_n$

$b_{n-1} = a_{n-1} + rb_n$

$b_i = a_i + b_{i+1}r + b_{i+2}s$ for i = n-2 den 0 a

Denklemlerden de görüleceği kök bulmak için gibi 2 adet bilinmeyen değerimizin (r ve s) simultane olarak çözülmesi gerekmektedir. B değerinin değişimini Taylor açılımı olarak yazarsak aşağıdaki denklemleri elde ederiz.

$$b_1^n = b_1^{n-1} + \frac{\partial b_1^{n-1}}{\partial r}\Delta r + \frac{\partial b_1^{n-1}}{\partial s}\Delta s$$

$$b_0^n = b_0^{n-1} + \frac{\partial b_0^{n-1}}{\partial r}\Delta r + \frac{\partial b_0^{n-1}}{\partial s}\Delta s$$ Taylor açılımındaki yüksek mertebeli türevler ihmal edilmiştir. Kökleri bulduğumuzda b_0 ve b_1 değerlerinin sıfır olması gerkliliğinden, bu denklemlerin sol taraflarını sıfıra eşitleyebiliriz. Bu durumda

$$\frac{\partial b_1}{\partial r}\Delta r + \frac{\partial b_1}{\partial s}\Delta s = -b_1$$

$$\frac{\partial b_0}{\partial r}\Delta r + \frac{\partial b_0}{\partial s}\Delta s = -b_0$$ Eğer kısmi türev değerleri bilinirse bu denklemler çözülebilir. Bairstow kısmi türevlerin yine sentetik bölme işlemi kullanılarak b değerlerinden elde edilebileceğini göstermiştir. Kısmi türevler bulunduktan sonra bu değerler üstteki denkleme koyularak denklem sistemi çözülebilir.

$c_n = c_n$

$c_{n-1} = b_{n-1} + rc_n$

$c_i = b_i + c_{i+1}r + c_{i+2}s$ for i = n-2 to 1

Ve denklem aşağıdaki formu alır :

$$c_2\Delta r + c_3\Delta s = -b_1$$

$$c_1\Delta r + c_2\Delta s = -b_0$$

Bu denklem sistemini Gauss eliminasyon prosesi kullanarak çözebiliriz.

$$W = \frac{c_1}{c_2}$$

$$\Delta s = \frac{-b_0 + Wb_1}{c_2 - Wc_3}$$

$$\Delta r = \frac{-b_1 - c_3\Delta s}{c_2}$$

$$r^{n+1} = r^n + \Delta r^{n+1}$$

$$s^{n+1} = s^n + \Delta s^{n+1}$$

Şüphesiz çözüm iteratif olarak devam ettirilmelidir. Eğer hata terimi önceden saptanmış bir seviyenin altına çekilebilirse iterasyon prosesi durdurulur.

$$|\varepsilon| = \left|\frac{\Delta r}{r}\right| + \left|\frac{\Delta s}{s}\right| < \varepsilon_s$$

Bundan sonra kökler ikinci dereceden polinom formülü kullanılarak çözülebilir.

$$x = \frac{r \pm \sqrt{r^2 + 4s}}{2}$$

Burada polinom köklerinin kompleks sayılar olduğunu göz önüne alarak tüm işlemin kompleks olarak yapılması gerektiğini tekrar belirtelim. Örnek program ve çıktısı aşağıda listelenmiştir

Program 2.17-1 Kompleks Bairstow metodu örneği

```
 public class bairstow
{
complex C[];

public static double f_poly(double c[],double x)
{
int n=c.length;
double ff;
if(n!=0.0)
  { ff=c[n-1];
  for(int i=n-2;i>=0;i--)
   {ff=ff*x+c[i];}
  }
else
  ff=0;
return ff;
}

public static complex f_poly(complex c[],complex x)
{
int n=c.length;
complex ff;
if(n!=0.0)
  { ff=new complex(c[n-1]);
  for(int i=n-2;i>=0;i--)
   { ff=complex.add(complex.multiply(ff,x),c[i]); }
  }
else
  ff=new complex(0,0);
return ff;
}
public double[] toDouble()
{int n=C.length;
 double d[]=new double[n];
 for(int i=0;i<n;i++)
```

```
    {   d[i]=complex.toDouble(C[i]);}
    return d;
}
public double[] toDouble(complex w[])
{int n=w.length;
 double d[]=new double[n];
 for(int i=0;i<n;i++)
    {   d[i]=complex.toDouble(w[i]);}
    return d;
}
public complex[] bairstow(double a[], complex r,complex s)
{   int n=a.length;
    boolean bb=true;
    int j=0;
                complex b[]=new complex[n];
                int n1=n-2;
                complex cp[]=new complex[n1];
                complex c[]=new complex[n];
                complex ds,dr;
                while(bb)
                {
                b[n-1]=new complex(a[n-1]);
                b[n-2]=complex.add(complex.multiply(r,b[n-1]),a[n-2]);
    for(int i=n1-1;i>=0;i--)
    {   complex c2=new complex(a[i]);
                    b[i]=complex.add(c2,complex.multiply(r,b[i+1]));
                    b[i]=complex.add(b[i],complex.multiply(s,b[i+2]));
    }
    c[n-1]=b[n-1];
    c[n-2]=complex.add(b[n-2],complex.multiply(r,c[n-1]));
    for(int i=n1-1;i>=0;i--)
    {   c[i]=complex.add(b[i],complex.multiply(r,c[i+1]));
                    c[i]=complex.add(c[i],complex.multiply(s,c[i+2]));
    }
                for(int i=n1-1;i>=0;i--)
                {cp[i]=b[i+2];}
                C=cp;
                complex CC=complex.divide(c[1],c[2]);
                complex b0=complex.multiply(CC,b[1]);
                b0=complex.substract(b0,b[0]);
                complex c0=complex.multiply(CC,c[3]);
                c0=complex.substract(c[2],c0);
                ds=complex.divide(b0,c0);
                complex d0=complex.multiply(ds,c[3]);
                d0=complex.add(d0,b[1]);
                d0=complex.multiply(d0,-1.0);
                dr=complex.divide(d0,c[2]);
                r=complex.add(r,dr);
                s=complex.add(s,ds);
                System.out.println("dr="+dr+"ds="+ds+"r="+r+"s="+s);
                double de=complex.abs(complex.divide(dr,r))+complex.abs(complex.divide(ds,s));
                if(de<1e-10) bb=false;
    }
    complex x[]=new complex[2];
    complex DD=complex.sqrt(complex.add(complex.multiply(r,r),complex.multiply(s,4.0)));
    x[0]=complex.add(r,DD);
    x[0]=complex.divide(x[0],2.0);
    x[1]=complex.substract(r,DD);
    x[1]=complex.divide(x[1],2.0);
   return x;

}
public complex[] bairstow(complex a[], complex r,complex s)
{   int n=a.length;
    boolean bb=true;
    int j=0;
                complex b[]=new complex[n];
                int n1=n-1;
                complex cp[]=new complex[n1];
                complex c[]=new complex[n];
                complex ds,dr;
                while(bb)
                {
                b[n-1]=a[n-1];
                complex z1=complex.multiply(r,b[n-1]);
                b[n-2]=complex.add(z1,a[n-2]);
```

```
    for(int i=n-3;i>=0;i--)
    {   complex c2=new complex(a[i]);
                b[i]=complex.add(c2,complex.multiply(r,b[i+1]));
                b[i]=complex.add(b[i],complex.multiply(s,b[i+2]));
    }
    c[n-1]=b[n-1];
    c[n-2]=complex.add(b[n-2],complex.multiply(r,c[n-1]));
    for(int i=n-3;i>=0;i--)
    {   c[i]=complex.add(b[i],complex.multiply(r,c[i+1]));
                c[i]=complex.add(c[i],complex.multiply(s,c[i+2]));
    }
                for(int i=n1-2;i>=0;i--)
                {cp[i]=b[i+2];}
                C=cp;
                complex CC=complex.divide(c[1],c[2]);
                complex b0=complex.multiply(CC,b[1]);
                b0=complex.substract(b0,b[0]);
                complex c0=complex.multiply(CC,c[3]);
                c0=complex.substract(c[2],c0);
                ds=complex.divide(b0,c0);
                complex d0=complex.multiply(ds,c[3]);
                d0=complex.add(d0,b[1]);
                d0=complex.multiply(d0,-1.0);
                dr=complex.divide(d0,c[2]);
                r=complex.add(r,dr);
                s=complex.add(s,ds);
                System.out.println("dr="+dr+"ds="+ds+"r="+r+"s="+s);
                double de=complex.abs(complex.divide(dr,r))+complex.abs(complex.divide(ds,s));
                if(de<1e-10) bb=false;
    }
    complex x[]=new complex[2];
    complex DD=complex.sqrt(complex.add(complex.multiply(r,r),complex.multiply(s,4.0)));
    x[0]=complex.add(r,DD);
    x[0]=complex.divide(x[0],2.0);
    x[1]=complex.substract(r,DD);
    x[1]=complex.divide(x[1],2.0);
   return x;

}

public String toString()
{int n1=C.length;
 String s="";
 for(int i=n1-1;i>=0;i--) {s+=C[i]+" ";}
 return s;
}

public String toString(complex w[])
{int n1=w.length;
 String s="";
 for(int i=n1-1;i>=0;i--) {s+=w[i]+" ";}
 return s;
}
public static void main(String arg[])
{
                double a[]={1.25,-3.875,2.125,2.75,-3.5,1};
                complex r=new complex(-1.0,0.0);
                complex s=new complex(-1.0,0.0);
                bairstow h=new bairstow();
                complex x[]=h.bairstow(a,r,s);
                System.out.println("Bairstow method x0="+x[0]);
                System.out.println("Bairstow method x1="+x[1]);
                System.out.println("C="+h.toString());
                r=new complex(-0.5,0.0);
                s=new complex(0.50,0.0);
    x=h.bairstow(h.C,r,s);
                System.out.println("Bairstow method x2="+x[0]);
                System.out.println("Bairstow method x3="+x[1]);
                System.out.println("C="+h.toString());
                complex x4=complex.multiply(complex.divide(h.C[0],h.C[1]),-1.0);
                System.out.println("Bairstow method x4="+x4);
}
}
```

```
---------- Capture Output ----------
```

```
> "C:\java\bin\javaw.exe" bairstow
dr=(   0.3558301400)ds=(   1.1381090170)r=(  -0.6441698600)s=(   0.1381090170)
dr=(   0.1330567636)ds=(   0.3316246083)r=(  -0.5111130965)s=(   0.4697336253)
dr=(   0.0114266228)ds=(   0.0304686946)r=(  -0.4996864737)s=(   0.5002023199)
dr=(  -0.0003135916)ds=(  -0.0002023303)r=(  -0.5000000653)s=(   0.4999999896)
dr=(   0.0000000653)ds=(   0.0000000104)r=(  -0.5000000000)s=(   0.5000000000)
dr=(  -0.0000000000)ds=(  -0.0000000000)r=(  -0.5000000000)s=(   0.5000000000)
Bairstow method x0=(   0.5000000000)
Bairstow method x1=(  -1.0000000000)
C=(   1.0000000000) (  -4.0000000000) (   5.2500000000) (  -2.5000000000)
dr=(   2.2321428571)ds=(   3.1607142857)r=(   1.7321428571)s=(   3.6607142857)
dr=(  -0.0669786588)ds=(  -5.0183431591)r=(   1.6651641983)s=(  -1.3576288733)
dr=(   0.2332468806)ds=(   0.1517126717)r=(   1.8984110789)s=(  -1.2059162016)
dr=(   0.0941871444)ds=(  -0.0352673665)r=(   1.9925982232)s=(  -1.2411835681)
dr=(   0.0074183706)ds=(  -0.0087613999)r=(   2.0000165939)s=(  -1.2499449681)
dr=(  -0.0000165958)ds=(  -0.0000550317)r=(   1.9999999981)s=(  -1.2499999997)
dr=(   0.0000000019)ds=(  -0.0000000003)r=(   2.0000000000)s=(  -1.2500000000)
dr=(   0.0000000000)ds=(   0.0000000000)r=(   2.0000000000)s=(  -1.2500000000)
Bairstow method x2=(   1.0000000000 +    0.5000000000i )
Bairstow method x3=(   1.0000000000 -    0.5000000000i )
C= (   1.0000000000) (  -2.0000000000)
Bairstow method x4=(   2.0000000000)

> Terminated with exit code 0.
```

2.18 Paket programlarla kök buluması

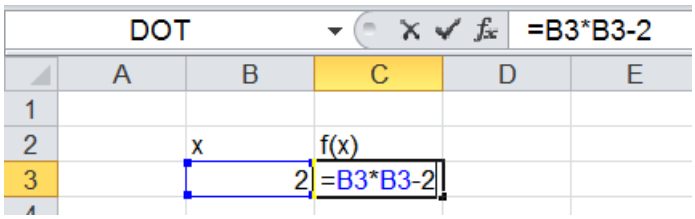
Hazır paket programlarda kök bulmak için oluşturulmuş direk olarak kullanabileceğimiz metodlar mevcuttur. Matlab-Octave programlarında kök bulmak için fzero foksiyonunu, polinomların kökünü bulmak içinse roots fonksiyonunu kullanabiliriz.

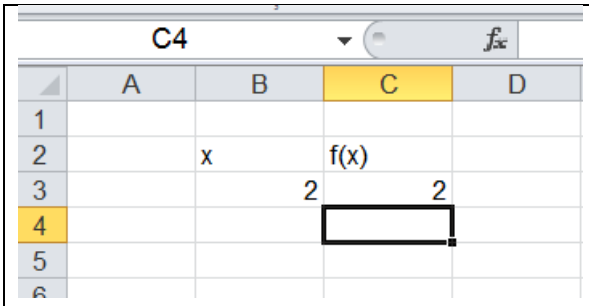
```
octave.exe:4> format long
octave.exe:5> x=2;
octave.exe:6> y=@(x)x*x-2;
octave.exe:7> fzero(y,x)
ans =  1.414213576373l0
octave.exe:8>
```

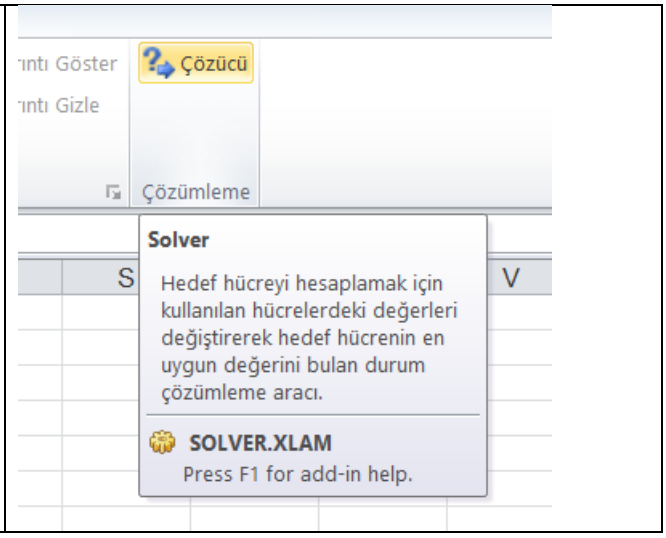
```
octave.exe:13> x=1;
octave.exe:14> y=@(x)x^5-3.5*x^4+2.75*x^3+2.125*x^2-3.875*x+1.25;
octave.exe:15> fzero(y,x)
ans =  0.499999986120035
octave.exe:16>
```

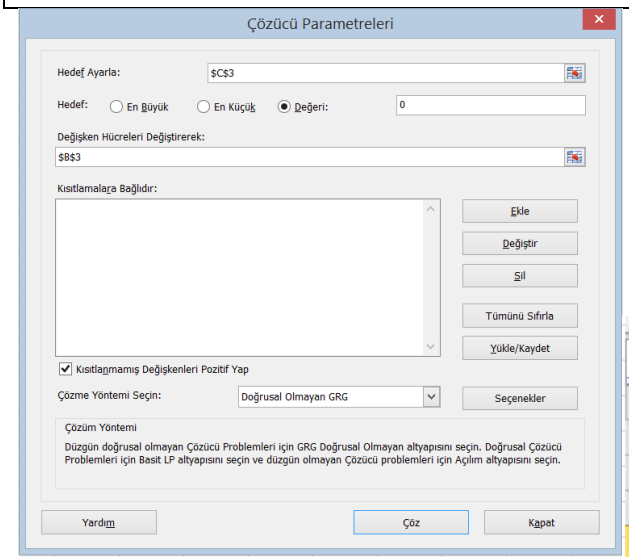
```
octave.exe:8>
octave.exe:8> a=[1 -3.5 2.75 2.125 -3.875 1.25];
octave.exe:9> roots(a)
ans =

   2.000000000000004 + 0.000000000000000i
  -1.000000000000001 + 0.000000000000000i
   0.999999999999999 + 0.500000000000001i
   0.999999999999999 - 0.500000000000001i
   0.499999999999999 + 0.000000000000000i

octave.exe:10>
```

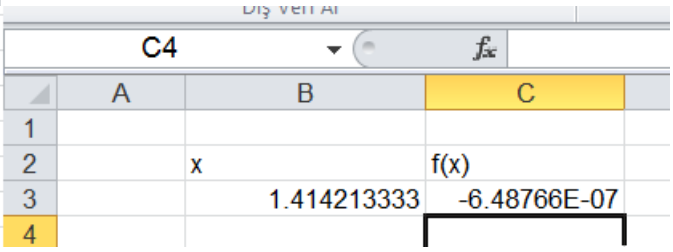
Excel de ise solver(çözücü) fonksiyonundan yararlanarak kök bulabiliriz.











2.19 Problemler

PROBLEM 1

r yarıçapındaki küresel bir tankın içindeki sıvının V hacmi derinliğine

$V = \pi h^2(3r - h) / 3$

Şeklinde bağlıdır. **r = 1 m ve V= 0.5 m^3 olarak verildiğine göre h yi bulunuz.**

Not : geometrinin(küre) gereği h nin değeri 0 dan küçük, d=2r den büyük olamaz. Kabın alabileceği maksimum sıvı miktarı 4.18879 m^3 dür (bu veriye problemi çözmek için ihtiyacınız yok, fikir edinebilmeniz için verilmiştir.) . Kök bulma sayısal çözümleme metodu metodu size bırakılmıştır (ikiye bölme, yer değiştirme, newton, sekant, müller...). Kullandığınız metodu ve formülü açık olarak belirtiniz. Çözüme başlarken önce formülü f(h) = 0 formuna getiriniz.

PROBLEM 2

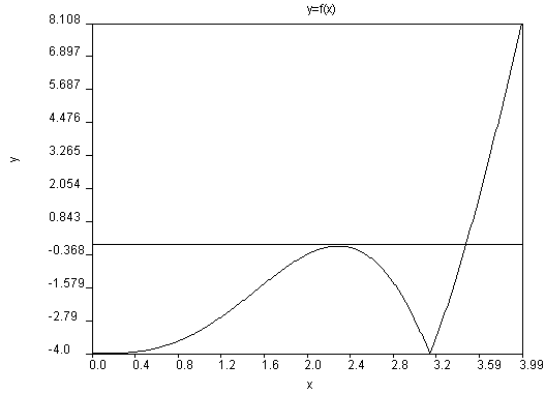
$f(x) = x^2 - 0.9x - 8.5$ fonksiyonunun köklerini ikiye bölme yöntemi kullanarak bulunuz.

PROBLEM 3

$\sqrt{7}$ değerini kök bulma yöntemlerinden birini kullanarak hesaplayınız.

PROBLEM 4

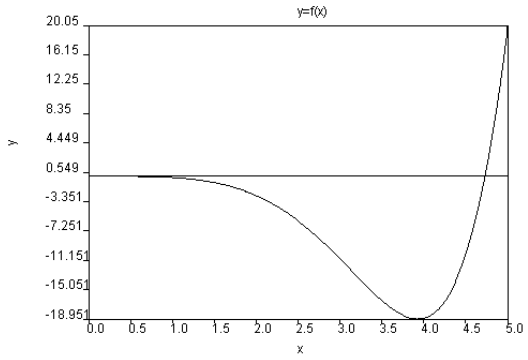
$f(x) = x^2|\sin(x)| - 4$ değerini kök bulma yöntemlerinden birini kullanarak hesaplayınız.



(bolum2_016.jpg,Resimaltı:)

PROBLEM 5

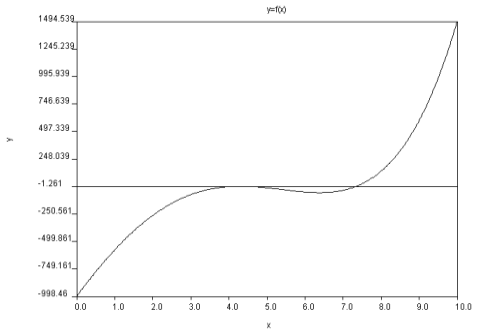
$f(x) = \cos(x)\cosh(x) - 1$ değerini kök bulma yöntemlerinden birini kullanarak hesaplayınız.



(bolum2_017.jpg,Resimaltı:)

PROBLEM 6

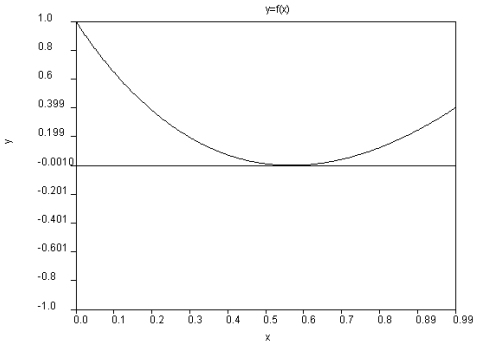
$f(x) = x^4 - 8.6x^3 - 35.5x^2 + 464.4x - 998.46$ değerini kök bulma yöntemlerinden birini kullanarak hesaplayınız. x_0=4 ve $7 \le x \le 8$ değerlerini deneyiniz



(bolum2_018.jpg,Resimaltı:)

PROBLEM 7

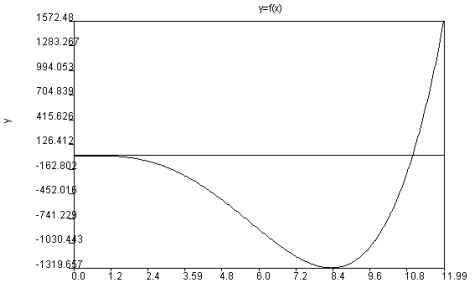
$f(x) = x^2 - 2e^{-x}x + e^{-2x}$ fonksiyonunun tek bir çoklu gerçek kök değeri mevcuttur. Newton yöntemini kullanarak hesaplayınız. x_0=1



(bolum2_019.jpg,Resimaltı:)

PROBLEM 8

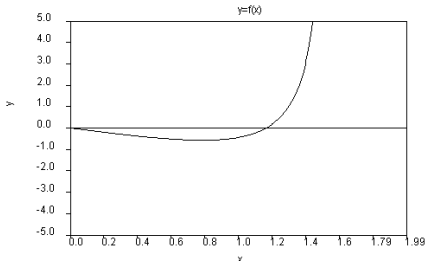
$f(x) = x^4 - 7.223x^3 + 13.447x^2 - 0.672x - 10.223$ fonksiyonunun **tüm gerçek** köklerini bulunuz.



(bolum2_020.jpg,Resimaltı:)

PROBLEM 9

$f(x) = \tan(x) - 2x$ fonksiyonunun köklerini bulunuz.



(bolum2_021.jpg,Resimaltı:)

PROBLEM 10

Boru içi akışta türbülanslı akış için (Re>2300) borudaki sürtünme kat sayısı boru yüzey pürüzlülüğünün fonksiyonu olarak Colebrook denklemiyle hesaplanır. Colebrook denklemi :

$$\frac{1}{\sqrt{f}} = -2.0\log_{10}\left[\frac{\varepsilon/D}{3.7} + \frac{2.51}{\mathrm{Re}\sqrt{f}}\right]$$

bu denklemden hesaplanacak sürtünme katsayısına ilk yaklaşım için (f_0) Olujic-Jain denklemini alabiliriz. Bu denklem :

$$f = 4.0\left[-1.737\ln\left[0.269\frac{\varepsilon}{D} - \frac{2.185}{\mathrm{Re}}\left(\ln(0.269\frac{\varepsilon}{D}) + \frac{14.5}{\mathrm{Re}}\right)\right]\right]^{-2}$$ formundadır.

Laminer akışta sürtünme katsayısı(dairesel boru profili için) f=64/Re eşitliğinden hesaplanabilir. Re sayısı ve ε/D (pürüzlülük katsayısı/boru çapı) ve yerel basınç düşümü katsayısı verildiğinde borudaki toplam basınç düşümünü hesaplayan bir program yazınız ve test ediniz.

Borudaki toplam basınç düşümü : DP= (K+f*L/D)*□*U^2/2

DP : basınç düşümü

K : yerel basınç düşüm katsayısı

f :sürtünme katsayısı
L boru boyu
D boru çapı
ρ : sıvı yoğunluğu
U sıvı hızı

PROBLEM 11

Dairesel pürüzsüz borularda türbülanslı akış bölgesinde Fanning sürtünme katsayısının hesaplanması için önerilen denklemlerden biri de aşağıdaki gibidir:

$$1/\sqrt{f} \;=\; 1{,}737 * \ln(.\mathrm{Re} * \sqrt{f}\;) - 0{,}4$$

Bu denklem $4 * 10^3 < \mathrm{Re} < 3 * 10^6$ aralığında geçerlidir. Reynolds sayısının 5000 değeri için fannning sürtünme katsayısını (f) hesaplayınız.
Not 1: Önce kökün(f) bulunduğu aralığı hesaplayınız. Kök değeri (f) 0,001 ile 0.01 arasında değişmektedir.
Not 2: ε_S = % 0,01 olana dek işlemi sürdürünüz.

PROBLEM 12
Gazlar için van der Waals denklemi :

$P = \dfrac{RT}{v-b} - \dfrac{a}{v^2}$ şeklindedir. Burada

$$a = \frac{27}{64}\frac{R^2T_c^2}{P_c}$$

$$b = \frac{RT_c}{8P_c}$$

olarak verilebilir.Redlich ve Kwong Denklemi

$P = \dfrac{RT}{v-b} - \dfrac{a}{v(v+b)T^{1/2}}$ şeklindedir. Burada

$$a = 0.42748\frac{R^2T_c^{5/2}}{P_c}$$

$$b = 0.08664\frac{RT_c}{P_c}$$

O2, oksijen için P_c=5.04 MPa T_c=154.6 K M=31.999 kg/kmol olarak verilmiştir.
N2, azot için P_c=3.39 MPa T_c=126.2 K M=28.013 kg/kmol olarak verilmiştir.

Benedict -Webb-Rubin (BWR) denklemi

$$P = \frac{RT}{v} + \frac{RTB_0 - A_0 - C_0/T^2}{v^2} + \frac{RTb - a}{v^3} + \frac{a\alpha}{v_6} + \frac{c}{v_3T^2}(1 + \frac{\gamma}{v^2})e^{-\frac{\gamma}{v^2}}$$

Gaz	A_0	B_0	$C_0.10^{-6}$	A	B	$c.10^{-6}$	□.10^3	□.10^3
O_2	1.49880	0.046524	0.0038617	-0.040507	-0.00027963	-0.00020376	0.008641	0.359
N_2	1.1925	0.0458	0.0058891	0.01490	0.00198154	0.000548064	0.291545	0.75

Java programı geliştirerek (Örnek programlardan yararlanabilirsiniz)T ve v verildiğinde P'yi hesaplayabilen, P ve v verildiğinde T yi hesaplayabilen, P ve T verildiğinde v'yi hesaplayabilen bir program geliştiriniz.

A) van der Waals denklemi : gaz : oksijen ve azot
B) Redlich Kwong denklemi : gaz oksijen ve azot
C) Benedict-webb rubin denklemi : gaz : oksijen
D) Benedict-webb rubin denklemi : gaz : azot

Bulunan değerleri ideal gaz denklemi ile karşılaştırınız $P = \frac{RT}{v}$

PROBLEM 13

$f(x)=x\,e^{-x/2} + x^3 + 1$ fonksiyonunun köklerini $x_1=-1$, $x_2=0$, $x_3=1$ alarak müler yöntemi ile çözünüz.

PROBLEM 14

$f(x)=x^4-20x^3-25x^2+100x+130$ fonksiyonunun köklerini Leguerre yöntemi ile çözünüz.

PROBLEM 15

$f(x)=x^2+1$ fonksiyonunun köklerini bulunuz.

PROBLEM 16

Paralel akışklı bir ısı değiştiricinin verimlilik deneyleri sonucu transfer ünitesi sayısı *NTU*=1.989 ve etkinliği ε = 0.574 olarak bulunmuştur. Isı değiştirici etkinliği (ε)

$$\varepsilon = \left\{1 - \exp\left[-NTU \cdot (1 + C^*)\right]\right\} \Big/ (1 + C^*)$$

denklemi ile tanımlanmaktadır. **A)** Denklemi sağlayan *C** değerinin bulunduğu aralığı belirleyiniz. Bunun için ilk değeri *C**=0.1, artımları da 0.1 olarak alınız, **B)** Yer değiştirme yöntemini kullanarak *C** değerini hesaplayınız. ε_a < % 0.1 olana dek işlemi sürdürünüz.

PROBLEM 17

Uzunluğu 5 cm, kesiti (0.1 cm × 20 cm) olan bir levhanın iki ucu 80 °C ve 60 °C sıcaklıkta tutulmaktadır. Çubuğun bulunduğu ortam T_∞ = 10 °C sıcaklıkta ve olup ısı taşınım katsatısı *h*=15 W/m^2 °C'dır. Çubuk kesit alanı *A*=2×10^{-4} m^2, çubuk çevresi *P*=0.402 m'dir. Yapılan deneyde çubuğun orta noktasının (*x*=0.025 m) sıcaklığı *T*=67.5 °C olarak bulunmuştur. Bu koşullarda çubuktaki sıcaklık dağılımı

$$T - T_\infty = 68.7 \cdot e^{-0.025m} + 11.3 \cdot e^{0.025m}$$

denklemi ile verilmektedir.

Bu denklemde $m = (h \cdot P / k \cdot A)^{0.5}$ olarak tanımlanmaktadır. **A)** Sıcaklık dağılımı denklemini sağlayan m değerinin bulunduğu aralığı belirleyiniz. Bunun için ilk değeri *m*=5, artımları da 5 olarak alınız. **B)** Newton-Raphson yöntemini kullanarak m değerini hesaplayınız. ε_a < % 1 olana dek işlemi sürdürünüz. **C)** Bulacağınız değeri kullanarak malzemenin ısı iletim katsayısını (*k*) hesaplayınız.

PROBLEM 18

Bir ısı değiştiricinin maliyet fonksiyonu (M), ısı değiştiricinin boyu (L) ve gövde çapının (D) fonksiyonu olup,

$$M=900 + 1100\ D^{2,5}\ L + 320\ D\ L$$

eşitliği ile tanımlanmaktadır. Yapılan optimizasyon çalışması sonucu ısı değiştiricinin maliyeti 1900 YTL, boyu ise 1,3 m elde edilmiştir. Bu verilere göre gövde çapını hesaplayınız. **A)** Kökün bulunduğu aralığı (gövde çapını 0,1 m arttırarak) bulunuz. **B)** Gövde çapını(D), ikiye bölme yöntemi ile hesaplayınız. Yaklaşık bağıl yüzde hata(ε_a) % 2 değerinden küçük olana dek işleme devam ediniz. Her adımda yaklaşık bağıl yüzde hatayı hesaplayınız.

PROBLEM 19

20.000.000. TL değerindeki bir cihazı hiç peşin ödemeden 6 yılda geri ödemek üzere yılda 4.000.000. TL taksitle satın alıyorsunuz. Borcunuza uygulanan faizin değerini Sekant yöntemini kullanarak hesaplayınız.? Bugünkü değer P, yıllık ödeme A, yıl sayısı n ve faiz oranı i arasındaki bağıntı aşağıda verilmiştir. **İp ucu:** Kökün bulunduğu aralığı 0.05 artımla belirleyiniz ve yaklaşık kök bulunması işlemine alt sınırdan başlayınız. Yaklaşık kök hesaplamasını üç adım yürütünüz ve yaklaşık bağıl yüzde hatayı bulunuz.

$$A = P\ \{i\ (1+i)^n\ /\ [(1+i)^n\ -1]\}$$

PROBLEM 20

Redlich- Kwong hal denklemi

$$P = [R\ T\ /(v-b)] - a\ /\ [v(v+b)\sqrt{T}\ ]$$

şeklinde verilmiştir. Burada *R*, gaz sabiti, *T* mutlak sıcaklık (K), *P* mutlak basınç (kPa), *v*, bir kg gazın hacmi (özgül hacim) (m^3/kg) olup a ve b parametreleri ise: a = 0,427($R^2\ T_c$ / P_c) ve b = 0,0866 (T_c/ P_c) bağıntılarıyla hesaplanabilir. Burada P_c = 4600 kPa, T_c = 191 K'dir. Elde mevcut bir tankın hacmi 3 m^3 olup içine –40 °C sıcaklıkta ve 65000 kPa basınçta metan gazı depolanmaktadır. Metan gazı için gaz sabiti R=0,518 kJ/kgK dir.
a) Kökün (v) bulunduğu aralığı,
b) Yer değiştirme yöntemini kullanarak bir kg gazın hacmini (v),
c) Depolanacak metan gazı kütlesini bulunuz.
Not: İşlemi üç adım yürütünüz ve her adımda yaklaşık bağıl yüzde hatayı hesaplayınız.

PROBLEM 21

Bir maddeyi belli bir sıcaklığa kadar ısıtmak için gerekli enerji Q = $\int$ m * C_p * dT eşitliği ile bulunmaktadır.

Sıcaklığı 30 °C, kütlesi 0,2 kg olan bir madde 80 °C'da bozunmaktadır. Bu maddenin özgül ısısının sıcaklıkla değişimi:
C_p =10 + 0,4 * T + 18 *10^{-4} * T^2 (kcal/kg K) olarak verilmektedir. Bu maddeye 733,16 kcal değerinde enerji verilmektedir. **A)** Ulaşılacak sıcaklığın (kökün) bulunduğu aralığı hesaplayınız. İlk değeri 283 K, artımları ise 5 olarak alınız. **B)** Son sıcaklığı (kök değerini) ikiye bölme yöntemi ile hesaplayınız. İşleme ε_a= %2 olana dek devam ediniz.

PROBLEM 22

Bir ısı değiştirici için yazılan enerji dengesi ve ısı transfer hız denklemleri birlikte çözüldüğünde sistem

$$f = 1 - \exp(-0{,}508\ /\ w) - 0{,}341\ /\ w$$

denklemi ile tanımlanmaktadır. Burada w (kg/s) kütlesel debidir. Bu koşullarda **a)** f = f(w) denkleminde kütlesel debinin (w) bulunduğu aralığı hesaplayınız. Debi 0,4 kg/s'den büyüktür, **b)** Newton-Raphson yöntemi ile debinin (w) değerini hesaplayınız. Kök hesaplama işlemini yaklaşık bağıl yanlış %0,5 olana dek sürdürünüz.

PROBLEM 23

Kurutma süreçlerinde, kurutma havasından, kurutulan malzemeye ısı
transferi taşınım yanında radyasyon ve iletimle olduğunda malzeme yüzey sıcaklığı T_S

$$f = [1{,}56*(T_G-T_S)] / Q_V - (1/C_P)*\left[\frac{0{,}622}{\{P- \exp(4288/T_S - 10{,}93) - 1\}} - Y_G\right]$$

eşitliği ile bulunur.

Burada C_P =1,005+1,884*Y_G ve Q_V = 2501- 2,44*(T_S-273) olarak tanımlanmaktadır. Kurutma havası sıcaklığı T_G=70°C, nemi Y_G =0,01066 ise a) 0 °C ile 100 °C arasında kökün bulunduğu aralığı (artımları en fazla 10 °C olarak alınız), b) malzeme sıcaklığını Bisection yöntemi ile hesaplayınız.

PROBLEM 24

Doğal dolaşımlı güneş enerjili sıcak su sisteminde kütlesel debi m(kg/sn) aşağıdaki eşitlikle hesaplanabilir.

$$m = - \frac{A_C * U_L * F'}{C_P * \ln\left[1 - \frac{U_L * (T_O - T_İ)}{S - U_L * (T_İ - T_A)}\right]}$$

Yapılan bir deneyde toplayıcı yüzeyinde absorbe edilen ışınım şiddeti S= 780 W/m^2, çevre sıcaklığı T_A=15 °C, akışkanın toplayıcıya giriş sıcaklığı $T_İ$ = 30 °C ve çıkış sıcaklığı T_O = 40 °C, toplayıcı alanı A_C = 4 m^2, toplayıcı verim faktörü F' = 0,91, suyun kütlesel debisi m= 0,0604 kg/sn olarak belirlenmiştir. Bu verilere göre toplayıcı ısı kayıp katsayısı U_L'nin değerini False Position yöntemi ile hesaplayınız. Kök bulma işlemine başlamadan önce kökün bulunduğu aralığı belirleyiniz. $0 < U_L < 10$ olarak düşününüz.

PROBLEM 25

$f(x) = x^3-6x^2+11x-6.1$ fonksiyonunun en büyük gerçek kökünü Newton-Raphson yöntemi kullanarak belirleyin

Bölüm 3 Doğrusal Denklem Sistemleri

3.1 Doğrusal denklem sistemleri, genel

Doğrusal denklem sistemleri birden fazla değişkenin doğrusal fonksiyonu olan denklemlerdir. Örneğin

$3x+2y-z=1$

$2x-2y+4z=-2$

$-x+0.5y-z=0$

x,y ve z olarak 3 değişken içeren bir doğrusal denklem sistemidir. Denklem sisteminin çözümü değişkenlerin her denklemi sağlayan çözüm setidir. Bu denklem için bu çözüm x=1,y=-2,z=-2 setidir. En genel formunda Doğrusal denklem sistemleri :

$$\mathbf{a_{11} X_1 + a_{12} X_2 + a_{13} X_3 + \ldots + a_{1n} X_3 = b_1}$$
$$\mathbf{a_{21} X_1 + a_{22} X_2 + a_{23} X_3 + \ldots + a_{2n} X_3 = b_2}$$
$$\mathbf{a_{31} X_1 + a_{32} X_2 + a_{33} X_3 + \ldots + a_{3n} X_3 = b_3}$$
$$\ldots\ldots\ldots$$
$$\mathbf{a_{n1} X_1 + a_{n2} X_2 + a_{n3} X_3 + \ldots + a_{nn} X_3 = b_n}$$

Şeklinde yazılabilir. Bu denklem matris formunda

[A]{X}={B} veya

$$A\vec{x}=\vec{B}$$

şeklinde yazılabilir. Matris formunu açık yazarsak :

$$\begin{bmatrix} a_{11} & a_{12} & .... & a_{1n} \\ a_{21} & a_{22} & ..... & a_{2n} \\ .... & .... & .... & .... \\ a_{n1} & a_{n2} & .... & a_{nn} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ ... \\ X_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ .... \\ b_n \end{bmatrix}$$

Bazı kaynaklarda da bu denklem sistemi kısaltılmış formda

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & ..... & a_{1n} & b_1 \\ a_{21} & a_{22} & ..... & a_{2n} & b_2 \\ .... & .... & .... & .... & .... \\ a_{n1} & a_{n2} & ..... & a_{nn} & b_n \end{array}\right] \quad \text{veya} \quad \begin{bmatrix} a_{11} & a_{12} & ..... & a_{1n} & b_1 \\ a_{21} & a_{22} & ..... & a_{2n} & b_2 \\ ..... & ..... & ..... & ..... & ..... \\ a_{n1} & a_{n2} & ..... & a_{nn} & b_n \end{bmatrix}$$

şeklinde de gösterilebilir. Bu denklem sisteminde A ve B değerleri bilinen değerler X değerleri bilinmeyenlerdir. X değerlerinin saptanması problemini doğrusal denklem sistemi çözümü olarak adlandıracağız. Burada yöntem bazında tanımlamalara gideceğiz, her verilen yöntemi örnek bir java programı ile destekleyeceğiz ve çeşitli örneklerin çözüm sonuçlarını da vereceğiz

3.2 Gauss eleme metodu

Gauss eleme yöntemi en basit formunda matrislerin temel özelliklerine dayanan bir işlemdir. Matrislerin satırlarını değiştirmek, matrisin bir satırını belli bir çarpanla çarpmak veya satırları birbiriyle toplamak(veya çıkarmak) doğrusal denklem sistemindeki bilinmeyenlerin değerini etkilemez. Bu yüzden doğrusal denklem sisteminde
(i numaralı satır)- Sabit * (j numaralı satır) gibi bir işlemi denklem sisteminin değerini değiştirmeden oluşturabiliriz. Gauss eleme metodunda bu manipülasyonu yapmaktaki gayemiz üst üçgen matris dediğimiz matrisin diyagonalinin altının tamamen sıfıra eşitlendiği eşdeğer formunu elde etmektir. Örneğin n=5 için

$$\begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & a'_{14} & a'_{15} \\ 0 & a'_{22} & a'_{23} & a'_{24} & a'_{25} \\ 0 & 0 & a'_{33} & a'_{34} & a'_{35} \\ 0 & 0 & 0 & a'_{44} & a'_{45} \\ 0 & 0 & 0 & 0 & a'_{55} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \end{bmatrix}$$

şekline getirmemiz gerekir. Bunu yapabilmek için sistematik bir eleme uygulayabiliriz. Örneğin a_{21} elemanını 0 a dönüştürmek için

$a'_{21} = 0 = a_{21} - C_1 * a_{11}$ işleminde sabitin değeri
$C_1=(a_{21}/a_{11})$ olacaktır. İkinci satırın her elemanı için C_1 çarpanı kullanılarak yeni değerine çevrilir. Örneğin
$a'_{22} = a_{22} - C_1*a_{12}$
$a'_{23} = a_{23} - C_1*a_{13}$
.....
$a'_{25} = a_{25} - C_1*a_{15}$
$b'_2=b_2 - C_1*b_1$

Bu işlemin sonunda a'_{21} elemanı sıfırlanmış olur. Aynı işlem önce a_{11} in altındaki tüm satırlara uygulanarak birinci sütundaki 0 değerleri oluşturulur. Daha sonra aynı işlem a_{22} nin altındaki kolon için yinelenir.
Bu işlemi aynı değişkenlerle java döngüsü olarak yazarsak

```
for(k=0;k<(n-1);k++)
 {   for(i=k+1;i<n;i++)
       {  C[k]=a[i][k]/a[k][k];
          a[i][k]=0;
          for(j=k+1;j<n;j++)  {   a[i][j]-= C[k]*a[k][j]; }
          b[i]    =b[i]  - C[k]*b[k];
       }
 }
```

Lineer denklem sistemi üst üçgen matris şekline dönüştürüldükten sonra tersten itibaren çözme yöntemiyle bilinmeyenler çözülür. Yukarıda verilen beşli sistemin çözüm seti :

$X_5 = b'_5 / a'_{55}$
$X_4 = (b'_4 – a'_{45}*X_5)/a'_{44}$
$X_3 = (b'_3 – a'_{34}*X_4 – a'_{35}*X_5)/a'_{33}$
$X_2 = (b'_2 – a'_{23}*X_3 – a'_{24}*X_3 – a'_{25}*X_5)/a'_{22}$
$X_1 = (b'_1 – a'_{12}*X_2 – a'_{13}*X_3 – a'_{14}*X_4 – a'_{15}*X_5)/a'_{11}$

Şeklindedir. Bu işlemi n*n matris için java kodu olarak yazacak olursak :

```
 x[n-1]=b[n-1]/a[n-1][n-1];
 for(i=n-2;i>=0;i--)
 {
   toplam=0;
   for(j=i+1;j<n;j++)
   {  toplam+=a[i][j]*x[j];}
   x[i]=(b[i]-toplam)/a[i][i];
 }
```

Doğrusal denklem sisteminin çıktılarını anlaşılır bir şekilde oluşturabilmek için bir String çıktı sınıfı olan matriscikti.java programını oluşturduk. Program kodu aşağıda verilmiştir.

Program 3.2-1 matris cikti.java doğrusal denklem sistemi formatlama sınıfı

```
import java.io.*;
import java.text.*;
import java.util.Locale;

public class matriscikti
{
public static String toString(double left, int w, int d)
// double değişkenini verilen boyutta(w) ve noktadan sonra uzaklığında(d)
// string değişkenine çevirir
{
            NumberFormat df=NumberFormat.getInstance(Locale.US);
            df.setMaximumFractionDigits(d);
   df.setMinimumFractionDigits(d);
   df.setGroupingUsed(false);
   String s = df.format(left);
   while (s.length() < w)
     s = " " + s;
   if (s.length() > w)
   {
```

```
        s = "";
        for (int i=0; i<w; i++)
          s = s + "-";
    }
    return s;
}

public static String toString(double left)
{// double değişkenini sabit boyutta(25) ve noktadan sonra uzaklığında(15)
// string değişkenine çevirir
  return toString(left,25,15);}

public static String toString(double[][] left,int w,int d)
{
//matrisin string eşdeğerini oluşturur
int n,m;
String b;
b="";
n=left.length;
m=left[0].length;
for(int i=0;i<n;i++)
  {
  for(int j=0;j<m;j++)
    {
    b=b+toString(left[i][j],w,d);
    }
    b=b+"\n";
  }
return b;
}

public static String toString(double[][] a,double b[],int w,int d)
{//doğrusal denklem sisteminin string eşdeğerini oluşrurur
int n,m;
String s;
s="";
n=a.length;
m=a[0].length;
for(int i=0;i<n;i++)
  {
  s+=" |";
  for(int j=0;j<m;j++)
    {
    s+=toString(a[i][j],w,d);
    }
    s+="| "+toString(b[i],w,d)+"| \n";
  }
return s;
}

public static String toString(double[][] a,double b[])
{//doğrusal denklem sisteminin string eşdeğerini oluşrurur
```

```
return toString(a,b,25,15);
}

public static String toStringT(double[][] left,int w,int d)
{ return toString(Matrix.Transpose(left),w,d);}

public static String toStringT(double[][] left)
{ return toString(Matrix.Transpose(left),15,10);}


public static String toStringT(double[] left)
{
// returns a vertical string representation
// of a double vector
int n,m;
String b;
b="";
n=left.length;
for(int i=0;i<n;i++)
  {
    b=b+toString(left[i])+"\n";
  }
return b;
}

public static String toString(double[] left)
{
// returns a horizontal string representation
// of a double vector
int n,m;
String b;
b="";
n=left.length;
for(int i=0;i<n;i++)
  {
    b=b+toString(left[i])+"\t ";
  }
  b=b+"\n";
return b;
}}
```

Şimdi Gauss eleme işlemine bir bütün program olarak bakalım :

Program 3.2-2 Gauss Eleme yöntemi

```
import java.io.*;

class SCO10A
{
  public static double[] gauss(double a[][],double b[])
  {
  int n=a.length;
  double x[]=new double[n];
```

```
  double carpan=0;
  double toplam=0;
  //gauss eleme
  int i,j,k;
  String s="";
  s="Orijinal doğrusal denklem sistemi :\n"+matriscikti.toString(a,b);
  for(k=0;k<(n-1);k++)
  {
   for(i=k+1;i<n;i++)
   {  carpan=a[i][k]/a[k][k];
      a[i][k]=0;//bu deyim sadece elimine matrisin doğru değerlerini hesaplamak için konulmuştur
      for(j=k+1;j<n;j++)
      {   a[i][j]-=carpan*a[k][j];  }
      b[i]   =b[i]   -carpan*b[k];
   }
   s+="elimine matris : k="+k+"\n";;
   s+=matriscikti.toString(a,b);
  }
  System.out.println(s);
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
   toplam=0;
   for(j=i+1;j<n;j++)
   {  toplam+=a[i][j]*x[j];}
   x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
  }

  public static void main(String args[]) throws IOException
  {
  double a[][]={{1,2,5},{2,1,2},{3,1,1}};
  double b[]={1,1,2};
  System.out.println("çözüm vektörü : \n"+matriscikti.toStringT(gauss(a,b)));
  }
}
```

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" SCO10A
Orijinal doğrusal denklem sistemi :
|       1.000000000000000       2.000000000000000       5.000000000000000|       1.000000000000000|
|       2.000000000000000       1.000000000000000       2.000000000000000|       1.000000000000000|
|       3.000000000000000       1.000000000000000       1.000000000000000|       2.000000000000000|
elimine matris : k=0
|       1.000000000000000       2.000000000000000       5.000000000000000|       1.000000000000000|
|       0.000000000000000      -3.000000000000000      -8.000000000000000|      -1.000000000000000|
|       0.000000000000000      -5.000000000000000     -14.000000000000000|       -1.000000000000000|
elimine matris : k=1
|       1.000000000000000       2.000000000000000       5.000000000000000|       1.000000000000000|
|       0.000000000000000      -3.000000000000000      -8.000000000000000|      -1.000000000000000|
```

```
|      0.000000000000000       0.000000000000000      -0.666666666666666|       0.666666666666667|

çözüm vektörü :
     -0.000000000000001
      3.000000000000003
     -1.000000000000001

> Terminated with exit code 0.
```

Buradaki örnek doğrusal denklem sistemi :

$$\begin{bmatrix} 1 & 2 & 5 \\ 2 & 1 & 2 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

Çıktının gerçek çözümleri 0,3 ve -1 değerleri olmalıdır, görüldüğü gibi denklem sistemimizin çözümünde ufak ta olsa bir hata mevcuttur. Buradaki elimine işlemlerini daha iyi anlamak için bir de açık hesap olarak yapalım

Sıfır'ıncı kolon(kolon numarası 0 dan başlıyor), birinci satır(satır numarası sıfırdan başlıyor) k=0
çarpan $C_1=a_{10}/a_{00} = 2/1=2$
$a_{10} = a_{10} - C_1 * a_{00} = a_{10} - (a_{10}/a_{00})* a_{00} = 2 - 2*1 = 0$
sıfırıncı kolon elimine prosesi tamamlandı birinci kolon,ikinci satır k=1
çarpan $C_3=a_{21}/a_{11} = -5/-3=(5/3)$
$a_{21} = a_{21} – C_3 * a_{11} = a_{21} - (a_{21}/a_{11})* a_{11} = -5 – (5/3)*(-3) = 0$
$a_{22} = a_{22} - C_3 * a_{12} = a_{11} - (a_{21}/a_{11})* a_{01} = -14 – (5/3)*(-8) = -0.6666666666666$
$b_2 = b_2 - C_3 * b_1 = b_2 - (a_{21}/a_{11})* b_1 = -1 – (5/3)*(-1) = 0.6666666666666$

$$\begin{bmatrix} 1 & 2 & 5 \\ 0 & -3 & -8 \\ 0 & 0 & -0.666666666 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0.666666666 \end{bmatrix}$$

Geriye koyma işlemi :

$X_2 = b_2/a_{22} = 0.6666666666/(-0.6666666666) = -1$
$X_1 = (b_1 - a_{12}*X_2)/a_{11} = (-1 - (-8)*(-1))/(-3) =(-9)/(-3)=3$
$X_0 = (b_0 - a_{01}*X_1 - a_{02}*X_2)/a_{00} = (1 – 2*3-5*(-1))/1=0$

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ -1 \end{bmatrix}$$

İkinci bir örnek daha verelim :

$$\begin{bmatrix} 7 & 1 & 2 \\ -1 & 4 & -1 \\ 3 & 15 & 20 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 47 \\ 19 \\ 87 \end{bmatrix}$$

A B

7	1	2	47
-1	4	-1	19

	3	15	20	87

c	7	1	2	47
-0.14286	0	4.142857	-0.71429	25.71429
0.428571	0	14.57143	19.14286	66.85714

c	7	1	2	47
3.517241	0	4.142857	-0.71429	25.71429
	0	0	21.65517	-23.5862

X_0	**6.165605**
X_1	**6.019108**
X_2	**-1.08917**

Gauss eliminasyon probleminin Matlab örneği aşağıda verilmiştir.:

PROGRAM 3.2-3 Gauss elimination in Matlab(Octave)

```
function x=gauss(a,b)
% Gauss elimination
n=length(b);
carpan=0;
toplam=0;
  %gauss eleme
  for k=1:n
    for i=k+1:n
      carpan=a(i,k)/a(k,k);
       a(i,k)=0;
       for j=k+1:n
          a(i,j)=a(i,j)-carpan*a(k,j);
       end
           b(i)    =b(i)    -carpan*b(k);
    end
  end
  %back substituting
  x(n)=b(n)/a(n,n);
  for i=n-1:-1:1
    toplam=0;
    for j=i+1:n
      toplam=toplam+a(i,j)*x(j);end
    x(i)=(b(i)-toplam)/a(i,i);
  end
end
```

```
>> a=[7 1 2;-1 4 -1;3 15 20]
a =
   7    1    2
  -1    4   -1
   3   15   20
>> b=[47 19 87]'
b =
   47
   19
   87
>> x=gauss(a,b)
x =
   6.1656    6.0191   -1.0892

>> xx=a\b
xx =
   6.1656
   6.0191
  -1.0892
>>
```

3.3 Kısmi pivot işlemi, kısmi pivotlu gauss eleme metodu

Yukardaki basit örnekte de görüldüğü gibi sonuçta belli bir hata oluşmuştur. Gauss eleme prosesi tek başına kullanıldığında bazı denklem sistemlerinde bu hata büyük boyutlara ulaşabilir. Bu durumun oluşmasındaki temel neden Gauss elimine prosesindeki çarpandaki bölüm teriminin üst tarafta kalan terime göre küçük olması ve bunun sonucunda çarpım teriminin yuvarlatma hatası verecek kadar büyük değerlere ulaşmasıdır. Hatta bölünen sayı sıfıra eşit olursa çarpan sonsuza da gidebilir. Bunu engellemenin bir yolu eleme prosesi yapılan kolon için bölünen değerin en büyük değer olacağı şekilde satırların yerlerinin değiştirilmesidir. Bu işleme kısmi pivotlama adı veriyoruz. Eleme prosesinin yapıdığı satır ve sütunları tarıyarak maksimum olan değerleri diagonal terimin üzerinde de toplayabiliriz, ancak sütunların yer değiştirilmesi durumunda değişkenlerin sıraları da değişeceğinden çok daha kompleks bir indeksleme sistemine ihtiyacımız olacaktır. Bu yüzden tam pivotlama daha seyrek olarak kullanılan bir prosestir.

Şimdi kısmi pivotlamayı bir önceki aynı örnekle anlatalım, örnek problemimiz :

$$\begin{bmatrix} 1 & 2 & 5 \\ 2 & 1 & 2 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

Şeklindeydi. Burada gauss eliminasyonu uygulamadan önce her kolon için eleme elemanını alttaki elemanlarla karşılaştırarak alttakiler arasında daha büyük mutlak değere sahip olan varsa eleme elemanıyla bu satır değiştirilecektir. Örneğimizin sıfırıncı sütununda olan kritik eleman a_{00} 1 e eşittir. Bu değer a_{10} (2) ve a_{20} (3) ile karşılaştırıldığında a_{20} değerinin en büyük olduğu görülecektir. Bu yüzden 2. satır ve 0. satır yer değiştirir ve sistem

$$\begin{bmatrix} 3 & 1 & 1 \\ 2 & 1 & 2 \\ 1 & 2 & 5 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

Şeklini alır. Bu işleme java program parçacığı olarak bakarsak :

```
double buyuk, dummy;
for(k=0;k<(n-1);k++)
  { //pivotlama
          p=k;
          buyuk=Math.abs(a[k][k]);
          for(ii=k+1;ii<n;ii++)
          {  dummy=Math.abs(a[ii][k]);
            if(dummy > buyuk) {buyuk=dummy;p=ii;}
          }
          if(p!=k)
          { for(jj=k;jj<n;jj++)
            { dummy=a[p][jj];
              a[p][jj]=a[k][jj];
              a[k][jj]=dummy;
            }
            dummy=b[p];
            b[p]=b[k];
            b[k]=dummy;
          }
}
```

Şeklinde gerçekleştirdiğimizi görürüz. Burada satırları değiştirmek için bir ek değişkene ihtiyaç duyulduğunu not etmede yarar görüyoruz. Kısmi pivotlama işleminden sonra gauss eleme işlemini bölüm 2.1 de anlattığımız gibi gerçekleştiririz.

Program 3.3-1 Pivotlu Gauss Eleme yöntemi

```
import java.io.*;

class SCO10B
{
  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
  int i,j,k,p,ii,jj;
  String s="Orijinal Matris :\n"+matriscikti.toString(a,b);
  for(k=0;k<(n-1);k++)
  { //pivotlama
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
              if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
                a[k][jj]=dummy;
              }
              dummy=b[p];
              b[p]=b[k];
              b[k]=dummy;
            }
              s+="pivotlu Matris : k="+k+"\n"+matriscikti.toString(a,b);
            //gauss elemeyi çözme
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
          b[i]   =b[i]   -carpan*b[k];
            }
  }
  s+="elimine prosesi bitmiş matris : k="+k+"\n"+matriscikti.toString(a,b);
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
```

```
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  System.out.println(s);
  return x;
  }
  public static void main(String args[]) throws IOException
  {
  double a[][]={{1,2,5},{2,1,2},{3,1,1}};
  double b[]={1,1,2};
  System.out.println("sonuç vektörü : \n"+matriscikti.toStringT(pivotlugauss(a,b)));
  }
}
```

Çıktı 3.3-1 kısmi pivotlu gauss eleme yöntemi

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" SCO10B
Orijinal Matris :
|      1.000000000000000      2.000000000000000      5.000000000000000|      1.000000000000000|
|      2.000000000000000      1.000000000000000      2.000000000000000|      1.000000000000000|
|      3.000000000000000      1.000000000000000      1.000000000000000|      2.000000000000000|
pivotlu Matris : k=0
|      3.000000000000000      1.000000000000000      1.000000000000000|      2.000000000000000|
|      2.000000000000000      1.000000000000000      2.000000000000000|      1.000000000000000|
|      1.000000000000000      2.000000000000000      5.000000000000000|      1.000000000000000|
pivotlu Matris : k=1
|      3.000000000000000      1.000000000000000      1.000000000000000|      2.000000000000000|
|      0.000000000000000      1.666666666666667      4.666666666666667|      0.333333333333333|
|      0.000000000000000      0.333333333333333      1.333333333333334|     -0.333333333333333|
elimine prosesi bitmiş matris : k=2
|      3.000000000000000      1.000000000000000      1.000000000000000|      2.000000000000000|
|      0.000000000000000      1.666666666666667      4.666666666666667|      0.333333333333333|
|      0.000000000000000      0.000000000000000      0.400000000000000|     -0.400000000000000|

sonuç vektörü :
      0.000000000000000
      3.000000000000000
     -1.000000000000000
```

Sonuçtan da görüldüğü gibi sonuç vektöründe bir önce görülen hata burada yok olmuştur. Küçük matris sistemlerinin çözümünde gauss eleme yöntemi ideal bir çözümdür, ancak matris boyutu büyüdüğünde toplam hesap miktarı çok fazlalaştığından pratik avantajını yitirir. İkinci olarak aynı kodun Matlab versiyonunu da verelim.

PROGRAM 3.3-2 Gauss elimination with partial pivoting in Matlab(Octave) code

```
function x=gauss1(a,b)
% Gauss elimination
n=length(b);
carpan=0;
toplam=0;
  for k=1:n
```

```
    %pivoting
    p=k;
    buyuk=abs(a(k,k));
    for ii=k+1:n
      dummy=abs(a(ii,k));
       if dummy > buyuk buyuk=dummy;p=ii;end
    end
    if p~=k
      for jj=k:n
        dummy=a(p,jj);
        a(p,jj)=a(k,jj);
        a(k,jj)=dummy;
        end
      dummy=b(p);
      b(p)=b(k);
      b(k)=dummy;
    end
    %gauss eleme
    for i=k+1:n
      carpan=a(i,k)/a(k,k);
       a(i,k)=0;
       for j=k+1:n
          a(i,j)=a(i,j)-carpan*a(k,j);
       end
           b(i)    =b(i)    -carpan*b(k);
    end
  end
  %back substituting
  x(n)=b(n)/a(n,n);
  for i=n-1:-1:1
    toplam=0;
    for j=i+1:n
      toplam=toplam+a(i,j)*x(j);end
    x(i)=(b(i)-toplam)/a(i,i);
  end
end
```

3.4 Gauss-Jordan metodu

Gauss Jordan metodu temel olarak gauss metoduna benzer. Temel ayrılığı pivot elemanı dediğimiz elemenin yapılmasında kullanılan temel elemanın bulunduğu satırın elemeden önce pivot elemanına bölünerek normalize edilmesi ve eleme prosesinin sadece pivot elemanının altındaki satırlara değil pivot elemanı satırı haricindeki tüm satırlar için uygulanmasıdır. Program 2.4-1 de Gauss-Jordan elemesi java programı olarak verilmiştir.

Program 3.3.2 Gauss_Jordan Eleme yöntemi

```
 import java.io.*;

class SCO10C
{

  public static double[] gaussjordan(double a[][],double b[])
  { //pivotsuz
  int n=b.length;
  int np1=n+1;
  double C[][]=new double[n][np1];
  double x[]=new double[n];
  double carpan=0;
  int i,j,k;

  for(i=0;i<n;i++)
  {for(j=0;j<n;j++){C[i][j]=a[i][j];};C[i][n]=b[i];}

  for(k=0;k<n;k++)
```

```
  {
            //gauss jordan eleme
            for(j=k+1;j<np1;j++)
            { C[k][j]/=C[k][k]; }
            for(i=0;i<n;i++)
   {  if(i!=k)
      {  carpan=C[i][k];
         C[i][k]=0;
         for(j=k+1;j<np1;j++)
         {   C[i][j]-=carpan*C[k][j]; }
      }
            }
  }
  for(i=0;i<n;i++)
  {x[i]=C[i][n];}
  return x;
  }

  public static void main(String args[]) throws IOException
  {
  double a[][]={{1,2,5},{2,1,2},{3,1,1}};
  double b[]={1,1,2};
  double x[]=gaussjordan(a,b);
  System.out.println("çözüm vektörü : \n"+matriscikti.toStringT(x));
  }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO10C
çözüm vektörü :
      0.000000000000001
      2.999999999999995
     -0.999999999999998
> Terminated with exit code 0.
```

ikinci bir örnek olarak :

$$\begin{bmatrix} 3 & 1 & -1 \\ 1 & 4 & 1 \\ 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 12 \\ 10 \end{bmatrix}$$

bu örneği programa girmek için veri satırını

double a[][]={{3,1,-1},{1,4,1},{2,1,2}};
double b[]={2,12,10};
Olarak değiştirmemiz kafidir. Çözüm seti :

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO10C
çözüm vektörü :
      1.000000000000000
      2.000000000000000
```

```
3.000000000000000

> Terminated with exit code 0.
```

Olacaktır. Bu denklem sistemini Gauss-Jordan eleme kullanarak elle çözelim

$$\begin{bmatrix} 1 & 0.3333 & -0.3333 \\ 1 & 4 & 1 \\ 2 & 1 & 2 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0.6667 \\ 12 \\ 10 \end{bmatrix}$$

ikinci satırdan birinci satırı çıkaralım

$$\begin{bmatrix} 1 & 0.3333 & -0.3333 \\ 0 & 3.6667 & 1.3333 \\ 2 & 1 & 2 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0.6667 \\ 11.3333 \\ 10 \end{bmatrix}$$ a

birinci satırı 2 ile çarpıp üçüncü satırdan çıkaralım

$$\begin{bmatrix} 1 & 0.3333 & -0.3333 \\ 0 & 3.6667 & 1.3333 \\ 0 & 0.3333 & 2.6667 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0.6667 \\ 11.3333 \\ 8.6667 \end{bmatrix}$$

Şimdi ikinci sütuna geçiyoruz. İkinci sütundakipivot elemanı 3.6667 dir. Tüm ikinci satırı bu değere bölerek normalize edelim

$$\begin{bmatrix} 1 & 0.3333 & -0.3333 \\ 0 & 1 & 0.3636 \\ 0 & 0.3333 & 2.6667 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0.6667 \\ 3.0909 \\ 8.6667 \end{bmatrix}$$

birinci satırı sıfırlayalım (ikinci satırı 0.333 ile çarpıp birinci satırdan çıkaralım):

$$\begin{bmatrix} 1 & 0 & -0.4545 \\ 0 & 1 & 0.3636 \\ 0 & 0.3333 & 2.6667 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} -0.3636 \\ 3.0909 \\ 8.6667 \end{bmatrix}$$

üçüncü satırı sıfırlayalım (ikinci satırı 0.333 ile çarpıp üçüncü satırdan çıkaralım):

$$\begin{bmatrix} 1 & 0 & -0.4545 \\ 0 & 1 & 0.3636 \\ 0 & 0 & 2.5455 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} -0.3636 \\ 3.0909 \\ 7.6365 \end{bmatrix}$$

üçüncü satırı 2.5455 e bölelim

$$\begin{bmatrix} 1 & 0 & -0.4545 \\ 0 & 1 & 0.3636 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} -0.3636 \\ 3.0909 \\ 3.0000 \end{bmatrix}$$

birinci satırı sıfırlayalım (üçüncü satırı -0.4545 ile çarpıp birinci satırla toplayalım)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.3636 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1.0000 \\ 3.0909 \\ 3.0000 \end{bmatrix}$$

ikinci satırı sıfırlayalım (üçüncü satırı 0.3636 ile çarpıp ikinci satırdan çıkaralım)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1.0000 \\ 2.0001 \\ 3.0000 \end{bmatrix}$$

çözüm vektörü

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1.0000 \\ 2.0001 \\ 3.0000 \end{bmatrix}$$

gerçek çözüm :

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1.0000 \\ 2.0000 \\ 3.0000 \end{bmatrix}$$

değerleridir.

Gauss-Jordan elemesi için bir detaylı örnek daha verelim :

$$\begin{bmatrix} 7 & 1 & 2 \\ -1 & 4 & -1 \\ 3 & 15 & 20 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 47 \\ 19 \\ 87 \end{bmatrix}$$

Gauss-Jordan

A				B
7	1	2	X_0	47
-1	4	-1	X_1	19
3	15	20	X_2	87
Normalizasyon				
1	0.142857	0.285714	X_0	6.714286
-1	4	-1	X_1	19
3	15	20	X_2	87
Eliminasyon				
1	0.142857	0.285714	X_0	6.714286
0	4.142857	-0.71429	X_1	25.71429
0	14.57143	19.14286	X_2	66.85714
Normalizasyon				
1	0.142857	0.285714	X_0	6.714286
0	1	-0.17241	X_1	6.206897
0	14.57143	19.14286	X_2	66.85714
Eliminasyon				
1	0	0.310345	X_0	5.827586
0	1	-0.17241	X_1	6.206897
0	0	21.65517	X_2	-23.5862
Normalizasyon				
1	0	0.310345	X_0	5.827586
0	1	-0.17241	X_1	6.206897
0	0	1	X_2	-1.08917

Eliminasyon

1	0	0	X_0	**6.165605**
0	1	0	X_1	**6.019108**
0	0	1	X_2	**-1.08917**

Gauss-Jordan eliminasyonunun Matlab kodunu da verelim

PROGRAM 3.4.1 Gauss_Jordan elimination in Matlab(Octave) code

```
function x=gaussjordan(a,b)
 n=length(b);
 np1=n+1;
 carpan=0;
 C=a;
 for i=1:n
    C(i,n+1)=b(i);
 end
 for k=1:n
  %gauss jordan elimination
  for j=k+1:np1
   C(k,j)=C(k,j)/C(k,k);
  end
  for i=1:n
   if i~=k
     carpan=C(i,k);
      C(i,k)=0;
      for j=k+1:np1
        C(i,j)=C(i,j)-carpan*C(k,j); end
    end
  end
 end
 for i=1:n
  C(i,i)=1;
  x(i)=C(i,np1);
 end
 end
```

3.5 Kısmi pivotlu gauss-jordan metodu

Gauss metodundaki gibi Gauss-Jordan metodunda da çözüm her zaman doğru olarak oluşmıyabilir. Doğru çözüme ulaşmamız pivotlama dediğimiz ek işlemi gerektirir. Kısmi pivotlama işleminde satırlar arasında en büyük mutlak değere sahip olan satırla pivot satırı yer değiştirilir.

Program 3.5-1 Kısmi Pivotlu Gauss_Jordan Eleme yöntemi

```
import java.io.*;

class SCO10C3
{ public static double[][] C(double a[][],double b[])
  {int n=a.length;int m=a[0].length;double c[][]=new double[n][m+1];
  for(int i=0;i<n;i++){for(int j=0;j<m;j++){c[i][j]=a[i][j];};c[i][m]=b[i];}
  return c;
  }

  public static double[][] C(double a[][],double b[][])
  {int n=a.length;int o=b[0].length;int m=n+o;double c[][]=new double[n][m];
  for(int i=0;i<n;i++)
  {for(int j=0;j<n;j++){c[i][j]=a[i][j];};
   for(int j=n;j<m;j++){c[i][j]=b[i][j];}
```

```
}
return c;
}

public static double[][] pivotlugaussjordan(double a[][])
{ //pivotlu
int n=a.length;
int m=a[0].length;
int o=m-n;
double x[][]=new double[n][o];
double carpan=0;
double toplam=0;
double buyuk;
double dummy=0;
int i,j,k,p,ii,jj;
i=0;
for(k=0;k<n;k++)
{ //pivotlama
        p=k;
        buyuk=Math.abs(a[k][k]);
        for(ii=0;ii<n;ii++)
        {  if(i!=k)
          {
          dummy=Math.abs(a[ii][k]);
          if(dummy > buyuk) {buyuk=dummy;p=ii;}
   }
        }
        if(p!=k)
        { for(jj=k;jj<m;jj++)
         { dummy=a[p][jj];
           a[p][jj]=a[k][jj];
           a[k][jj]=dummy;
         }
        }
        //gauss jordan elemeyi çözme
        for(j=k+1;j<m;j++)
        { a[k][j]/=a[k][k]; }
        a[k][k]=1.0;
 for(i=0;i<n;i++)
 {  if(i!=k)
    {
            carpan=a[i][k];
      a[i][k]=0;
      for(j=k+1;j<m;j++)
        {a[i][j]-=carpan*a[k][j]; }
    }
        }
}
for(jj=0;jj<o;jj++)
{for(i=0;i<n;i++) {x[i][jj]=a[i][n+jj];}}
return x;
}
```

```
  public static void main(String args[]) throws IOException
  {
  double a[][]={{3,0,2,8},{0,0,2,-7},{6,2,8,26}};
  //double a[][]=Text.readDouble_Table();
  double x[][]=pivotlugaussjordan(a);
  System.out.println(Matrix.toString(x));
  //Text.print(x,"Şekil 3.5-1 Pivotlu Gauss Jordan Metodu çözümü");
  }
}
```

Çıktı 3.5-1 Kısmi pivotlu Gauss_Jordan Eleme yöntemi çözüm seti

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO10C3
       5.000000000000000
      12.000000000000000
      -3.500000000000000

> Terminated with exit code 0.
```

3.6 LU-alt-üst üçgen matris- Gauss (Dolittle) metodu

Gauss ve Gauss-Jordan eleme metodlarında sol taraftaki matris ile sağ taraftaki vektör aynı anda işleme girmektedir. Bu durumda yeni bir sol taraf vektörü değerlendirmeye alınacağı zaman sağ tarafın tekrar işleme alınması gerekmektedir. Bunu engellemek için sol taraf matrisinin üst üçgen matrisi(U) ve alt üçgen matrisi(L) olmak üzere iki matrisin çarpımı şekline getirilmesi, sonra bu iki alt matris vektörünün çözülmesine gidilebilir. Örneğin N=5 için

$$[L]=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 & 0 \\ l_{41} & l_{42} & l_{43} & 1 & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & 1 \end{bmatrix} \qquad [U]=\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} \\ 0 & 0 & u_{33} & u_{34} & u_{35} \\ 0 & 0 & 0 & u_{44} & u_{45} \\ 0 & 0 & 0 & 0 & u_{55} \end{bmatrix}$$

şeklindedir. Çözüm için :

[A]=[L][U]
[A]{X}={B}
[L][U]{X}={B}
[L]{D}={B}
[U]{X}={D}

Temel matris çarpım işlemleri dizisinden yararlanılır. Buradaki U matrisini Gauss eleme prosesinden sonraki üst üçgen matris olarak hesaplayabiliriz. L matrisi Gauss eleme prosesindeki çarpanlardan oluşur. L matrisinin diagonal elemanları 1 değerine eşit olduğundan L ve U matrisleri giriş A matrisinin aynı yerine yerleştirilebilir, ek bilgisayar alanı kullanılması gerekmez. Program 3.6.1 de Pivot kullanılmadan uygulanan Dolittle metodu verilmiştir. Program 3.6.2 de LU işlemi pivotlama işlemi ile birlikte uygulanmıştır. Pivotlu versiyonumuzda LU ayrıştırma prosesi ve denklem sistemi çözme prosesi ayrı ayrı uygulandığından matris satır değiştirme işlemi bir referans değeri üzerinden taşınmakta ve denklem sistemi çözümünde B vektörü aynı referansa göre yeniden düzenlenmektedir.

Program 3.6.1 Gauss (Dolittle) Eleme yöntemi

```
import java.io.*;
import javax.swing.*;
class SCO10D1
{
  public static double[][] gaussLU(double ai[][])
  {
  int n=ai.length;
  double x[]=new double[n];
  double a[][]=new double[n][n];
  double carpan=0;
  double toplam=0;
  //gauss eleme
  int i,j,k;
  for(i=0;i<n;i++)
      for(j=0;j<n;j++){a[i][j]=ai[i][j];};
  for(k=0;k<(n-1);k++)
  {
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=carpan;///L elemanı
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
            }
  }
  return a;
  }


  public static double[] yerine_koyma(double ai[][],double bi[])
  { //gauss LU çözümüyle LU indirgenmesi yapılmış Matris kullanarak lineer
    //denklem sistemi kökleri
    int n=ai.length;
    double toplam;
    double x[]=new double[n];
     double a[][]=new double[n][n];
    double b[]=new double[n];
    int i,j;
  for(i=0;i<n;i++)
      {for(j=0;j<n;j++){a[i][j]=ai[i][j];};b[i]=bi[i];}
  //System.out.println("a=\n "+Matrix.toString(a));
  //System.out.println("b=\n "+Matrix.toStringT(b));
  for(i=1;i<n;i++)
      {toplam=b[i];for(j=0;j<i;j++){toplam-=a[i][j]*b[j];};b[i]=toplam;}
   //System.out.println("d=\n "+Matrix.toStringT(b));
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=(n-2);i>=0;i--)
      {toplam=0;
       for(j=i+1;j<n;j++)
        {toplam+=a[i][j]*x[j];};
        x[i]=(b[i]-toplam)/a[i][i];
      }
  //System.out.println("d=\n "+Matrix.toStringT(x));
```

```
return x;
}

public static double[][] yerine_koyma(double ai[][],double bi[][])
{
  int n=ai.length;
  int m=bi[0].length;
  double toplam;
  double x[][]=new double[n][m];
  double a[][]=new double[n][n];
  double b[][]=new double[n][m];
  int i,j,k;
for(i=0;i<n;i++)
    {for(j=0;j<n;j++){a[i][j]=ai[i][j];};for(k=0;k<m;k++) {b[i][k]=bi[i][k];}}
//System.out.println("a=\n "+Matrix.toString(a));
//System.out.println("b=\n "+Matrix.toStringT(b));
for(k=0;k<m;k++)
{
for(i=1;i<n;i++)
    {toplam=b[i][k];for(j=0;j<i;j++){toplam-=a[i][j]*b[j][k];};b[i][k]=toplam;}
 //System.out.println("d=\n "+Matrix.toStringT(b));
x[n-1][k]=b[n-1][k]/a[n-1][n-1];
for(i=(n-2);i>=0;i--)
    {toplam=0;
     for(j=i+1;j<n;j++)
      {toplam+=a[i][j]*x[j][k];};
      x[i][k]=(b[i][k]-toplam)/a[i][i];
    }
}//kin sonu
//System.out.println("d=\n "+Matrix.toStringT(x));
return x;
}

public static double [] AXB(double a[][],double b[])
{
//denklem sistemi çözümü b vektörü için
int n=a.length;
double c[][]=gaussLU(a);
double s[]=yerine_koyma(c,b);
return s;
}

public static double [][] AXB(double a[][],double b[][])
{//denklem sistemi çözümü b matrisi için
int n=a.length;
double c[][]=gaussLU(a);
double s[][]=yerine_koyma(c,b);
return s;
}

public static void main(String args[]) throws IOException
{
```

```
  double a[][]={{3,-0.1,-0.2},{0.1,7,-0.3},{0.3,-0.2,10}};
  JPanel y=new JPanel();
  Text.add(y,a,"a matrisi");
  double b[]={7.85,-19.3,71.4};
  Text.addT(y,b,"b matrisi");
  double x[]=AXB(a,b);
  Text.addT(y,x,"çözüm matrisi");
  Text.print(y,"Gauss LU ayrıştırma ve yerine koyma metodu çözümü");
  }
}
```

Program 3.6-2 Kısmi Pivotlu LU Gauss (Dolittle) Eleme yöntemi

```
 import java.io.*;
import javax.swing.*;
class SCO10D
{
  public static void pivot(double a[][],int o[],double s[],int k)
  {// GaussLU (Dolittle) için pivot metodu
    int n=a.length;
    int dummy1=0;
    double buyuk;
    double dummy=0;
    //pivotlama
            int p=k;
            buyuk=Math.abs(a[o[k]][k]);
            for(int ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[o[ii]][k]/s[o[ii]]);
              if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            dummy1=o[p];
            o[p]=o[k];
            o[k]=dummy1;
  }

  public static int[] sirali_indeks(int n)
  {   // sıralı rakamları taşıyan bir referans vektörü oluşturur
              int o[]=new int[n];
              for(int i=0;i<n;i++) {o[i]=i;}
              return o;
  }

  public static double[][] gaussLU(double ai[][],int o[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=ai.length;
  double a[][]=new double[n][n];
  double s[]=new double[n];
  double carpan=0;
  double toplam=0;
  double absaij;

  //gauss eleme
  int i,j,k,p,ii,jj;
```

```
//System.out.println("Orijinal Matris :\n"+Matrix.toString(ai));
for(i=0;i<n;i++)
{for(j=0;j<n;j++){a[i][j]=ai[i][j];}}
for(i=0;i<n;i++)
{ o[i]=i;s[i]=Math.abs(a[i][0]);
  for(j=1;j<n;j++)
    { absaij=Math.abs(a[i][j]);
      if(absaij>s[i]) {s[i]=absaij;}
    }
}
for(k=0;k<(n-1);k++)
{ pivot(a,o,s,k);
          //pivot referansını değiştir
          //System.out.println("pivotlu Matris : k="+k+"\n"+Matrix.toString(a));
          //gauss elemeyi çözme
  for(i=k+1;i<n;i++)
  {  carpan=a[o[i]][k]/a[o[k]][k];
     a[o[i]][k]=carpan;
     for(j=k+1;j<n;j++)
     {   a[o[i]][j]-=carpan*a[o[k]][j]; }
          }
}
return a;
}

public static double[] yerine_koyma(double ai[][],double bi[])
{   int n=ai.length;
          return yerine_koyma(ai,bi,sirali_indeks(n));
}

public static double[][] yerine_koyma(double ai[][],double bi[][])
{   int n=ai.length;
          return yerine_koyma(ai,bi,sirali_indeks(n));
}

public static double[] yerine_koyma(double ai[][],double bi[],int o[])
{ //gauss LU çözümüyle LU indirgenmesi yapılmış Matris kullanarak lineer
  //denklem sistemi kökleri
  int n=ai.length;
  double toplam;
  double x[]=new double[n];
   double a[][]=new double[n][n];
  double b[]=new double[n];
  int i,j;
for(i=0;i<n;i++)
   {for(j=0;j<n;j++){a[i][j]=ai[i][j];};b[i]=bi[i];}
//System.out.println("a=\n "+Matrix.toString(a));
//System.out.println("b=\n "+Matrix.toStringT(b));
for(i=1;i<n;i++)
   {toplam=b[o[i]];for(j=0;j<i;j++){toplam-=a[o[i]][j]*b[o[j]];};b[o[i]]=toplam;}
 //System.out.println("d=\n "+Matrix.toStringT(b));
x[n-1]=b[o[n-1]]/a[o[n-1]][n-1];
```

```
for(i=(n-2);i>=0;i--)
    {toplam=0;
     for(j=i+1;j<n;j++)
      {toplam+=a[o[i]][j]*x[j];};
      x[i]=(b[o[i]]-toplam)/a[o[i]][i];
    }
//System.out.println("d=\n "+Matrix.toStringT(x));
return x;
}

public static double[][] yerine_koyma(double ai[][],double bi[][],int o[])
{
  int n=ai.length;
  int m=bi[0].length;
  double toplam;
  double x[][]=new double[n][m];
  double a[][]=new double[n][n];
  double b[][]=new double[n][m];
  int i,j,k;
for(i=0;i<n;i++)
    {for(j=0;j<n;j++){a[i][j]=ai[i][j];};for(k=0;k<m;k++) {b[i][k]=bi[i][k];}}
//System.out.println("a=\n "+Matrix.toString(a));
//System.out.println("b=\n "+Matrix.toStringT(b));
for(k=0;k<m;k++)
{
for(i=1;i<n;i++)
    {toplam=b[o[i]][k];for(j=0;j<i;j++){toplam-=a[o[i]][j]*b[o[j]][k];};b[o[i]][k]=toplam;}
 //System.out.println("d=\n "+Matrix.toStringT(b));
x[n-1][k]=b[o[n-1]][k]/a[o[n-1]][n-1];
for(i=(n-2);i>=0;i--)
    {toplam=0;
     for(j=i+1;j<n;j++)
      {toplam+=a[o[i]][j]*x[j][k];};
      x[i][k]=(b[o[i]][k]-toplam)/a[o[i]][i];
    }
}//kin sonu
//System.out.println("d=\n "+Matrix.toStringT(x));
return x;
}

public static double [] AXB(double a[][],double b[])
{
//denklem sistemi çözümü b vektörü için
int n=a.length;
int q[]=new int[n];
double c[][]=gaussLU(a,q);
double s[]=yerine_koyma(c,b,q);
return s;
}

public static double [][] AXB(double a[][],double b[][])
{//denklem sistemi çözümü b matrisi için
```

```
  int n=a.length;
  int q[]=new int[n];
  double c[][]=gaussLU(a,q);
  double s[][]=yerine_koyma(c,b,q);
  return s;
  }

  public static void main(String args[]) throws IOException
  {
  double a[][]={{3,-0.1,-0.2},{0.1,7,-0.3},{0.3,-0.2,10}};
  System.out.println("a:\n"+Matrix.toString(a));
  double b[]={7.85,-19.3,71.4};
  System.out.println("b:\n"+Matrix.toStringT(b));
  double x[]=AXB(a,b);
  System.out.println("x:\n"+Matrix.toStringT(x));
  }
}
```

```
      ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO10D
a:
       3.000000000000000      -0.100000000000000      -0.200000000000000
       0.100000000000000       7.000000000000000      -0.300000000000000
       0.300000000000000      -0.200000000000000      10.000000000000000

b:
       7.850000000000000000000000
      -19.300000000000000000000000
       71.400000000000000000000000

x:
       3.000000000000000000000000
      -2.500000000000000000000000
       7.000000000000020000000000

> Terminated with exit code 0.
```

3.7 Craut LU metodu

Gauss (Dolittle) LU ayrıştırmasında L matrisinin köşegen elemanları 1 idi. Craut ayrıştırmasının temel farkı U matrisinin köşegen elemanlarının 1 olmasıdır. Örneğin N=5 için

$$[L]=\begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & l_{55} \end{bmatrix} \qquad [U]=\begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & 1 & u_{23} & u_{24} & u_{25} \\ 0 & 0 & 1 & u_{34} & u_{35} \\ 0 & 0 & 0 & 1 & u_{45} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

şeklindedir. Çözüm için yine :

[A]=[L][U]
[A]{X}={B}
[L][U]{X}={B}
[L]{D}={B}
[U]{X}={D}

Temel matris çarpım işlemleri dizisinden yararlanılabilir. Craut algoritması işlemleri yaparken sırası ile kolon ve satır taramaları yaparak hesaplama işlemini gerçekleştirir. İşlem steplerini şu şekilde sıralayabiliriz :

$l_{i,1} = a_{i,1}$ döngü i=1,2,….,n

$$u_{1,j} = \frac{a_{1,j}}{l_{11}}$$ döngü i=2,3,….,n

döngü j=2,3,….,n – 1

$$l_{i,j} = a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} u_{k,j}$$ döngü k=j,j+2,….,n

$$u_{i,j} = \frac{a_{j,k} - \sum_{i=1}^{j-1} l_{j,i} u_{i,k}}{l_{jj}}$$ döngü k=j+1,j+2,….,n

döngü sonu

$$l_{n,n} = a_{n,n} - \sum_{k=1}^{n-1} l_{n,k} u_{k,n}$$

Algoritmayı Alttaki örnek programda verelim :

Program 3.7.1 LU Craut yöntemi

```
import java.io.*;

class SCO10F
{
 public static void pivot(double a[][],int o[],double s[],int k)
 {// GaussLU için pivot metodu
  int n=a.length;
  int dummy1=0;
  double buyuk;
  double dummy=0;
  //pivotlama
        int p=k;
        buyuk=Math.abs(a[o[k]][k]);
        for(int ii=k+1;ii<n;ii++)
        {  dummy=Math.abs(a[o[ii]][k]/s[o[ii]]);
          if(dummy > buyuk) {buyuk=dummy;p=ii;}
        }
        dummy1=o[p];
        o[p]=o[k];
        o[k]=dummy1;
 }

 public static int[] sirali_indeks(int n)
```

```
{   // sıralı rakamları taşıyan bir vektör oluşturur
            int o[]=new int[n];
            for(int i=0;i<n;i++) {o[i]=i;}
            return o;
}

public static double[] yerine_koyma(double ai[][],double bi[])
{   int n=ai.length;
            return yerine_koyma(ai,bi,sirali_indeks(n));
}

public static double[][] yerine_koyma(double ai[][],double bi[][])
{   int n=ai.length;
            return yerine_koyma(ai,bi,sirali_indeks(n));
}

public static double[] yerine_koyma(double ai[][],double bi[],int o[])
{ //gauss LU çözümüyle LU indirgenmesi yapılmış Matris kullanarak lineer
  //denklem sistemi kökleri
  int n=ai.length;
  double toplam;
  double x[]=new double[n];
   double a[][]=new double[n][n];
  double b[]=new double[n];
  int i,j;
for(i=0;i<n;i++)
    {for(j=0;j<n;j++){a[i][j]=ai[i][j];};b[i]=bi[i];}
//System.out.println("a=\n "+Matrix.toString(a));
//System.out.println("b=\n "+Matrix.toStringT(b));
for(i=1;i<n;i++)
    {toplam=b[o[i]];for(j=0;j<i;j++){toplam-=a[o[i]][j]*b[o[j]];};b[o[i]]=toplam;}
 //System.out.println("d=\n "+Matrix.toStringT(b));
x[n-1]=b[o[n-1]]/a[o[n-1]][n-1];
for(i=(n-2);i>=0;i--)
    {toplam=0;
     for(j=i+1;j<n;j++)
      {toplam+=a[o[i]][j]*x[j];};
      x[i]=(b[o[i]]-toplam)/a[o[i]][i];
    }
//System.out.println("d=\n "+Matrix.toStringT(x));
return x;
}

public static double[][] yerine_koyma(double ai[][],double bi[][],int o[])
{
  int n=ai.length;
  int m=bi[0].length;
  double toplam;
  double x[][]=new double[n][m];
  double a[][]=new double[n][n];
  double b[][]=new double[n][m];
  int i,j,k;
```

```
  for(i=0;i<n;i++)
     {for(j=0;j<n;j++){a[i][j]=ai[i][j];};for(k=0;k<m;k++) {b[i][k]=bi[i][k];}}
  //System.out.println("a=\n "+Matrix.toString(a));
  //System.out.println("b=\n "+Matrix.toStringT(b));
  for(k=0;k<m;k++)
  {
  for(i=1;i<n;i++)
     {toplam=b[o[i]][k];for(j=0;j<i;j++){toplam-=a[o[i]][j]*b[o[j]][k];};b[o[i]][k]=toplam;}
   //System.out.println("d=\n "+Matrix.toStringT(b));
  x[n-1][k]=b[o[n-1]][k]/a[o[n-1]][n-1];
  for(i=(n-2);i>=0;i--)
     {toplam=0;
      for(j=i+1;j<n;j++)
       {toplam+=a[o[i]][j]*x[j][k];};
       x[i][k]=(b[o[i]][k]-toplam)/a[o[i]][i];
     }
  }//kin sonu
  //System.out.println("d=\n "+Matrix.toStringT(x));
  return x;
  }

public static double[][] crautLU(double c[][])
{
//craut LU ayrıştırma, pivotsuz

int n=c.length;
double sum;
int i,j,k;
double d[][]=new double[n][n];
for(i=0;i<n;i++)
{for(j=0;j<n;j++) d[i][j]=c[i][j];}

for(j=1;j<=n;j++)
{
  for(i=1;i<j;i++)
    {
    sum=d[i-1][j-1];
    for(k=1;k<i;k++)
            { sum-=d[i-1][k-1]*d[k-1][j-1];}
    d[i-1][j-1]=sum;
    }
  for(i=j;i<=n;i++)
    {
    sum=d[i-1][j-1];
    for(k=1;k<j;k++)
      {sum-=d[i-1][k-1]*d[k-1][j-1];}
    d[i-1][j-1]=sum;
    }
  if(j != n)
  { for(i=j+1;i<=n;i++)
       d[i-1][j-1]/=d[j-1][j-1];
  }
```

```
}
return d;
}

  public static void main(String args[]) throws IOException
  {
  double a[][]={{2,-6,1},{-1,7,-1},{1,-3,2}};
  double b[]={12,-8,16};
  double c[][]= crautLU(a);
  double x[]=yerine_koyma(c,b);
  System.out.println("x:\n"+Matrix.toStringT(x));
  }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO10F
x:
       3.666666666666666500000000
       0.333333333333333700000000
       6.666666666666667000000000

> Terminated with exit code 0.
```

3.8 Jacobi, Gauss-Seidel ve rahatlamalı interpolasyon iteratif metodları

İteratif methodlar bilhassa büyük matrislerin çözülmesinde çok önemli olabilir. Gauss eleme metodu ve LU metodu toplam olarak $n^3/3$, Gauss-Jordan metodu $n^3/2$ işleme gerek duyar n çok büyüdüğünde toplam işlem sayısı da çok büyür. İterativ metodlar eleme işlemi yapmadan bir önceki çözüm setini kullanarak, bir sonraki iterativ çözüme ulaşır. Çözüm sabit hale geldiğinde sonuç elde edilmiş olur. İteratif yöntemlerin en basiti Jacobi metodudur. Burada her denklem için bir x değeri seçilerek değerlendirilir. Diğer x değerleri bir önceki iterasyondan alınır.

$$x_i^{k+1} = \frac{1}{A_{ij}}(b_i - \sum_{j \neq i} A_{ij} x_j^k) \qquad , \qquad 1 \leq i \leq n$$

Bu denklemin daha çabuk bir şekilde çözüme ulaşması için tüm değerleri bir önceki iterasyondan almak yerine hesaplanan tüm değerler formüle koyulur. Bu iteratif yönteme Gauss-Seidel metodu adı verilir.

$$x_i^{k+1} = \frac{1}{A_{ij}}(b_i - \sum_{j=1}^{i-1} A_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} A_{ij} x_j^k), \quad 1 \leq i \leq n$$

İterativ yöntemin dönüşmesini biraz daha çabuklaştırmanın yolu eski iterasyonla yeni iterasyon arasında rahatlama katsayısı ismi verilen α gibi bir ağırlık katsayısıyla hesaplamaya gitmektir.

$$x_i^{k+1} = (1-\alpha)x_i^k + \frac{\alpha}{A_{ij}}(b_i - \sum_{j=1}^{i-1} A_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} A_{ij} x_j^k), \quad 1 \leq i \leq n$$

α birden büyük veya küçük olabilir. Bu yönteme rahatlamalı interpolasyon (relaxation) adı verilir. α bire eşit olduğunda yöntem normal Gauss-Seidel formülüne dönüşür.

Aşağıda bu metodun java dilinde yazılmış programı görülmektedir.

Program 3.7-1 Gauss_seidel rahatlamalı interpolasyon formülü

```
 import java.io.*;

class SCO10E
{
  public static double [] gauss_seidel(double a[][],double b[])
  {
  double lambda=1.2;
  int n=a.length;
  //ilk x tahmin değerleri 0 olarak alındı
  double x[]=new double[n];
  return gauss_seidel(a,b,x,lambda);
  }

  public static double [] gauss_seidel(double a[][],double b[],double x[])
  {
  double lambda=1.2;
  int n=a.length;
  return gauss_seidel(a,b,x,lambda);
  }

  public static double [] gauss_seidel(double a[][],double b[],double x[],double lambda)
  {
  //denklem sistemi çözümü b iteratif gauss_seidel metodu metodu
  //yakınsama faktörü lambda ile birlikte verilmiş formül
  int n=a.length;
  int imax=500;//maksimum iterasyon sayısı
  double es=1.0e-20;
  double dummy=0;
  double ea=1e-1;
  double sum=0;
  double old=0;
  int iter=0;
  int sentinel=0;
  System.out.print("iter = "+(iter+1)+" ");
  for(int i=0;i<n;i++)
  {  dummy=a[i][i];
     for(int j=0;j<n;j++)
       {a[i][j]/=dummy;}
     b[i]/=dummy;
  }
  for(int i=0;i<n;i++)
  {  sum=b[i];
     for(int j=0;j<n;j++)
       {if(i!=j)sum-=a[i][j]*x[j];}
     x[i]=sum;
     System.out.print(x[i]+ "  ");
  }
   System.out.println();
  iter++;
```

```
  do
  {
  sentinel=1;
  System.out.print("iter = "+(iter+1)+" ");
  for(int i=0;i<n;i++)
    {
            old=x[i];
            sum=b[i];
            for(int j=0;j<n;j++)
               {if(i!=j) sum-=a[i][j]*x[j];}
               x[i]=lambda*sum+(1.0-lambda)*old;
               if(sentinel==1 && x[i]!=0.0 ) {ea=Math.abs((x[i]-old)/x[i])*100.0;}
               if(ea<es) sentinel=0;
             System.out.print(x[i]+ "  ");
    }
  System.out.println();
  iter++;
  }while(sentinel==1 && iter<imax);
  if(iter>=imax) System.out.println("Maksimum iterasyon sayısı aşıldı sonuç geçerli olmayabilir iter = "+iter);
  return x;
  }

  public static void main(String args[]) throws IOException
  {
  double a[][]={{2.04, -1 ,0,0},
              {-1,2.04,-1,0},
              {0,-1,2.04,-1},
              {0,0,-1,2.04}};
  double b[]={40.8,0.8,0.8,200.8};
  double x[]={0,0,0};
  double lambda=0.0;
  System.out.println("x:\n"+Matrix.toStringT(gauss_seidel(a,b)));
  }
}
```

Çıktı 3.7-1 Gauss_seidel rahatlamalı interpolasyon formülü

```
      ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO10E
iter = 1 20.0  10.196078431372548  5.390234525182622  101.07364437508953
iter = 2 25.997693194925027  16.894859443200577  68.78577887042823  138.36514104876343
iter = 3 28.738613974662393  54.45890625553084  100.13934264020509  149.35011451977974
iter = 4 50.28692794361507  78.06484850408215  114.21622736834831  155.43364025389602
iter = 5 59.86311353132531  87.25723082840335  120.3866786923888  157.84661235650833
iter = 6 63.355160133972205  91.10257667335522  122.83395192614785  158.80359042643235
iter = 7 64.91871895753215  92.69282047925837  123.84286308870615  159.20567196689362
iter = 8 65.54144472570437  93.33455814791921  124.25509215567808  159.36774334525546
iter = 9 65.79439231834101  93.5974910021921  124.42264883206882  159.4338918204012
iter = 10 65.8984691846801  93.70468886823738  124.49110593278539  159.46093100814647
iter = 11 65.94071137967421  93.74836652779935  124.51901265811696  159.47193889138072
iter = 12 65.9579556815942  93.76619042368198  124.53039118311936  159.47643056473527
iter = 13 65.96499146584704  93.7734575911262  124.53503244329465  159.4782623831086
iter = 14 65.96785911337543  93.77642116216893  124.53692500856316  159.47900929312132
```

```
iter = 15 65.96902886095369  93.77762980845847  124.53769682274604  159.47931391946165
iter = 16 65.96950587984365  93.77812268689047  124.53801158036379  159.47943814573344
iter = 17 65.96970040455507  93.77832368904475  124.53813993967911  159.47948880595868
iter = 18 65.96977973617413  93.77840565975178  124.53819228601148  159.47950946587386
iter = 19 65.96981208791327  93.77843908800537  124.53821363331488  159.4795178911281
iter = 20 65.96982528124403  93.77845272037474  124.53822233892691  159.47952132702548
iter = 21 65.9698306616187  93.7784582797754  124.53822588915632  159.4795227282163
iter = 22 65.96983285577944  93.7784605469483  124.53822733697143  159.47952329963405
iter = 23 65.96983375057839  93.778461471522  124.5382279274034  159.47952353266342
iter = 24 65.9698341154855  93.77846184857141  124.53822816818686  159.4795236276949
iter = 25 65.96983426429783  93.77846200233552  124.53822826638049  159.47952366644955
iter = 26 65.96983432498486  93.77846206504192  124.53822830642477  159.4795236822541
iter = 27 65.96983434973356  93.77846209061416  124.53822832275519  159.4795236886993
iter = 28 65.96983435982632  93.77846210104278  124.53822832941489  159.47952369132776
iter = 29 65.96983436394225  93.77846210529563  124.53822833213076  159.4795236923996
iter = 30 65.96983436562074  93.77846210703  124.53822833323831  159.47952369283672
iter = 31 65.96983436630528  93.77846210773727  124.53822833368997  159.47952369301498
iter = 32 65.9698343665844  93.77846210802569  124.53822833387413  159.47952369308769
iter = 33 65.96983436669824  93.77846210814332  124.53822833394929  159.47952369311736
iter = 34 65.96983436674465  93.77846210819129  124.53822833397993  159.47952369312944
iter = 35 65.96983436676359  93.77846210821086  124.53822833399241  159.47952369313435
iter = 36 65.96983436677131  93.77846210821883  124.53822833399751  159.47952369313637
iter = 37 65.96983436677446  93.77846210822212  124.53822833399961  159.47952369313725
iter = 38 65.96983436677576  93.77846210822344  124.53822833400048  159.4795236931375
iter = 39 65.96983436677627  93.77846210822399  124.53822833400078  159.4795236931377
iter = 40 65.96983436677651  93.77846210822418  124.53822833400095  159.4795236931377
x:
      65.969834366776510000000000
      93.778462108224180000000000
     124.538228334000950000000000
     159.479523693137700000000000

> Terminated with exit code 0.
```

3.9 Cholesky simetrik matris LU metodu

Cholesky simetrik matris LU metodu temel olarak Dolittle LU metoduna benzer ancak matris simetrik olduğundan ayrıştırma alt üçgen matris ve alt üçgen matrisin transpozu(üst üçgen matris) şeklinde ayrılabilir.

$[A]=[L][L]^T$

Ayrıştırma işlemini

$$l_{ki}=\frac{a_{ki}-\sum_{j=1}^{i-1} l_{ij}l_{kj}}{l_{ii}} \qquad k=1,2,.....,k-1$$

ve

$$l_{kk}=\sqrt{a_{kk}-\sum l_{kj}^2}$$

şeklinde gerçekleştirebiliriz. Cholesky programı altta verilmiştir.

Program 3.8-1 Cholesky simetrik matris LU formülü

```
 import java.io.*;

class SCO10I
{

public static double[][] choleskyLU(double c[][])
{
//cholesky LU
//sadece simetrik matrisler içindir
double factor=0;
double buyuk;
int i,j,k,p,ii,jj;
int n=c.length;
double d[][]=new double[n][n];
double dummy;
double toplam;
for(i=0;i<n;i++)
{for(j=0;j<n;j++) d[i][j]=c[i][j];}
for(k=1;k<=n;k++)
{
 for(i=1;i<=(k-1);i++)
    {toplam=0;;
     for(j=1;j<=(i-1);j++)
     { toplam+=d[i-1][j-1]*d[k-1][j-1]; }
    d[k-1][i-1]=(d[k-1][i-1]-toplam)/d[i-1][i-1];
    }
 toplam=0.0;
 for(j=1;j<=(k-1);j++)
 { toplam+=d[k-1][j-1]*d[k-1][j-1]; }
 d[k-1][k-1]=Math.sqrt((d[k-1][k-1]-toplam));
 for(j=k+1;j<=n;j++) d[k-1][j-1]=0.0;
}
return d;
}

  public static double[] substitutecholesky(double c[][],double b[])
  {
  //ileriye doğru yerine koyma cholesky simetrik LU metoduyla
  //indirgenmiş olmalıdır
  int i,j;
  double U[][]=T(c);
  double L[][]=c;
  System.out.println("L*L(T) = \n"+Matrix.toString(Matrix.multiply(L,U)));
  int n=b.length;
  double toplam=0;
  double x[]=new double[n];
  for(i=1;i<n;i++)
    {toplam=b[i];
     for(j=0;j<i;j++)
       {toplam-=U[i][j]*b[j];}
```

```
   b[i]=toplam/L[i][i];;
    }
  //geriye yerine koyma
  x[n-1]=b[n-1]/L[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=(i+1);j<n;j++)
    {  toplam+=U[i][j]*x[j];}
    x[i]=(b[i]-toplam)/U[i][i];
  }
  return x;
  }

  public static double[][] T(double [][] left)
  {
  //transpose matrix (if A=a(i,j) T(A)=a(j,i)
  int i,j;
  int n=left.length;
  int m=left[0].length;
  double b[][];
  b=new double[m][n];
  for(i=0;i<n;i++)
  {
  for(j=0;j<m;j++)
    {
    b[j][i]=left[i][j];
    }
  }
  return b;
  }

  public static void main(String args[]) throws IOException
  {
  double a[][]={{6,15,55},{15,55,225},{55,225,979}};
  double b[]={1,1,1};
  double L[][]=choleskyLU(a);
  System.out.println("Matris L : \n"+Matrix.toString(L));
  double x[]=substitutecholesky(L,b);
  System.out.println("Cholesky : \n"+Matrix.toStringT(x));
  }
}
```

Program 3.8-1 Cholesky simetrik matris LU formülü

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO10I
Matris L :
       2.449489742783178       0.000000000000000       0.000000000000000
       6.123724356957946       4.183300132670377       0.000000000000000
      22.453655975512470      20.916500663351886       6.110100926607781
L*L(T) =
       5.999999999999999      15.000000000000000      55.000000000000000
```

```
      15.000000000000000      55.000000000000000     225.000000000000000
      55.000000000000000     225.000000000000000     979.000000000000000

Cholesky :
       0.35467686189243460000000000
      -0.07678571428571453000000000
       0.02678571428571433600000000


> Terminated with exit code 0.
```

İkinci bir problemi elde adım adım takip ederek çözelim:

$$\begin{bmatrix} 4 & -1 & 1 \\ -1 & 4.25 & 2.75 \\ 1 & 2.75 & 3.5 \end{bmatrix}$$

$$l_{11} = \sqrt{a_{11}} = \sqrt{4} = 2$$

$$l_{21} = \frac{a_{21}}{l_{11}} = \frac{-1}{2} = -0.5$$

$$l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{4.25 - (-0.5)^2} = 2$$

$$l_{31} = \frac{a_{31}}{l_{11}} = \frac{1}{2} = 0.5$$

$$l_{32} = \frac{a_{32} - l_{21} - l_{31}}{l_{22}} = \frac{2.75 - (-0.5) * 0.5}{2} = 1.5$$

$l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2} = \sqrt{3.5 - 0.5^2 - 1.5^2} = 1$ bu şekilde L matrisini oluşturmuş olduk :

$$[L] = \begin{bmatrix} 2 & 0 & 0 \\ -0.5 & 2 & 0 \\ 0.5 & 1.5 & 1 \end{bmatrix}$$

U matrisi L nin transpozundan ibaret olduğundan

$$[U] = [L]^T = \begin{bmatrix} 2 & -0.5 & 0.5 \\ 0 & 2 & 1.5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[L]\{D\} = \{B\}$$

$$\begin{bmatrix} 2 & 0 & 0 \\ -0.5 & 2 & 0 \\ 0.5 & 1.5 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}$$

d_1=1/2=0.5

d_2=(1-(-0.5)*0.5)/2=0.625

d_3=(1-0.5*0.5-1.5*0.625)/1=-0.1875

$$\begin{bmatrix} 2 & -0.5 & 0.5 \\ 0 & 2 & 1.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 0.5 \\ 0.625 \\ -0.1875 \end{Bmatrix}$$

x_3=-0.1875
x_2=(0.625-1.5*(-0.1875))/2=0.453125
x_1=(0.5-(-0.5)*0.453125-0.5*(-0.1875))/2=0.41015625

3.10 3lü ve 5li bant matris

Üçlü köşegen sistem – yani bant genişliği 3 olan sistem genel olarak

$$\begin{bmatrix} f_1 & g_1 & & & & & & \\ e_2 & f_2 & g_2 & & & & & \\ & e_3 & f_3 & g_3 & & & & \\ & & e_4 & f_4 & g_4 & & & \\ & & & e_5 & f_5 & g_5 & & \\ & & & & . & . & . & \\ & & & & & e_{n-1} & f_{n-1} & g_{n-1} \\ & & & & & & e_n & f_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ . \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ . \\ r_{n-1} \\ r_n \end{bmatrix}$$

şeklide ifade edilebilir. Tüm LU işlemlernde olduğu gibi ayrıştırma, yerine koyma ve geriye doğru yerine koyma proseslerinden oluşur.

Ayrıştırma :

```
for(int k=1;k<n;k++)
  {e[k]=e[k]/f[k-1];
   f[k]=f[k]-e[k]*g[k-1];
  }
```

İleriye doğru yerine koyma :

```
for(int k=1;k<n;k++)
  {r[k]=r[k]-e[k]*r[k-1];
  }
```

Geriye doğru yerine koyma :

```
x[n-1]=r[n-1]/f[n-1];
for(int k=(n-2);k>=0;k--)
  {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
```

Tüm işlem java programı olarak alttaki programda tanımlanmıştır.

Program 3.9-1 3-lü bant matris için Thomas algoritması

```
import java.io.*;

class SCO10H
{

 public static double [] thomas(double a[][],double r[])
 {
 //
 int n=a.length;
 double f[]=new double[n];
 double e[]=new double[n];
```

```
  double g[]=new double[n];
  double x[]=new double[n];
  for(int i=0;i<n;i++)     {f[i]=a[i][i];}
  for(int i=0;i<(n-1);i++) {g[i]=a[i][i+1];}
  for(int i=0;i<(n-1);i++) {e[i+1]=a[i+1][i];}
  for(int k=1;k<n;k++)
     {e[k]=e[k]/f[k-1];
      f[k]=f[k]-e[k]*g[k-1];
     }
  for(int k=1;k<n;k++)
     {r[k]=r[k]-e[k]*r[k-1];
     }
  x[n-1]=r[n-1]/f[n-1];
  for(int k=(n-2);k>=0;k--)
     {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
  return x;
  }

  public static double [] thomas(double f[],double e[],double g[],double r[])
  {
  int n=f.length;
  double x[]=new double[n];
  for(int k=1;k<n;k++)
     {e[k]=e[k]/f[k-1];
      f[k]=f[k]-e[k]*g[k-1];
     }
  for(int k=1;k<n;k++)
     {r[k]=r[k]-e[k]*r[k-1];
     }
  x[n-1]=r[n-1]/f[n-1];
  for(int k=(n-2);k>=0;k--)
     {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
  return x;
  }

  public static void main(String args[]) throws IOException
  {
  double f[]={2.04,2.04,2.04,2.04};
  double g[]={-1,-1,-1,0};
  double e[]={0,-1,-1,-1};
  double b[]={40.8,0.8,0.8,200.8};
  System.out.println("thomas : \n"+Matrix.toStringT(thomas(f,e,g,b)));
  double a[][]={{2.04,-1,0,0},{-1,2.04,-1,0 },{0,-1,2.04,-1},{0,0,-1,2.04}};
  double d[]={40.8,0.8,0.8,200.8};
  System.out.println("thomas : \n"+Matrix.toStringT(thomas(a,d)));
  }
}
```

Çıktı 3.9-1 3-lü bant matris için Thomas algoritması

```
thomas :
      65.969834366776620
```

93.778462108224330
124.538228334001000
159.479523693137740

thomas :

65.969834366776620
93.778462108224330
124.538228334001000
159.479523693137740

3ü bant matris hesabını elde hesaplanan bir örnekle de gösterelim :

$$\begin{bmatrix} f_1 & g_1 & 0 \\ e_2 & f_2 & g_2 \\ 0 & e_3 & f_3 \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 41 \\ 25 \\ 105 \end{bmatrix}$$

$$\begin{bmatrix} 0.8 & -0.4 & 0 \\ -0.4 & 0.8 & -0.4 \\ 0 & -0.4 & 0.8 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 41 \\ 25 \\ 105 \end{bmatrix}$$

$$e_k = \frac{e_k}{f_{k-1}}$$

$$f_k = f_k - e_k g_{k-1}$$

$$e_2 = \frac{e_2}{f_1} = \frac{-0.4}{0.8} = -0.5$$

$$f_2 = f_2 - e_2 g_1 = 0.8 - (-0.5)(-0.4) = 0.6$$

$$e_3 = \frac{e_3}{f_2} = \frac{-0.4}{0.6} = -0.6666666666$$

$$f_3 = f_3 - e_3 g_2 = 0.8 - (-0.66666666\ )(-0.4) = 0.5333333$$

Matrisin yeni formu:

$$\begin{bmatrix} 0.8 & -0.4 & 0 \\ -0.5 & 0.6 & -0.4 \\ 0 & -0.66666 & 0.53333 \end{bmatrix}$$

LU şekline ayrılmış matris :

$$[A] = [L][U] = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0 & -0.66666 & 1 \end{bmatrix}\begin{bmatrix} 0.8 & -0.4 & 0 \\ 0 & 0.6 & -0.4 \\ 0 & 0 & 0.53333 \end{bmatrix}$$

İleriye doğru yerine koyma

$$[L]\{D\} = \{B\}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0 & -0.66666 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 41 \\ 25 \\ 105 \end{Bmatrix}$$

$d_1=41$

$d_2=25-(-0.5)*41=45.5$

$d_3=105-(-0.66666)*45.5=135.3333333$

Geriye doğru yerine koyma

$$[U]\{X\} = \{D\}$$

$$\begin{bmatrix} 0.8 & -0.4 & 0 \\ 0 & 0.6 & -0.4 \\ 0 & 0 & 0.53333 \end{bmatrix} \begin{Bmatrix} X_1 \\ X_2 \\ X_3 \end{Bmatrix} = \begin{Bmatrix} 41 \\ 45.5 \\ 135.333333 \end{Bmatrix}$$

$X_3=135.33333/0.53333=253.75$

$X_2=(45.5-(-0.4)*\ 253.75)/0.6=245$

$X_1=(41-(-0.4)*245)/0.8=173.75$

burada örnek problem olarak 3 lü band matriste aldığımız matrisi çözdük, aynı sonucu görmüş olduk.

İkinci bir örnekte 5li band matris çözelim :

$$A=\begin{bmatrix} 6 & -4 & 1 & 0 & 0 & . & .. & .. \\ -4 & 6 & -4 & 1 & 0 & ... & ... & \\ 1 & -4 & 6 & -4 & 1_3 & 0 & .. & \\ ... & .... & .... & ... & ... & .. & & \\ & & ... & ... & ... & ... & & .. \\ & & & . & . & . & . & \\ & & ... & 0 & 1 & -4 & 6 & -4 \\ & & ... & 0 & 0 & 1 & -4 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ . \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ . \\ 0 \\ 4 \end{bmatrix}$$

matrisini alalım. Programın metodlarında bir değişiklik olmayacağından sadece ana program listesini veriyoruz. Bu örnekte tüm A matrisini vermek yerine bantları verdik. Bu şekilde giriş çok büyük boyuttaki matrisler için oldukça önemli bir hafıza kazanımı sağlar.

Beşli köşegen sistem – yani bant genişliği 5 olan sistemi şu şekilde yazarsak :

$$\begin{bmatrix} d_1 & c_1 & f_1 & & & & & \\ a_1 & d_2 & c_2 & f_2 & & & & \\ e_1 & a_2 & d_3 & c_3 & f_3 & & & \\ & e_2 & a_3 & d_4 & c_4 & f_4 & & \\ & & e_3 & a_4 & d_5 & c_5 & f_5 & \\ & & & . & . & . & . & f_{n-2} \\ & & & & e_{n-3} & a_{n-2} & d_{n-1} & c_{n-1} \\ & & & & & e_{n-2} & a_{n-1} & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ . \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ . \\ b_{n-1} \\ b_n \end{bmatrix}$$

buradaki LU ayrıştırma ve yerine koyma prosesi program 3.9.2 de görüldüğü gibi oluşturulur.

Program 3.9-2 5-lü bant matris için band algoritması

```
import java.io.*;

class SCO10J
```

```
{
  public static double [] besliband(double A[][],double b[])
  {
  //
  int n=A.length;
  double f[]=new double[n-1];
  double c[]=new double[n-1];
  double d[]=new double[n];
  double a[]=new double[n-1];
  double e[]=new double[n-2];
  double x[]=new double[n];

  for(int i=0;i<n;i++)     {d[i]=A[i][i];}
  for(int i=0;i<(n-1);i++) {c[i]=A[i][i+1];}
  for(int i=0;i<(n-1);i++) {a[i]=A[i+1][i];}
  for(int i=0;i<(n-2);i++) {f[i]=A[i][i+2];}
  for(int i=0;i<(n-2);i++) {e[i]=A[i+2][i];}
  int i;
  double xmult;
  for(i=1;i<(n-1);i++)
  {xmult=a[i-1]/d[i-1];
   d[i]+=-xmult*c[i-1];
   c[i]+=-xmult*f[i-1];
   b[i]+=-xmult*b[i-1];
   xmult=e[i-1]/d[i-1];
   a[i]+=-xmult*c[i-1];
   d[i+1]+=-xmult*f[i-1];
   b[i+1]+=-xmult*b[i-1];
   //System.out.println(i+"d="+d[i]+"c="+c[i]+"b="+b[i]);
  }
  xmult=a[n-2]/d[n-2];
  d[n-1]+=-xmult*c[n-2];
  x[n-1]=(b[n-1]-xmult*b[n-2])/d[n-1];
  x[n-2]=(b[n-2]-c[n-2]*x[n-1])/d[n-2];
  for(i=n-3;i>=0;i--)
  {x[i]=(b[i]-f[i]*x[i+2]-c[i]*x[i+1])/d[i];}
  return x;
  }

   public static double [] besliband(double f[],double c[],double d[],double a[],double e[],double b[])
  {
  int n=d.length;
  double x[]=new double[n];
  int i;
  double xmult;
  for(i=1;i<(n-1);i++)
  {xmult=a[i-1]/d[i-1];
   d[i]+=-xmult*c[i-1];
   c[i]+=-xmult*f[i-1];
   b[i]+=-xmult*b[i-1];
   xmult=e[i-1]/d[i-1];
   a[i]+=-xmult*c[i-1];
   d[i+1]+=-xmult*f[i-1];
   b[i+1]+=-xmult*b[i-1];
   //System.out.println(i+"d="+d[i]+"c="+c[i]+"b="+b[i]);
  }
  xmult=a[n-2]/d[n-2];
  d[n-1]+=-xmult*c[n-2];
```

```
  x[n-1]=(b[n-1]-xmult*b[n-2])/d[n-1];
  x[n-2]=(b[n-2]-c[n-2]*x[n-1])/d[n-2];
  for(i=n-3;i>=0;i--)
  {x[i]=(b[i]-f[i]*x[i+2]-c[i]*x[i+1])/d[i];}
  return x;
  }

  public static void main(String args[]) throws IOException
  {
  double d[]={6,6,6,6,6,6,6,6,6,7};
  double c[]={-4,-4,-4,-4,-4,-4,-4,-4,-4,-4};
  double f[]={1,1,1,1,1,1,1,1,1};
  double a[]={-4,-4,-4,-4,-4,-4,-4,-4,-4};
  double e[]={1,1,1,1,1,1,1,1,1};
  double b[]={3,0,0,0,0,0,0,0,0,4};
  double x[]=besliband(f,c,d,a,e,b);
  System.out.println("x:\n"+Matrix.toStringT(x));
  }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO10J
x:
      2.3872180451127853000000000
      4.1954887218045190000000000
      5.4586466165413650000000000
      6.2105263157894880000000000
      6.4849624060150520000000000
      6.3157894736842240000000000
      5.7368421052631680000000000
      4.7819548872180520000000000
      3.4849624060150410000000000
      1.8796992481203020000000000


> Terminated with exit code 0.
```

3.11 Konjuge gradyan metodu

Vektör x^* **pozitif ve tanımlı** Ax = b denklem sisteminin çözüm seti ise, bu çözüm setinin geçerli olması için x^* vektörünün $g(x) = x^T Ax - 2b^T x$ fonksiyonunu minimum yapması gerekir. x vektörünü çözüm x* vektöründen farklı bir vektör olarak tanımlıyalım. Ek olarak 0 dan farklı bir v vektörü ve gerçek bir sayı olan t sayısını tanımlayalım.

$g(x+tv)=g(x)+2t\,[v^T(Ax-b)] + t^2\,[v^T(Av)]$

şeklinde yazılabilir. Bu denklem için x ve v vektörleri bilindiğinde t nin hangi değeri için fonksiyonun minimum olacağını irdeleyelim(değişken boyutunu bir boyuta indirgeyelim). Bu fonksiyonun sadece t'nin fonksiyonu olduğunu kabul edersek :

h(t)=g(x+tv) yazabiliriz ve minimum h fonksiyonunun t'ye göre türevinde oluşacaktır.
$h'(t)= 2[v^T(Ax-b)]-2t\,[v^T(Av)] = 0$. Bu denklemden

$t = \dfrac{v^T(Ax-b)}{v^T Av}$ olduğu görülebilir. Bu özellikten yararlanarak çözüme gitmek için önce

r = b – A x tanımını yapalım. Bu tanımı bir önceki t tanımında yerine koyarsak :

$t = \frac{v^T r}{v^T A v}$ tanım, şeklini alır.

t ye iteratif proses açısında baktığımızda iteratif olarak değişen

$x_k = x_{k-1} + t_k v_k$

ve $r_k = b - A x_k$ için t_k değeri

$$t_k = \frac{v_k^T r_k}{v_k^T A v_k}$$

şeklinde yazılabilir. Yeni bulunan x_k vektörü d

A x_k = b denklemini sağlamalıdır.

Çözüme ulaştığımızda A x_k = A(x_{k-1} + $t_k v_k$) = b olacağını da biliyoruz. Ancak arama yönü v_k i hala bilmiyoruz. v_k yönü olarak **en dik yamaç yönü** seçilebilir. Bu yön

$\nabla g(x) = 2(Ax - b) = -2r$ olur.

Bu durumda $v_{k+1} = -\nabla g(x_k) = r_k$ olarak seçilebilir. bu yön g(x_k) deki en büyük negatif değişimi verir. Bu metoda en dik yamaç metodu adı verilir. Alternatif bir seçim ise şöyle yapılabilir : 0 dan farklı öyle v vektörleri seçebilirizki bu vektörler için $v_i^T A v_j = 0 \quad i \neq j$ olur. Bu tür vektörlere ortogonal vektör adını veriyoruz. Eğer ortagonal v vektörleri bulabilirsek, bu bizim çözümümüzü oldukça basitleştirecektir (sette i ve j nin eşit olmadığı terimler yok olacağından matris çözümleri yerlerini çarpma ve bölme işlemlerine bırakacaktır.). Hestenes ve Stiefel'in önerdiği Ortoganal set

$$s_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$$

$v_{k+1} = r_k + s_k v_k$

şeklinde bir önceki v vektör değeri kullanılarak türetilir. Özet olarak metodu şu şekilde uygulayabiliriz :

$r_0 = b - A x_0$ ve $v_0 = r_0$ (en dik yamaç medodu değeri) alınır.
sonra k=1 den n e kadar

$$t_k = \frac{r_{k-1}^T r_k}{v_k^T A v_k}$$

$x_k = x_{k-1} + t_k v_k$

$r_k = b - A x_k$

$$s_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$$

$v_{k+1} = r_k + s_k v_k$

şeklinde iteratif proses oluşturulur. Görüldüğü gibi yöntem uzun bir yöntemdir, ve küçük matrisler için kullanımı Gauss yöntemi gibi direk metodlara göre daha zordur. Ancak denklem sistemi çok büyük boyutlara ulaştığında tercih edilen bir yöntem haline gelir.

Program 3.11-1 konjuge gradyen metodu ile matris sistemi çözümü

```
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class conjugate_gradient1
```

```
{
public static double VT_V(double [] left)
{
//multiplies a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*left[i];
  }
return tot;
}

public static double[][] V_VT(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
for(int i=0;i<n;i++)
  {  for(int j=0;j<n;j++)
        {aa[i][j]=left[i]*left[j];}
  }
return aa;
}
public static double  VT_X(double [] left,double [] right)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*right[i];
  }
return tot;
}
public static double[] multiply(double[] left,double[][] right)
{
//multiplication of one vector with one matrix
int ii,jj,i,j,k;
int m2=right[0].length;
int n2=right.length;
int m1=left.length;
double[] b;
b=new double[m1];
 if(n2 != m1)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n2;ii++)
   {
    b[ii]=0;
   }
   return b;
 }
 for(i=0;i<m2;i++)
   {
    b[i]=0;
    for(k=0;k<m1;k++)
            b[i]+=right[i][k]*left[k];
```

```
   }
return b;
//end of multiply of a vector and a matrix
}
public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right.length;
double[] b;
b=new double[m2];
 if(n1 != m2)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
   {
     b[ii]=0;
   }
   return b;
 }
 for(i=0;i<m1;i++)
   {
     b[i]=0;
     for(k=0;k<n1;k++)
              b[i]+=left[i][k]*right[k];
   }
return b;
//end of multiply of a matrix and a vector
}
public static double[] multiply(double left,double[] right)
{
//multiplying a vector with a constant
int i;
int n=right.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {
  b[i]=left*right[i];
  }
return b;
}
public static double[] substract(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
```

```
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]-right[i];
  }
return b;
//end of vector substraction method
}
public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
}
public static double[] equals(double[] left)
{
//addition of two vectors
int n1=left.length;
double b[];
b=new double[n1];
for(int i=0;i<n1;i++)
  {
    b[i]=left[i];
  }
return b;
//end of vector equal method
}
public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}
public static double VT_G_V(double V[],double G[][])
{
  double x1[]=multiply(V,G);
  return VT_X(V,x1);
}

public static double[] conjugate_gradient(double A[][],double b[])
```

```
{int n=A.length;
 double x[]=new double[n];
 return conjugate_gradient(x,A,b);
}

public static double[] conjugate_gradient(double x[],double A[][],double b[])
{ // iterative system of equation solution
double eps=1.0e-15;
int maxiter=500;
int n=x.length;
double r[]=new double[n];
double re[]=new double[n];
double v[]=new double[n];
double t;
double s;
r=substract(b,multiply(A,x));
v=equals(r);
int k=0;
do
  {
  re=equals(r);
  t=VT_V(re)/VT_G_V(v,A);
  x=add(x,multiply(t,v));
  r=substract(r,multiply(multiply(t,v),A));
  s=VT_V(r)/VT_V(re);
  v=add(r,multiply(s,v));
  k++;
  }while(norm(r)>eps && k<maxiter);
System.out.println("k = "+k+" norm(r)="+norm(r));
return x;
}

  public static void main(String args[]) throws IOException
  {
  double A[][]={{0.2,0.1,1,1,0},{0.1,4,-1,1,-1},{1,-1,60,0,-2},{1,1,0,8,4},{0,-1,-2,4,700}};
  double b[]={1,2,3,4,5};
  double x[]=conjugate_gradient(A,b);
  System.out.println("Konjuge gradyan metodu : \n"+Matrix.toStringT(x));
  }
}
```

Program 3.11-1 konjuge gradyen metodu ile matris sistemi çözümü

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" conjugate_gradient1
k = 9 norm(r)=5.161554370481792E-18
Konjuge gradyan metodu :
       7.859713075445860
       0.422926408295008
      -0.073592239024046
      -0.540643016894627
       0.010626162854036

> Terminated with exit code 0.
```

Konunun başında konjuge gradyan yönteminin başarıyle uygulanabilmesi için denklem sisteminin pozitif ve tanımlı olması gerektiğini belirtmiştik. Pozitif ve tanımlı olmıyan denklem sistemlerinin çözümleri normal olarak iteratif yöntemlerle yapılamaz, ancak denklem sistemi ön şartlandırması veya iyileştirmesi diyebileceğimiz bir yöntem

kullanarak, matris sistemini pozitif tanımlı şekle dönüştürebiliriz. Pozitif ve tanımlı bir C matrisi alalım. Bu matrisi kullanarak

$A^* = C^{-1}A\,C^{-1}$ $x^* = C^{-1}x$ ve $b^* = C^{-1}b$ tanımlarını yaparak denklem sistemimizi daha kararlı olan

$A^* x^* = b^*$ formuna dönüştürebiliriz. Bu dönüşümden sonra Konjuge gardyan yöntemini kullanarak çözüme ulaşabiliriz. Pozitif ve Tanımlı C matrisi olarak örneğin A matrisinin diagonal elemanlarının kare köklerinin alındığı, diğer elemanların 0 olduğu bir matris seçilebilir. Ters matris alınması işlemi de sayıların 1 e bölünmesiyle oluşturulabilir. Eğer bu programı kullanarak şartlandırılmamış konjuge gradyan metodu kullanmak istersek C değerini I birim matrise eşit almalıyız. Bu metodun prosedurunu verirsek :

Girdi : A matrisi ,b ve x vektörleri, C^{-1} pozitif ve tanımlı matrisi

Adım 1 : $r_0 = b - A\,x$

$w_0 = C^{-1}r_0$

$v_0 = C^{-1}\,w_0$

$$\alpha_0 = \sum_{j=1}^{n} w_j^2$$

Adım 2 : k=1

Adım 3 : while(k<=N) do adım 4-5

Adım 4 : $u=Av_k$

$$t_k = \frac{\alpha_{k-1}}{\sum_{j=1}^{n} v_j u_j}$$

$x_k=x_{k-1} + t_k v_k$

$r_k=r_{k-1} + t_k u_k$

$w_k = C^{-1}r_k$

$$\beta_k = \sum_{j=1}^{n} w_j^2$$

Adım 5 : $s_k= \beta_k/\alpha_k$

$v_{k+1} = C^{-1}\,w_k + s_k v_k$

$\alpha_k = \beta_k$

k=k+1

ön şartlandırmalı konjuge gradyan programımızın bu şekilde oluşturulmuş bir versiyonu aşağıda verilmiştir.

Program 3.11-2 şartlandırılmış konjuge gradyen metodu ile matris sistemi çözümü

```
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class conjugate_gradient2
{
public static double VT_V(double [] left)
{
//multiplies a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*left[i];
  }
return tot;
```

```
}

public static double[][] V_VT(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
for(int i=0;i<n;i++)
  {
  for(int j=0;j<n;j++)
    {
            aa[i][j]=left[i]*left[j];
            }
  }
return aa;
}

public static double  VT_X(double [] left,double [] right)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*right[i];
  }
return tot;
}
public static double[][] multiply(double left,double[][] right)
{
//multiplying a matrix with a constant
int i,j;
int n=right.length;
int m=right[0].length;
double b[][];
b=new double[n][m];
for(i=0;i<n;i++)
  {
  for(j=0;j<m;j++)
    b[i][j]=right[i][j]*left;
  }
return b;
//end of multiplying a matrix with a constant double
}
public static double[] multiply(double[] left,double[][] right)
{
//multiplication of one vector with one matrix
int ii,jj,i,j,k;
int m2=right[0].length;
int n2=right.length;
int m1=left.length;
double[] b;
```

```
b=new double[m1];
 if(n2 != m1)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n2;ii++)
   {
     b[ii]=0;
   }
   return b;
 }
 for(i=0;i<m2;i++)
   {
     b[i]=0;
     for(k=0;k<m1;k++)
             b[i]+=right[i][k]*left[k];
   }
return b;
//end of multiply of a vector and a matrix
}

public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right.length;
double[] b;
b=new double[m2];
 if(n1 != m2)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
   {
     b[ii]=0;
   }
   return b;
 }
 for(i=0;i<m1;i++)
   {
     b[i]=0;
     for(k=0;k<n1;k++)
             b[i]+=left[i][k]*right[k];
   }
return b;
//end of multiply of a matrix and a vector
}
public static double[] multiply(double left,double[] right)
{
//multiplying a vector with a constant
int i;
int n=right.length;
```

```
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {
  b[i]=left*right[i];
  }
return b;
}
public static double[] substract(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]-right[i];
  }
return b;
//end of vector substraction method
}
public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
}
```

```
public static double[] equals(double[] left)
{
//addition of two vectors
int n1=left.length;
double b[];
b=new double[n1];
for(int i=0;i<n1;i++)
  {
    b[i]=left[i];
  }
return b;
//end of vector equal method
}
public static double norm(double v[])
{return Math.sqrt(normkare(v));}

public static double normkare(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return total;
}

public static double VT_G_V(double V[],double G[][])
{
  double x1[]=multiply(V,G);
  return VT_X(V,x1);
}

public static double[][] I(int n)
{
//unit matrix
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
    b[i][i]=1.0;
return b;
}

public static double[][] D(double A[][])
{
//Diagonal elements of matrix
int n=A.length;
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
    b[i][i]=A[i][i];
return b;
}
```

```
public static double[][] invsqrtD(double A[][])
{
//square root of Diagonal elements of a matrix
int n=A.length;
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
   b[i][i]=1.0/Math.sqrt(A[i][i]);
return b;
}

public static double[] conjugate_gradient(double x[],double A[][],double b[])
{   int n=A.length;
            double C[][]=new double[n][n];
            C=invsqrtD(A);
   return conjugate_gradient(x,A,b,C);
}

public static double[] conjugate_gradient(double A[][],double b[])
{   int n=A.length;
            double C[][]=new double[n][n];
            C=invsqrtD(A);
            double x[]=new double[n];
            //başlangıç x değerleri 0 olarak alındı
   return conjugate_gradient(x,A,b,C);
}

public static double[] conjugate_gradient(double x[],double A[][],double b[],double C[][])
{ // iterative system of equation solution
double eps=1.0e-15;
int maxiter=500;
int n=x.length;
double r[]=new double[n];
double re[]=new double[n];
double v[]=new double[n];
double w[]=new double[n];
double u[]=new double[n];
double t;
double alfa,beta,s;
int k=0;
r=substract(b,multiply(A,x));
w=multiply(C,r);
v=multiply(C,w);
alfa=normkare(w);
do
  {
  k++;
  if(norm(v)<eps) return x;
  re=equals(r);
  u=multiply(A,v);
  t=alfa/VT_X(v,u);
  x=add(x,multiply(t,v));
```

```
  r=substract(r,multiply(t,u));
  w=multiply(C,r);
  beta=normkare(w);
  if(beta<eps)
  {if(norm(r)<eps) return x;}
  s=beta/alfa;
  v=add(multiply(C,w),multiply(s,v));
  alfa=beta;
  }while(norm(r)>eps && beta>eps && k<maxiter );
  System.out.println("k="+k);
return x;
}
  public static void main(String args[]) throws IOException
  {
  double A[][]={{0.2,0.1,1,1,0},{0.1,4,-1,1,-1},{1,-1,60,0,-2},{1,1,0,8,4},{0,-1,-2,4,700}};
  double b[]={1,2,3,4,5};
  double x[]=conjugate_gradient(A,b);
  System.out.println("Konjuge gradyan metodu : \n"+Matrix.toStringT(x));
  }
}
```

Çıktı 3.11-2 şartlandırılmış konjuge gradyen metodu ile matris sistemi çözümü

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" conjugate_gradient2
k=5
Konjuge gradyan metodu :
       7.859713075445862
       0.422926408295008
      -0.073592239024046
      -0.540643016894627
       0.010626162854036

> Terminated with exit code 0.
```

3.12Matris temel işlemleri toplama ve çarpma tanımları

Matrislerin temel işlemlerinin yapıldığı Matrix.java programı matrislerle ilgili tüm işlemleri barındıran bir kütüphanedir. Bu kitabın ekinde de bütün olarak listelenecektir. Bu bölümde matrisle ilgilili işlemleri alt bölümler olarak aldığımızdan Matrix kütüphanesinden alınmış temel işlemler küçük program parçacıkları halinde (ilgili bölümün programları şeklinde) verilecektir. Bu alt bölümde matrisin toplanması ve çarpmasını tanımlıyalım. Matris çıktısını almak için formatlı çıktı metodları da programımıza eklenmiştir. Matris toplama işlemi :

$c_{ij} = a_{ij} + b_{ij}$

şeklinde özetlenebilir

matris çarpma işlemi ise :

şeklinde gerçekleştirilir.

Program 3.1-1 matris toplama ve çarpma işlemleri algoritması

```
import javax.swing.*;
```

```
import java.text.*;
import java.util.Locale;

public class matris1
{
public static String toString(double left, int w, int d)
// converts a double to a string with given width and decimals.
{
            NumberFormat df=NumberFormat.getInstance(Locale.US);
            df.setMaximumFractionDigits(d);
   df.setMinimumFractionDigits(d);
   df.setGroupingUsed(false);
   String s = df.format(left);
   while (s.length() < w)
     s = " " + s;
   if (s.length() > w)
   {
      s = "";
      for (int i=0; i<w; i++)
        s = s + "-";
   }
   return s;
}

public static String toString(double[][] left)
{
//return a string representation of a matrix
return toString(left,25,15);
}

public static String toString(double[][] left,int w,int d)
{
//return a string representation of a matrix
int n,m;
String b;
b="";
n=left.length;
m=left[0].length;
for(int i=0;i<n;i++)
  {
  for(int j=0;j<m;j++)
    {
    b=b+toString(left[i][j],w,d);
    }
    b=b+"\n";
  }
return b;
}

public static double[][] carp(double[][] left,double[][] right)
{
//multiplication of two matrices
```

```
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right[0].length;
int n2=right.length;
double[][] b;
b=new double[m1][n2];
 if(n1 != m2)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
   {
   for(jj=0;jj<m2;jj++)
     b[ii][jj]=0;
   }
   return b;
 }
 for(i=0;i<m1;i++)
   {
   for(j=0;j<n2;j++)
     {
     for(k=0;k<n1;k++)
                b[i][j]+=left[i][k]*right[k][j];
     }
   }
return b;
//iki matrisin çarpımı
}
public static double[][] topla(double[][] left,double[][] right)
{
//addition of two matrices
int n1=left.length;
int m1=left[0].length;
int n2=right.length;
int m2=right[0].length;
int nMax,mMax;
int i,j;
if(m1>=m2) mMax=m1;
else       mMax=m2;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[][];
b=new double[nMax][mMax];
for(i=0;i<n1;i++)
  {
  for(j=0;j<m1;j++)
    {
    b[i][j]=b[i][j]+left[i][j];
    }
  }
for(i=0;i<n2;i++)
  {
```

```
  for(j=0;j<m2;j++)
    {
    b[i][j]=b[i][j]+right[i][j];
    }
  }
return b;
//end of matrix addition method
}
  public static void main(String args[])
  {
  double a[][]={{3,-0.1,-0.2},{0.1,7,-0.3},{0.3,-0.2,10}};
  double b[][]={{3,0,2},{1,2,-7},{6,8,26}};
  double c[][]=carp(a,b);
  double d[][]=topla(a,b);
  String s="a=\n"+toString(a);
  s+="b=\n"+toString(b);
  s+="a*b=\n"+toString(b);
  s+="a+b=\n"+toString(b);
  String s1="matris çarpma ve toplama işlemleri";
  System.out.println(s1+"\n"+s);
  }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" matris1
matris çarpma ve toplama işlemleri
a=
      3.000000000000000      -0.100000000000000      -0.200000000000000
      0.100000000000000       7.000000000000000      -0.300000000000000
      0.300000000000000      -0.200000000000000      10.000000000000000
b=
      3.000000000000000       0.000000000000000       2.000000000000000
      1.000000000000000       2.000000000000000      -7.000000000000000
      6.000000000000000       8.000000000000000      26.000000000000000
a*b=
      3.000000000000000       0.000000000000000       2.000000000000000
      1.000000000000000       2.000000000000000      -7.000000000000000
      6.000000000000000       8.000000000000000      26.000000000000000
a+b=
      3.000000000000000       0.000000000000000       2.000000000000000
      1.000000000000000       2.000000000000000      -7.000000000000000
      6.000000000000000       8.000000000000000      26.000000000000000


> Terminated with exit code 0.
```

3.13Norm tanımı

Normlar sayısal metodlar kullanılırken matrisleri değerlendirmek için önemli araçlardır. Çeşitli norm tanımları mevcuttur. Genel tanımlamaları burada verelim :

Sonsuz normu :

$$\|A\|_{\infty} = \sum_{i=1}^{n} maksimum(\sum_{j=1}^{n} |a_{ij}|)$$

norm

$$\|A\| = \sum_{j=1}^{n} maksimum(\sum_{i=1}^{n} |a_{ij}|)$$

Öklit normu

$$\|A\|_e = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2}$$

m normu

$$\|A\|_m = \sqrt[m]{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^m}$$

Norm tanımlarını vektörler için de yapabiliriz.

$$\|x\|_\infty = \sum_{i=1}^{n} \max(|x_i|)$$

$$\|x\| = \sum_{i=1}^{n} \max(|x_i|)$$

$$\|x\|_e = \|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

$$\|x\|_m = \sqrt[m]{\sum_{i=1}^{n} x_i^m}$$

Temel norm tanımları program 3.2 de verilmiştir.

Program 3.2 Matris norm tanımlama metodları

```
public class norm
{
//**************norm methods definition*****************
public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}

public static double norm(double v[],int p)
  {
   //p vector norm
  if(p==0) p=1;
  double total=0;
  double x,y;
  for(int i=0;i<v.length;i++)
    {
    x=Math.abs(v[i]);
    total+=Math.pow(x,p);
    }
  return Math.pow(total,(1.0/p));
  }

public static double norminf(double v[])
    {
     //infinite vector norm
    double x;
    double max=0;
    for(int i=0;i<v.length;i++)
      {
      x=Math.abs(v[i]);
      if(x>max) max=x;
      }
    return max;
```

```
    }

public static double[] vek(double mat[][],int n)
{ double a[]=new double[mat.length];
  for(int i=0;i<mat.length;i++) {a[i]=mat[i][n];}
  return a;
}

public static double norminf(double a[][])
       {
         //infinite matrix norm
         double x;
         double max = 0;
         double total;
         int i,j;
         int n=a.length;
         for(i=0;i<n;i++)
         {
         total=0;
         for(j=0;j<n;j++)
         {total+=Math.abs(a[i][j]);}
         x=total;
         if(x>max) max=x;
         }
        return max;
       }

public static double norm(double a[][])
      {
      //matrix norm
              double x;
              double max = 0;
              double total;
              int i,j;
              int n=a.length;
              for(j=0;j<n;i++)
                {
                total=0;
                for(i=0;i<n;j++)
                {total+=Math.abs(a[i][j]);}
                x=total;
                if(x>max) max=x;
                }
              return max;
     }

public static double normE(double a[][])
            {
            //Euclidian matrix norm
            double x;
            double total;
            int i,j;
            total=0;
            int n=a.length;
            for(j=0;j<n;i++)
            {for(i=0;i<n;j++) {total+=a[i][j]*a[i][j];}}
            return Math.sqrt(total);
           }
}
```

3.14 BİRİM MATRİS, TRANSPOSE MATRİS VE TERS MATRİS

Birim matris diagonal elemanları 1 diğer elemanları 0 olan matristir.

$$a_{ii} = 1 \qquad a_{ij, i \neq j} = 0$$

örneğin n=4 için

$$[I] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transpose matris satır ve sütunlarının yer değiştirdiği matristir. Genelde matris işaretinin üzerinde bir T sembolü ile gösterilir.
Java dilinde matrisin transposunu almak için:

```
public static double[][] Transpose(double [][] left)
{
//transpose matrix (if A=a(i,j) Transpose(A)=a(j,i)
int i,j;
int n=left.length;
int m=left[0].length;
double b[][];
b=new double[m][n];
for(i=0;i<n;i++)
  {
  for(j=0;j<m;j++)
    {
    b[j][i]=left[i][j];
    }
  }
return b;
}
```

şeklinde bir metod kullanılabilir.

Ters matris matrisin kendi değeri ile çarpıldığında birim matrise eşit olan matristir. Matrisin bölme işlemi olarakta düşünebiliriz.
$[A][A]^{-1} = [I]$
Ters matrisleri bir önceki bölümde gördüğümüz lineer denklem sistemi çözüm yöntemlerini kullanarak elde edebiliriz.

Program 3.3-1 Ters matris programı(kısmi pivotlu gauss LU metodu kullanarak hesaplar

```
import java.io.*;

class SCO10G
{
  public static void pivot(double a[][],int o[],double s[],int k)
  {// GaussLU için pivot metodu
    int n=a.length;
    int dummy1=0;
    double buyuk;
    double dummy=0;
    //pivotlama
                int p=k;
                buyuk=Math.abs(a[o[k]][k]);
                for(int ii=k+1;ii<n;ii++)
                {  dummy=Math.abs(a[o[ii]][k]/s[o[ii]]);
                  if(dummy > buyuk) {buyuk=dummy;p=ii;}
                }
                dummy1=o[p];
                o[p]=o[k];
                o[k]=dummy1;
  }
```

```
public static int[] sirali_indeks(int n)
{   // sıralı rakamları taşıyan bir vektör oluşturur
                int o[]=new int[n];
                for(int i=0;i<n;i++) {o[i]=i;}
                return o;
}

public static double[][] gaussLU(double ai[][],int o[])
{ //kısmi pivotlu gauss eleme yöntemi
int n=ai.length;
double a[][]=new double[n][n];
double s[]=new double[n];
double carpan=0;
double toplam=0;
double absaij;

//gauss eleme
int i,j,k,p,ii,jj;
//System.out.println("Orijinal Matris :\n"+Matrix.toString(ai));
for(i=0;i<n;i++)
{for(j=0;j<n;j++){a[i][j]=ai[i][j];}}
for(i=0;i<n;i++)
{ o[i]=i;s[i]=Math.abs(a[i][0]);
  for(j=1;j<n;j++)
    { absaij=Math.abs(a[i][j]);
      if(absaij>s[i]) {s[i]=absaij;}
    }
}
for(k=0;k<(n-1);k++)
{ pivot(a,o,s,k);
                //pivot referansını değiştir
                //System.out.println("pivotlu Matris : k="+k+"\n"+Matrix.toString(a));
                //gauss elemeyi çözme
  for(i=k+1;i<n;i++)
  {  carpan=a[o[i]][k]/a[o[k]][k];
     a[o[i]][k]=carpan;
     for(j=k+1;j<n;j++)
     {   a[o[i]][j]-=carpan*a[o[k]][j]; }
                }
}
return a;
}

public static double[] yerine_koyma(double ai[][],double bi[])
{   int n=ai.length;
                return yerine_koyma(ai,bi,sirali_indeks(n));
}

public static double[][] yerine_koyma(double ai[][],double bi[][])
{   int n=ai.length;
                return yerine_koyma(ai,bi,sirali_indeks(n));
}

public static double[] yerine_koyma(double ai[][],double bi[],int o[])
{ //gauss LU çözümüyle LU indirgenmesi yapılmış Matris kullanarak lineer
  //denklem sistemi kökleri
  int n=ai.length;
  double toplam;
  double x[]=new double[n];
   double a[][]=new double[n][n];
  double b[]=new double[n];
  int i,j;
for(i=0;i<n;i++)
    {for(j=0;j<n;j++){a[i][j]=ai[i][j];};b[i]=bi[i];}
//System.out.println("a=\n "+Matrix.toString(a));
```

```
//System.out.println("b=\n "+Matrix.toStringT(b));
for(i=1;i<n;i++)
   {toplam=b[o[i]];for(j=0;j<i;j++){toplam-=a[o[i]][j]*b[o[j]];};b[o[i]]=toplam;}
 //System.out.println("d=\n "+Matrix.toStringT(b));
x[n-1]=b[o[n-1]]/a[o[n-1]][n-1];
for(i=(n-2);i>=0;i--)
   {toplam=0;
    for(j=i+1;j<n;j++)
     {toplam+=a[o[i]][j]*x[j];};
     x[i]=(b[o[i]]-toplam)/a[o[i]][i];
   }
//System.out.println("d=\n "+Matrix.toStringT(x));
return x;
}

public static double[][] yerine_koyma(double ai[][],double bi[][],int o[])
{
  int n=ai.length;
  int m=bi[0].length;
  double toplam;
  double x[][]=new double[n][m];
  double a[][]=new double[n][n];
  double b[][]=new double[n][m];
  int i,j,k;
for(i=0;i<n;i++)
   {for(j=0;j<n;j++){a[i][j]=ai[i][j];};for(k=0;k<m;k++) {b[i][k]=bi[i][k];}}
//System.out.println("a=\n "+Matrix.toString(a));
//System.out.println("b=\n "+Matrix.toStringT(b));
for(k=0;k<m;k++)
{
for(i=1;i<n;i++)
   {toplam=b[o[i]][k];for(j=0;j<i;j++){toplam-=a[o[i]][j]*b[o[j]][k];};b[o[i]][k]=toplam;}
 //System.out.println("d=\n "+Matrix.toStringT(b));
x[n-1][k]=b[o[n-1]][k]/a[o[n-1]][n-1];
for(i=(n-2);i>=0;i--)
   {toplam=0;
    for(j=i+1;j<n;j++)
     {toplam+=a[o[i]][j]*x[j][k];};
     x[i][k]=(b[o[i]][k]-toplam)/a[o[i]][i];
   }
}//kin sonu
//System.out.println("d=\n "+Matrix.toStringT(x));
return x;
}

public static double [] AXB(double a[][],double b[])
{
//denklem sistemi çözümü b vektörü için
int n=a.length;
int q[]=new int[n];
double c[][]=gaussLU(a,q);
double s[]=yerine_koyma(c,b,q);
return s;
}

public static double [][] AXB(double a[][],double b[][])
{//denklem sistemi çözümü b matrisi için
int n=a.length;
int q[]=new int[n];
double c[][]=gaussLU(a,q);
double s[][]=yerine_koyma(c,b,q);
return s;
}

public static double[][] I(int n)
{
```

```
  //birim matris
  double b[][];
  b=new double[n][n];
  for(int i=0;i<n;i++)
    b[i][i]=1.0;
  return b;
  }

  public static double[][] inverse(double a[][])
  {
  int n=a.length;
  int q[]=new int[n];
  double c[][]=gaussLU(a,q);
  double s[][]=yerine_koyma(c,I(n),q);
  return s;
  }

  public static void main(String args[]) throws IOException
  {
  double a[][]={{3,-0.1,-0.2},{0.1,7,-0.3},{0.3,-0.2,10}};
  System.out.println("Matris a : \n"+Matrix.toString(a));
  System.out.println("Ters Matris a : \n"+Matrix.toString(inverse(a)));
  }
}
```

Çıktı 3.3-1 Ters matris çıktısı

```
Matris a :
      3.000000000000000     -0.100000000000000     -0.200000000000000
      0.100000000000000      7.000000000000000     -0.300000000000000
      0.300000000000000     -0.200000000000000     1.10.000000000000000

Ters Matris a :
      0.332488721339843      0.004944070205797      0.006798096532971
     -0.005181765888768      0.142902644602169      0.004183444020290
     -0.010078296957971      0.002709730785869      0.099879725984417
```

Aşağıda ikinci bir ters matris programı veriyoruz. Bunda Gauss eleme metodu kullandık ancak tam pivot uyguladık. Tam pivot hem satırların hem de sütunların diagonal elemana göre büyükten küçüğe sıralanmasını gerektirir. Tabi böyle bir sıralama yapıldığında bilinmeyen X değerlerinin yerleri de değişeceğinden bunların yer sıra değişimlerini de göz önüne alan bir işlem gerekmektedir.

Program 3.3-2 Tam pivotlu Gauss eleme kullanan ters matris metodu

```
public class tersmatris
{
public static double[][] inv(double[][] a)
{
// INVERSION OF A MATRIX
// inversion by using gaussian elimination
// with full pivoting
int n=a.length;
int m=a[0].length;
double b[][];
b=new double[n][n];
int indxc[];
int indxr[];
double ipiv[];
indxc=new int[n];
indxr=new int[n];
ipiv=new double[n];
int i,j,k,l,ll,ii,jj;
int icol=0;
int irow=0;
double big,dum,pivinv,temp;
```

```
if(n!=m)
{
  System.out.println("Matrix must be square ");
                for(ii=0;ii<n;ii++)
                  for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
                return b;
}
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
     b[i][j]=a[i][j];
for(i=0;i<n;i++)
{
 big=0.0;
 for(j=0;j<n;j++)
 {
  if(ipiv[j] != 1)
    for(k=0;k<n;k++)
    {
              if(ipiv[k] == 0)
              {
                if(Math.abs(b[j][k]) >= big)
                {
                  big=Math.abs(b[j][k]);
                  irow=j;
                  icol=k;
                }
              }
              else if(ipiv[k] > 1 )
                {
                System.out.println("error : inverse of the matrix : singular matrix-1");
                for(ii=0;ii<n;ii++)
                  for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
                return b;
                }
    }
 }
 ++ ipiv[icol];
 if(irow != icol)
  for(l=0;l<n;l++)
  {
  temp=b[irow][l];
  b[irow][l]=b[icol][l];
  b[icol][l]=temp;
  }
 indxr[i]=irow;
 indxc[i]=icol;
 if(b[icol][icol] == 0.0)
 {
  System.out.println("error : inverse of the matrix : singular matrix-2");
                for(ii=0;ii<n;ii++)
                  for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
  return b;
 }
 pivinv=1.0/b[icol][icol];
 b[icol][icol]=1.0;
 for(l=0;l<n;l++) b[icol][l] *=pivinv;
 for(ll=0;ll<n;ll++)
 if(ll != icol)
 {
  dum=b[ll][icol];
  b[ll][icol]=0.0;
  for(l=0;l<n;l++) b[ll][l]-= b[icol][l]*dum;
 }
```

```
}
for(l=n-1;l>=0;l--)
{
  if(indxr[l] != indxc[l])
    for(k=0;k<n;k++)
    {
      temp=b[k][indxc[l]];
      b[k][indxc[l]]=b[k][indxr[l]];
      b[k][indxr[l]]=temp;
    }
}
return b;
}

  public static void main(String args[])
  {
  double a[][]={{3,-0.1,-0.2},{0.1,7,-0.3},{0.3,-0.2,10}};
  System.out.println("Matris a : \n"+Matrix.toString(a));
  System.out.println("Ters Matris a : \n"+Matrix.toString(inv(a)));
  }
}
```

Çıktı 3.3-2 Tam pivotlu gauss eliminasyon kullanan inv metodu çıktısı

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" tersmatris
Matris a :
      3.000000000000000     -0.100000000000000     -0.200000000000000
      0.100000000000000      7.000000000000000     -0.300000000000000
      0.300000000000000     -0.200000000000000     10.000000000000000

Ters Matris a :
      0.332488721339843      0.004944070205797      0.006798096532971
     -0.005181765888768      0.142902644602169      0.004183444020290
     -0.010078296957971      0.002709730785869      0.099879725984417

> Terminated with exit code 0.
```

Eğer ters matrisi biliyorsak bu bilgiyi denklem sistemini çözmek için kullanabiliriz

$$[A]\{x\} = \{b\}$$

$$\{x\} = [A]^{-1}\{b\}$$

Bu yolla denklem sistemi çözümüne bir excel örneği ile verelim: Eğer

$$\begin{bmatrix} -8 & 1 & 2 \\ 5 & 7 & -3 \\ 2 & 1 & -2 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \\ -2 \end{bmatrix}$$

	A	B	C	D	E	F	G
1			A		x =	b	
2		-8	1	2	x0	0	
3		5	7	-3	x1	10	
4		2	1	-2	x2	-2	
5							
6			A'				
7		-0.14865	0.054054	-0.22973		1	
8		0.054054	0.162162	-0.18919	x=	2	
9		-0.12162	0.135135	-0.82432		3	
10							

Bu denklemi çözmek için ters matrisi hesaplayan **MINVERSE**(Türkçe excelde **DIZEY_TERS**) ve matrisleri direk çarpan **MMULTIPLY**(Türkçe excelde **DÇARP**) fonksiyonlarını kullanabiliriz. Bu fonksiyonları kullanırken Enter(giriş) tuşu yerine **Ctrl-Shift-Enter** (**ctrl** ⇑ ↵) tuşlarını kullanacağız. A' ters matrisini yaratacağımız bölgeyi seçtikten sonra DIZEY_TERS(A2:D4) gibi bir komutla(parantez içindeki A matrisinin yeri) Ctrl-Shift-Enter basarak ters matrisi elde ederiz sonra DÇARP(B7:D9;F2:F4) ifadesiyle (parantez içinde A' matrisinin yeri ve b vektörünün yeri tanımlanmıştır) yine Ctrl-Shift-Enter kullanarak x vektörünü buluruz.

3.14 Problemler

PROBLEM 1

$$\begin{bmatrix} 7 & 1 & 2 \\ -1 & 4 & -1 \\ 3 & 15 & 20 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix}=\begin{bmatrix} 47 \\ 19 \\ 87 \end{bmatrix}$$

Denklem sistemini çözünüz.

PROBLEM 2

$$\begin{bmatrix} 1.1348 & 3.8326 & 1.1651 & 3.4017 \\ 0.5301 & 1.7875 & 2.5330 & 1.5435 \\ 3.4129 & 4.9317 & 8.7643 & 1.3142 \\ 1.2371 & 4.9998 & 10.6721 & 0.0147 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix}=\begin{bmatrix} 9.5342 \\ 6.3941 \\ 18.4231 \\ 16.9237 \end{bmatrix}$$

denklemini çözünüz. Tam çözüm : $\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix}=\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

excel inverse matrix çözümü:

	A			x	b
1.1348	3.8326	1.1651	3.4017	x0	9.5342
0.5301	1.7875	2.533	1.5435	x1	6.3931
3.4129	4.9317	8.7643	1.3142	x2	18.4231
1.2371	4.9998	10.6721	0.0147	x3	16.9237

A'

-0.07646	-0.2301	0.471788	-0.32449
0.528763	-1.07023	-0.11496	0.290695
-0.23859	0.526548	-0.0008	-0.00457
-0.19455	1.102209	-0.0276	-0.2177

x0	1.00023
x1	1.00107
x2	0.999473
x3	0.998898

PROBLEM 3

$$\begin{bmatrix} 4 & 3 & -1 \\ 7 & -2 & 3 \\ 5 & -18 & 13 \end{bmatrix}\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix}=\begin{bmatrix} 6 \\ 9 \\ 3 \end{bmatrix}$$

denklem sistemini çözünüz.

PROBLEM 4

$$\begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/4 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \begin{bmatrix} 137/60 \\ 87/60 \\ 459/60 \\ 743/840 \\ 1879/2520 \end{bmatrix}$$

denklem sistemini çözünüz.

PROBLEM 5

$$\begin{bmatrix} 8 & 2 & 3 \\ 1 & -9 & 2 \\ 2 & 3 & 6 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 30 \\ 1 \\ 31 \end{bmatrix}$$

denklem sistemini Gauss-Seidel iteratif yöntemini kullanarak çözünüz

PROBLEM 6

$$\begin{bmatrix} 3 & -5 & 47 & 20 \\ 11 & 16 & 17 & 10 \\ 56 & 22 & 11 & -18 \\ 17 & 66 & -12 & 7 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 18 \\ 26 \\ 34 \\ 82 \end{bmatrix}$$

denklem sistemini rahatlamalı gauss seidel veya diğer iteratif yöntemleri kullanarak çözünüz.

PROBLEM 7

$$\begin{bmatrix} -2 & 1 & & & & & & & \\ 1 & -2 & 1 & & & & & & \\ & 1 & -2 & 1 & & & & & \\ & & & \cdots & & & & & \\ & & & 1 & -2 & 1 & & & \\ & & & & 1 & -2 & 1 & & \\ & & & & & 1 & -2 & & \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \\ X_7 \\ X_8 \\ X_9 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -1.5 \\ -1.5 \\ \cdots \\ -1.5 \\ -1.5 \\ 0.5 \end{bmatrix}$$

denklem sistemini Thomas algoritması (3'lü bant matris yöntemi) kullanarak çözünüz.

PROBLEM 8

$$\begin{bmatrix} -4 & 1 & & & & & & & \\ 1 & -4 & 1 & & & & & & \\ & 1 & -4 & 1 & & & & & \\ & & & \cdots & & & & & \\ & & & 1 & -4 & 1 & & & \\ & & & & 1 & -4 & 1 & & \\ & & & & & 1 & -4 & & \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \\ X_7 \\ X_8 \\ X_9 \end{bmatrix} = \begin{bmatrix} -27 \\ -15 \\ -15 \\ \cdots \\ -15 \\ -15 \\ -15 \end{bmatrix}$$

denklem sistemini Thomas algoritması kullanarak çözünüz.

PROBLEM 9

$$\begin{bmatrix} 3 & -5 & 6 & 4 & -2 & -3 & 8 \\ 1 & 1 & -9 & 15 & 1 & -9 & 2 \\ 2 & -1 & 7 & 5 & -1 & 6 & 11 \\ -1 & 1 & 3 & 2 & 7 & -1 & -2 \\ 4 & 3 & 1 & -7 & 2 & 1 & 1 \\ 2 & 9 & -8 & 11 & -1 & -4 & -1 \\ 7 & 2 & -1 & 2 & 7 & -1 & 9 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix} = \begin{bmatrix} 47 \\ 17 \\ 24 \\ 8 \\ 13 \\ -10 \\ 34 \end{bmatrix}$$

denklem sistemini çözünüz.

PROBLEM 10

$$\begin{bmatrix} 1 & -1 & 2 & 5 & -7 & -8 \\ 3 & -9 & 1 & -1 & 8 & 1 \\ -1 & 1 & 9 & -9 & 2 & 3 \\ 1 & 7 & 2 & -3 & -1 & 4 \\ 7 & 1 & -2 & 4 & 1 & -1 \\ 2 & 3 & -9 & 12 & -2 & 7 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \end{bmatrix} = \begin{bmatrix} -12 \\ 8 \\ 22 \\ 41 \\ 15 \\ 50 \end{bmatrix}$$

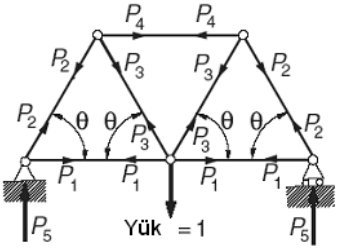
denklem sistemini çözünüz.

PROBLEM 11

$$\begin{bmatrix} 10 & -2 & -1 & 2 & 3 & 1 & -4 & 7 \\ 5 & 11 & 3 & 10 & -3 & 3 & 3 & -4 \\ 7 & 12 & 1 & 5 & 3 & -12 & 2 & 3 \\ 8 & 7 & -2 & 1 & 3 & 2 & 2 & 4 \\ 2 & -15 & -1 & 1 & 4 & -1 & 8 & 3 \\ 4 & 2 & 9 & 1 & 12 & -1 & 4 & 1 \\ -1 & 4 & -7 & -1 & 1 & 1 & -1 & -3 \\ -1 & 3 & 4 & 1 & 3 & -4 & 7 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 0 \\ 12 \\ -5 \\ 3 \\ -25 \\ -26 \\ 9 \\ -7 \end{bmatrix}$$

denklem sistemini çözünüz.

PROBLEM 12

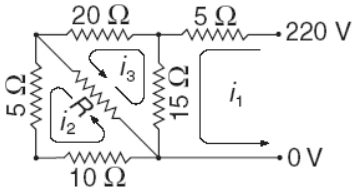


(bolum3_000.jpg,Resimaltı:)
Şekilde görülen sistemdeki yük dengesi

$$\begin{bmatrix} c & 1 & 0 & 0 & 0 \\ 0 & s & 0 & 0 & 1 \\ 0 & 0 & 2s & 0 & 0 \\ 0 & -c & c & 1 & 0 \\ 0 & s & s & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

şeklinde yazılabilir. Burada c=cos(θ),s=sin(θ) dır. θ =60° için yük dağılımını bulunuz.

PROBLEM 13



(bolum3_001.jpg,Resimaltı:)

Şekilde görülen devre için Voltaj akım şiddeti denklemleri

$$5i_1 + 15(i_1 - i_3) = 220\,\mathrm{V}$$

$$R(i_2 - i_3) + 5i_2 + 10i_2 = 0$$

$$20i_3 + R(i_3 - i_2) + 15(i_3 - i_1) = 0$$

şeklinde yazılabilir. R=10 Ω için akım şiddeti, i (Amper), değerlerini hesaplayınız.

PROBLEM 14

$-x_1 + 3x_2 + 5x_3 + x_4 = 8$
$x_1 + 9x_2 + 3x_3 + 4x_4 = 10$
$x_2 + x_4 = 2$
$3x_1 + x_2 + x_3 - x_4 = -4$

doğrusal denklem sistemini uygun bir yöntem ile çözümleyiniz.

PROBLEM 15

$$\begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & -1 & 3 & 5 \\ 0 & 0 & 2 & 5 \\ -2 & -6 & -3 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

doğrusal denklem sistemini LU çarpanlara ayırma (Doolittle) yöntemi ile çözünüz.

PROBLEM 16

$3x_1 + 5x_2 + 4x_3 + 2x_4 = 11$
$6x_1 + 14x_2 + 11x_3 + 6x_4 = 26$
$9x_1 + 11x_2 + 16x_3 + 5x_4 = 68$
$3x_1 + 13x_2 + 17x_3 + 12x_4 = 43$

doğrusal denklem sistemini LU çarpanlara ayırma yöntemi ile çözümleyiniz.

PROBLEM 17

$$x_1 + 3x_2 + x_3 + 5x_4 = 4$$
$$2x_1 + x_2 + 3x_4 = 5$$
$$4x_1 + 2x_2 + 2x_3 + x_4 = 11$$
$$-3x_1 + x_2 + 3x_3 + 2x_4 = 3$$

denklem sistemini,

a) Gauss yoketme yöntemi ile
b) Gauss-seidel yöntemi ile
c) Konjuge granyan yöntemi ile çözünüz.

PROBLEM 18

$x_1 + x_2 - x_3 = -3$
$6x_1 + 2x_2 + 2x_3 = 2$
$-3x_1 + 4x_2 + x_3 = 1$
denklem sistemini kısmi pivotlamalı gauss eleme yöntemi ile çözün.

PROBLEM 19 Verilen matrisin sonsuz, iki ve üç normlarını bulunuz.

$$\begin{bmatrix} 7 & 1 & 2 \\ -1 & 4 & -1 \\ 3 & 15 & 20 \end{bmatrix}$$

PROBLEM 20

Verilen matrisin ters matrisini bulunuz

$$\begin{bmatrix} 1.1348 & 3.8326 & 1.1651 & 3.4017 \\ 0.5301 & 1.7875 & 2.5330 & 1.5435 \\ 3.4129 & 4.9317 & 8.7643 & 1.3142 \\ 1.2371 & 4.9998 & 10.6721 & 0.0147 \end{bmatrix}$$

PROBLEM 21

Verilen matrisin ters matrisini bulunuz

$$\begin{bmatrix} 4 & 3 & -1 \\ 7 & -2 & 3 \\ 5 & -18 & 13 \end{bmatrix}$$

PROBLEM 22

Verilen matrisin sonsuz, iki ve üç normlarını bulunuz.

$$\begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/4 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{bmatrix}$$

PROBLEM 23

Verilen matrisin sonsuz, iki normlarını ve ters matrisini bulunuz.

$$\begin{bmatrix} 8 & 2 & 3 \\ 1 & -9 & 2 \\ 2 & 3 & 6 \end{bmatrix}$$

PROBLEM 24

Verilen matrisin sonsuz, iki normlarını ve ters matrisini bulunuz.

$$\begin{bmatrix} 3 & -5 & 47 & 20 \\ 11 & 16 & 17 & 10 \\ 56 & 22 & 11 & -18 \\ 17 & 66 & -12 & 7 \end{bmatrix}$$

PROBLEM 25

Verilen matrisin sonsuz, iki normlarını ve ters matrisini bulunuz.

$$\begin{bmatrix} -2 & 1 & & & & & & \\ 1 & -2 & 1 & & & & & \\ & 1 & -2 & 1 & & & & \\ & & & \ddots & & & & \\ & & & 1 & -2 & 1 & & \\ & & & & 1 & -2 & 1 & \\ & & & & & 1 & -2 \end{bmatrix}$$

Bölüm 4 Doğrusal olmayan denklem Sistemlerinin kökleri

4.1 Newton - Raphson yöntemi

Bir boyutlu Newton-Raphson formülünde

$x_{n+1} = x_n - f(x_n) / f'(x_n)$

formülüne ulaşmıştık. Bu formülü biraz değiştirirsek aynı formül

$\Delta x_{n+1} = x_{n+1} - x_n = - f(x_n) / f'(x_n)$

$f'(x_n) * \Delta x_{n+1} = - f(x_n)$

formunda yazılabilir. Şimdi aynı işlemi çok boyutlu bir denklem sisteminin köklerini bulma problemi olarak düşününelim. Türev denklemi:

$$[\nabla f] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \cdots & \frac{\partial f_3}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_3} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \qquad \{\Delta x\} = \begin{Bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \cdots \\ \Delta x_n \end{Bmatrix} \quad \{f\} = \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ \cdots \\ f_n \end{Bmatrix}$$

şeklinde verilebilir. Bu durumda Newton-raphson denklemi

$[\nabla f]\ \{\Delta x\} = - \{f\}$ şeklini alır.

Görüldüğü gibi denklemimiz bir lineer denklem sistemidir ve bir önceki bölümde gördüğümüz tekniklerle çözülebilir. Her döngüsel çözümde $\{\Delta x\}$ in yeni değerlerini elde edeceğimizden, bunlardan da yeni x değerlerini buluruz.

Program 4.16-1 Çok boyutlu lineer olmayan denklem çözümü (köklerini bulma)

```
 // SCO5 Newton Raphson metoduyla
// birden fazla değişkenli denklem sisteminin köklerini bulmak

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

public class SCO5
{
// Newton Raphson Metodunu kullanarak denklem sisteminin köklerini bulmak
public static double[] f(double x[])
{
//çözümü istenen fonksiyon
double ff[]=new double[2];
ff[0]=x[0]*x[0]+x[0]*x[1]-10.0;
ff[1]=x[1]+3.0*x[0]*x[1]*x[1]-57.0;
return ff;
}

public static double[][] df(double x[])
{
//çözümü istenen fonksiyonun türev fonksiyonları
double a[][]=new double[2][2];
a[0][0]=2.0*x[0]+x[1];
a[0][1]=x[0];
a[1][0]=3.0*x[1]*x[1];
a[1][1]=1.0+6.0*x[0]*x[1];
return a;
}

public static double[] multiply(double[] left,double right)
{
//multiplying a vector with a constant
int i;
int n=left.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {
  b[i]=right*left[i];
  }
return b;
}

public static double abs(double[] left)
{
// absolute value of a vector
int i;
int n=left.length;
double b=0;
```

```
for(i=0;i<n;i++)
  {
  b=b+Math.abs(left[i]);
  }
return b;
}

public static double[] divide(double[] left,double[][] right)
{
return pivotlugauss(right,left);
}
  //en kucuk kareler metodu
  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
              p=k;
              buyuk=Math.abs(a[k][k]);
              for(ii=k+1;ii<n;ii++)
              {  dummy=Math.abs(a[ii][k]);
                 if(dummy > buyuk) {buyuk=dummy;p=ii;}
              }
              if(p!=k)
              { for(jj=k;jj<n;jj++)
                { dummy=a[p][jj];
                  a[p][jj]=a[k][jj];
                  a[k][jj]=dummy;
                }
                dummy=b[p];
                b[p]=b[k];
                b[k]=dummy;
              }
              //gauss elemeyi çözme
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
           b[i]   =b[i]   -carpan*b[k];
              }
  }
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
```

```
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
}

public static double[] newton(double x[])
{
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi dfj/dxi de tanımlanacaktır
// Newton-Raphson metodu ile lineer olmıyan denklem sistemi çözümü
//ti :önem katsayısı
//x bağımsız değişken vektörü
//y bağımlı değişken vektörü
//dy bağımlı değişkenin türev fonksiyonları vektörü
double ti=1.0;
int i;
int nmax=100;
double tolerance=1.0e-15;
int n=x.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
{
b[i]=1.0;
```

```
}
i=0;
while( i++ < nmax && abs(b) > tolerance )
{ b=multiply(divide(f(x),df(x)),-ti);
  x=add(x,b);
  System.out.print("i="+i+Matrix.toString(x));
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
            " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);

return x;
}

public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
   Double ax=new Double(token.nextToken());
   a[j++]=ax.doubleValue();
   }
   return a;
}
    public static void main (String args[])
    {
    double [] x0=verigir();
    double [] r= newton(x0);
    String s=" kök değerleri  : \n"+Matrix.toStringT(r);
    String s1="çok değişkenli Newton-Raphson metodu ile kök bulma : \n";
    System.out.println(s1+s);
    //JOptionPane.showMessageDialog(null,s,s1,JOptionPane.PLAIN_MESSAGE);
    //System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO5
i=1       2.17647058823529440000000000       1.94117647058823550000000000
i=2       1.90083304356406240000000000       3.21523798689502540000000000
i=3       1.99912715214104960000000000       2.99716665219926930000000000
i=4       1.99999967905467570000000000       3.00000274124264000000000000
i=5       1.99999999999957460000000000       3.00000000000011010000000000
i=6       2.00000000000000000000000000       3.00000000000000040000000000
i=7       2.00000000000000000000000000       2.99999999999999960000000000
çok değişkenli Newton-Raphson metodu ile kök bulma :
 kök değerleri  :
```

```
      2.000000000000000000000000
      2.999999999999996000000000

> Terminated with exit code 0.
```

Newton-Raphson denkleminde en önemli problemlerden biri türevlerin değerlerinin gerekmesidir. Bu zorluğu türevleri sayısal yöntemlerle (fark denklemleri olarak) hesaplamaktır.

Program 4.16-2 Çok boyutlu lineer olmayan denklem çözümü (köklerini bulma)
Türev matrisi sayısal olarak hesaplanıyor

```
 // FSCO5A Newton Raphson metoduyla
// birden fazla değişkenli denklem sisteminin köklerini bulmak

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============== Tanımı gereken fonksiyon ================
abstract class fi_xi
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x) returns the value of f[0] and f[1]
  // as a two dimensional vector

  abstract double[] func(double x[]);
}

class fa extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyonun türev fonksiyonları
//çözümü istenen fonksiyon
double ff[]=new double[2];
ff[0]=x[0]*x[0]+x[0]*x[1]-10.0;
ff[1]=x[1]+3.0*x[0]*x[1]*x[1]-57.0;
return ff;
}
}

public class FSCO5A
{
// Newton Raphson Metodunu kullanarak denklem sisteminin köklerini bulmak
// Bu metod denklem sistemi çözümü için Matrix.java programını
// kullanmaktadır.



public static double[] newtond( fi_xi f,double x[])
```

```
{
// lineer olmayan denklem siteminin kökleri
// fj(xi)=0 lineer olmayan denklem sistemi için çözüm matrisi
// türev matrisi dfj/dxi turev alt programıyla hesaplanır.
// Newton-Raphson metodu ile lineer olmayan denklem sistemi çözümü
//ti :önem katsayısı
//x bağımsız değişken vektörü
//y bağımlı değişken vektörü
//dy bağımlı değişkenin türev fonksiyonları vektörü
double ti=1.0;
int i,ii,jj;
int nmax=500;
double tolerance=1.0e-15;
int n=x.length;
double b[];
b=new double[n];
double dy[][];
dy=new double[n][n];
i=0;
for(i=0;i<n;i++)
{
b[i]=1.0;
}
while( i++ < nmax && Matrix.abs(b) > tolerance )
{
  for(ii=0;ii<n;ii++)
  {
    for(jj=0;jj<n;jj++)
    {
       dy[ii][jj]=turev(f,x,ii,jj);
    }
  }
    b=Matrix.multiply(Matrix.divide(f.func(x),dy),-ti);
    x=Matrix.add(x,b);
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}

public static double turev(fi_xi f,double x[],int denklem_ref,int x_ref)
{
// fonksiyon f in denklem_ref sayılı fonksiyonunun(sayılar 0 dan başlar)
// x[x_ref} değişkenine göre türevi (sayılar 0 dan başlar)
// df_denklem_ref(x)/d_x_ref
// bu metod newtond metodu içinde kullanılmak içindir.
double h0=0.0256;
int i,m;
int n=7;
double f1[];
f1=new double[x.length];
```

```
double f2[];
f2=new double[x.length];
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
x2[i]=x[i];
}
//derivative of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i];
f1=f.func(x1);
f2=f.func(x2);
T[i][0]=( f1[denklem_ref] - f2[denklem_ref])/(2.0*h[i]);
x1[x_ref]=x[x_ref];
x2[x_ref]=x[x_ref];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}
```

```
public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
   Double ax=new Double(token.nextToken());
   a[j++]=ax.doubleValue();
   }
   return a;
}

    public static void main (String args[])
    {
          fa f =new fa();
    double [] x0=verigir();
    String s2="türev fonksiyonları sayısal hesaplanmış Newton-Raphson kök bulma : ";
    double [] r1= newtond(f,x0);
    String s=" kök değerleri  : \n"+Matrix.toStringT(r1);
    System.out.println(s);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO5A
 kök değerleri  :
       2.000000000000000000000000
       3.000000000000000004000000

> Terminated with exit code 0.
```

4.2 Sherman-Morrison formüllü Broyden kiriş yöntemi

f(x)=0 lineer olmıyan denklem sistemini çözmek istiyoruz. Burada x={x_1, x_1,…, x_{n-1}, x_n} çok boyutlu değişken setidir. Bu durumda kiriş metodunu uygulayarak türev fonksiyonunu yaklaşık değerini oluşturabiliriz.
Hatırlanacağı gibi, Newton-Raphson denklemi

$$[\nabla f] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \cdots & \frac{\partial f_3}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_3} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \qquad \{\Delta x\} = \begin{Bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \cdots \\ \Delta x_n \end{Bmatrix} \quad \{f\} = \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ \cdots \\ f_n \end{Bmatrix}$$

şeklinde tanımlanmıştı. Bu durumda Newton-raphson denklemi

$[\nabla f]\ \{\Delta x\} = -\{f\}$ şeklinde döngüsel olarak çözülebilir.

Burada $[\nabla f]$ yerine yaklaşık türev formülünden yararlanmak istersek

$\frac{\partial f_j}{\partial x_k}(x^{(i)}) \approx \frac{f_j(x^{(i)} + e_k h) - f_j(x^{(i)})}{h}$ gibi bir formülle hesaplayabiliriz. Formüldeki h küçük bir sonlu değişim miktarı, e_k ise k inci bileşeni 1 diğer bileşenleri 0 olan bir vektördür. Buradaki (i) iterasyon sayısını belirtir. Kiriş yöntemi uygulandığında Newton-raphson formülünün kök civarındaki hızlı yakınsaması yerini daha yavaş bir yakınsama hızına bırakır. Bu yüzden burada standart kiriş yöntemi yerine yakınsama hızı birim hesaplama başına daha etkin olan Broyden yöntemini inceleyeceğiz. Broyden yönteminde $x^{(0)}$ ilk iterasyonu verilmiş ise $x^{(1)}$ iterasyonunu Newton metodunu kullanarak hesaplayabiliriz. Bundan sonraki stepler için şu yaklaşımı kullanırız :

$[\nabla f(x^{(0)})]\ \{x^{(1)} - x^{(0)}\} = -\{f(x^{(0)}\}$ denklemi

$[A_1]\ \{x^{(1)} - x^{(0)}\} = \{f(x^{(1)} - f(x^{(0)}\}$ şeklini alabilir. Bu denklemin geçerli olabilmesi için $[A_1]$ vektörünün her z vektörü için

$[A_1]\ \{z\} = [\nabla f(x^{(0)})]z$ $\{x^{(1)} - x^{(0)}\}^t z = 0$ tanımlarını gerçekleştirmesi gerekir. Bu temel tanımdan $[A_1]$ formülünü oluşturabiliriz.

$$[A_1] = \nabla f(x^{(0)}) + \frac{[\{f(x^{(1)}) - f(x^{(0)})\} - \nabla f(x^{(0)})(x^{(1)} - x^{(0)})] * (x^{(1)} - x^{(0)})^t}{\left\| x^{(1)} - x^{(0)} \right\|_2^2}$$

buradaki $\left\| x^{(1)} - x^{(0)} \right\|_2^2$ ifadesi 2 normudur. $[A_1]$ değerini bulduktan sonra x vektörünün yeni değerini oluşturabiliriz.

$$x^{(2)} = x^{(1)} - A_1^{-1} f(x^{(1)})$$

$x^{(2)}$ bulunduktan sonra bu değeri aynı yolla $x^{(3)}$ ü hesaplamak için kullanırız. Genel olarak formülümüz

$$[A_i] = [A_{i-1}] + \frac{[\{f(x^{(1)}) - f(x^{(0)})\} - [A_{i-1}](x^{(1)} - x^{(0)})] * (x^{(1)} - x^{(0)})^t}{\left\| x^{(1)} - x^{(0)} \right\|_2^2}$$

$$x^{(i+1)} = x^{(i)} - A_i^{-1} f(x^{(i)})$$

formunu alır. Bu formülde her seferinde A matrisinin ters matrisini çözme zorunluluğumuz mevcuttur. Bu işlemi Gauss eleme yöntemi gibi bir yöntem kullanarak gerçekleştirebiliriz. Ancak Gauss yöntemide oldukça yüksek hesaplama zamanına ihtiyaç gösterir. Daha iyi bir alternatif iteratif Sherman-Morrison formülünü kullanmaktır. Sherman_Morrison formülü A_i^{-1} değerini direk olarak A_{i-1}^{-1} değerinden hesaplıyabilmemizi sağlar. Bu formülü şu stepleri kullanarak oluştururuz:

$$s_i = \{x^{(1)} - x^{(i-1)}\}$$

$$y_i = \{f(x^{(i)}) - f(x^{(i-1)})\}$$

$$(A + xy^t)^{-1} = A^{-1} - \frac{A^{-1} x y^t A^{-1}}{1 + y^t A^{-1} x}$$

$$A_i^{-1} = \left(A^{-1} + \frac{y_i - A_{i-1}s_i}{\|s_i\|_2^2} s_i^t \right)^{-1}$$

$$A_i^{-1} = A^{-1} + \frac{A_{i-1}^{-1}\left(\frac{y_i - A_{i-1}s_i}{\|s_i\|_2^2} s_i^t \right) A_{i-1}^{-1}}{1 + s_i^t A_{i-1}^{-1}\left(\frac{y_i - A_{i-1}s_i}{\|s_i\|_2^2} s_i^t \right)} s_i^t$$

$$A_i^{-1} = A^{-1} + \frac{(s_i - A_{i-1}^{-1}y_i)s_i^t A_{i-1}^{-1}}{s_i^t A_{i-1}^{-1} y_i}$$

bu denklem sadece matris çarpımlarını kapsadığından toplam işlem olarak yapılması çok daha kolaydır.
Bu denklemleri kullanarak oluşturulmuş Broyden algoritması Program 4.2.1 de verilmiştir.

Program 4.2.1 Çok boyutlu lineer olmayan denklem çözümü (köklerini bulma) Broyden kiriş metodu

```
 //  SCO5E Sherman-Morrison formülü destekli broyden kiriş metoduyla
//  birden fazla değişkenli denklem sisteminin köklerini bulmak

// Metod Referansı : Numerical Analysis, 8th edition Richard L. Burden, J. Douglas Faires 3
// sayfa 617

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============= Tanımı gereken fonksiyon ================
abstract class fi_xi
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x) returns the value of f[0] and f[1]
  // as a two dimensional vector

  abstract double[] func(double x[]);
}

abstract class fij_xi
{
  // multifunction multi independent variable
  // for n independent variable n*n matrix of
  // dependent variables are returned
  // example f[0][0]=x[0]+Math.sin(x[1])   f[0][1]=x[0]-x[1]
  //         f[1][0]=x[0]*x[0]-x[1]        f[1][1]=Math.exp(x[0]+x[1]*x[1]
```

```
  // func(x) returns the value of f[0][0], f[0][1]
  //                              f[1][0], f[1][1]
  // as a two dimensional matrix
  abstract double[][] func(double x[]);
}

class fa extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyon
double ff[]=new double[3];
ff[0]=3.0*x[0]-Math.cos(x[1]*x[2])-0.5;
ff[1]=x[0]*x[0]-81.0*(x[1]+0.1)*(x[1]+0.1)+Math.sin(x[2])+1.06;
ff[2]=Math.exp(-x[0]*x[1])+20.0*x[2]+(10.0*Math.PI-3.0)/3.0;
return ff;
}
}

class fb extends fij_xi
{
fi_xi f;

public fb(fi_xi fi)
{f=fi;}

public double[][] func(double x[])
{
  int n=x.length;
  double a[][]=new double[n][n];
  for(int i=0;i<n;i++)
  {
    for(int j=0;j<n;j++)
    {
       a[i][j]=turev(x,i,j);
    }
  }
return a;
}

public double turev(double x[],int denklem_ref,int x_ref)
{
// fonksiyon f in denklem_ref sayılı fonksiyonunun(sayılar 0 dan başlar)
// x[x_ref} değişkenine göre türevi (sayılar 0 dan başlar)
// df_denklem_ref(x)/d_x_ref
// bu metod newtond metodu içinde kullanılmak içindir.
double h0=0.0256;
int i,m;
int n=7;
double f1[];
f1=new double[x.length];
double f2[];
```

```
f2=new double[x.length];
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
x2[i]=x[i];
}
//derivative of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i];
f1=f.func(x1);
f2=f.func(x2);
T[i][0]=( f1[denklem_ref] - f2[denklem_ref])/(2.0*h[i]);
x1[x_ref]=x[x_ref];
x2[x_ref]=x[x_ref];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}

}
```

```
public class FSCO5E
{
// SCO5E Sherman-Morrison formülü destekli broyden sekant metoduyla
//denklem sistemlerinin köklerini (çözümünü) bulmak

public static double[] multiply(double left,double[] right)
{
//multiplying a vector with a constant
int i;
int n=right.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {b[i]=left*right[i];}
return b;
}

public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right.length;
double[] b;
b=new double[m2];
 if(n1 != m2)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
   {
     b[ii]=0;
   }
   return b;
 }
 for(i=0;i<m1;i++)
   {
     b[i]=0;
     for(k=0;k<n1;k++)
            b[i]+=left[i][k]*right[k];
   }
return b;
//end of multiply of a matrix and a vector
}

public static double[] multiply(double[] left,double[][] right)
{
//multiplication of one vector with one matrix
int ii,jj,i,j,k;
int m2=right[0].length;
int n2=right.length;
```

```
int m1=left.length;
double[] b;
b=new double[m1];
 if(n2 != m1)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n2;ii++)
   {
     b[ii]=0;
   }
   return b;
 }
 for(i=0;i<m2;i++)
   {
     b[i]=0;
     for(k=0;k<m1;k++)
             b[i]+=right[i][k]*left[k];
   }
return b;
//end of multiply of a vector and a matrix
}

public static double  VT_X(double [] left,double [] right)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*right[i];
  }
return tot;
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]+right[i];
```

```
  }
return b;
//end of vector addition method
}

public static double[][] multiply(double left,double[][] right)
{
//multiplying a matrix with a constant
int i,j;
int n=right.length;
int m=right[0].length;
double b[][];
b=new double[n][m];
for(i=0;i<n;i++)
  {
  for(j=0;j<m;j++)
    b[i][j]=right[i][j]*left;
  }
return b;
//end of multiplying a matrix with a constant double
}

public static double[][] add(double[][] left,double[][] right)
{
//addition of two matrices
int n1=left.length;
int m1=left[0].length;
int n2=right.length;
int m2=right[0].length;
int nMax,mMax;
int i,j;
if(m1>=m2) mMax=m1;
else       mMax=m2;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[][];
b=new double[nMax][mMax];
for(i=0;i<n1;i++)
  {
  for(j=0;j<m1;j++)
    {
    b[i][j]=b[i][j]+left[i][j];
    }
  }
for(i=0;i<n2;i++)
  {
  for(j=0;j<m2;j++)
    {
    b[i][j]=b[i][j]+right[i][j];
    }
  }
return b;
```

```
//end of matrix addition method
}

public static double[] substract(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]-right[i];
  }
return b;
//end of vector addition method
}

public static double[][] inv(double[][] a)
{
// INVERSION OF A MATRIX
// inversion by using gaussian elimination
// with full pivoting
int n=a.length;
int m=a[0].length;
double b[][];
b=new double[n][n];
int indxc[];
int indxr[];
double ipiv[];
indxc=new int[n];
indxr=new int[n];
ipiv=new double[n];
int i,j,k,l,ll,ii,jj;
int icol=0;
int irow=0;
double big,dum,pivinv,temp;
if(n!=m)
{
   System.out.println("Matrix must be square ");
              for(ii=0;ii<n;ii++)
                for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
              return b;
```

```
}
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
      b[i][j]=a[i][j];
for(i=0;i<n;i++)
{
  big=0.0;
  for(j=0;j<n;j++)
  {
    if(ipiv[j] != 1)
      for(k=0;k<n;k++)
      {
            if(ipiv[k] == 0)
            {
              if(Math.abs(b[j][k]) >= big)
              {
                big=Math.abs(b[j][k]);
                irow=j;
                icol=k;
              }
            }
            else if(ipiv[k] > 1 )
              {
              System.out.println("error : inverse of the matrix : singular matrix-1");
              for(ii=0;ii<n;ii++)
                for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
              return b;
              }
      }
  }
  ++ ipiv[icol];
  if(irow != icol)
    for(l=0;l<n;l++)
    {
    temp=b[irow][l];
    b[irow][l]=b[icol][l];
    b[icol][l]=temp;
    }
  indxr[i]=irow;
  indxc[i]=icol;
  if(b[icol][icol] == 0.0)
  {
    System.out.println("error : inverse of the matrix : singular matrix-2");
              for(ii=0;ii<n;ii++)
                for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
    return b;
  }
  pivinv=1.0/b[icol][icol];
  b[icol][icol]=1.0;
  for(l=0;l<n;l++) b[icol][l] *=pivinv;
```

```
  for(ll=0;ll<n;ll++)
  if(ll != icol)
  {
    dum=b[ll][icol];
    b[ll][icol]=0.0;
    for(l=0;l<n;l++) b[ll][l]-= b[icol][l]*dum;
  }
}
for(l=n-1;l>=0;l--)
{
  if(indxr[l] != indxc[l])
     for(k=0;k<n;k++)
     {
       temp=b[k][indxc[l]];
       b[k][indxc[l]]=b[k][indxr[l]];
       b[k][indxr[l]]=temp;
     }
}
return b;
}
public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}

public static double[][] multiply(double[] left,double[] right)
{
//multiplication of one vector with one matrix
int ii,jj,i,j,k;
int m2=left.length;
int n2=right.length;
int m1=left.length;
double[][] b;
b=new double[m2][n2];
 for(i=0;i<m2;i++)
   {
     for(k=0;k<n2;k++)
               b[i][k]=left[i]*right[k];
   }
return b;
//end of multiply of a vector and a matrix
}

public static double[] broyden(fi_xi f,double x[])
{
//=====================================================
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
```

```
// türev matrisi J de tanımlanacaktır
// Sherman-Morrison formülü uygulanmış broyden (kiriş tipi) metodu
// ile lineer olmıyan denklem sistemi çözümü
fb J=new fb(f);
int i;
int nmax=400;
double tolerance=1.0e-15;
int n=x.length;
double b[]=new double[n];
double v[]=new double[n];
double s[]=new double[n];
double w[]=new double[n];
double y[]=new double[n];
double z[]=new double[n];
double p;
double ut[]=new double[n];
double x1[]=new double[n];
double A[][]=new double[n][n];
x1=x;
A=J.func(x1);
v=f.func(x1);
A=inv(A);
s=multiply(-1.0,multiply(A,v));
x=add(x,s);
int k=1;
while(k<nmax)
{
w=v;
v=f.func(x);
y=substract(v,w);
z=multiply(-1.0,multiply(A,y));
p=-VT_X(s,z);
ut=multiply(s,A);  //*********
A=add(A,multiply((1.0/p),multiply(add(s,z),ut)));
s=multiply(-1.0,multiply(A,v));
x=add(x,s);
if(norm(s)<tolerance) break;
k++;
}
return x;
}

public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
```

```
   Double ax=new Double(token.nextToken());
   a[j++]=ax.doubleValue();
   }
   return a;
}

    public static void main (String args[])
    {
            fa f =new fa();
    double [] x0=verigir();
    double [] r= broyden(f,x0);
    String s=" kök değerleri broyden kiriş metodu: \n"+Matrix.toStringT(r);
    System.out.println(s);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO5A
 kök değerleri  :
      2.000000000000000000000000
      3.000000000000000400000000

> Terminated with exit code 0.
```

4.3 Süreklilik yöntemi

F(x)=0 lineer olmıyan denklem sistemini çözmek istiyoruz. Bulmak istediğimiz çözüm setine x^* diyelim. Şimdi 0 ile 1 arasında bir λ değeri alan bir problem seti düşünelim. Bu sette λ=0 değeri x(0) çözümüne karşı gelsin ve λ=1 değeri x(1)= x^* çözüm setine karşı gelsin. G(λ,x) fonksiyonunu

G(λ,x) = λF(x) + (1-λ)[F(x) - F(x(0))] olarak tanımlayacak olur isek, ve bu denklemden

G(λ,x) = 0 çözümünü elde etmek istersek, kök değerini veren çözümler için λ=0 olduğunda

0=G(0,x) = F(x) - F(x(0)) olur.

λ=1 olduğunda ise

0=G(1,x) = F(x) olur.

Bu yüzden x(1)=x^* çözüm setine eşit olacaktır. Öyleyse bu özellikleri sağlayan bir G(λ,x) fonksiyonu bulabilirsek bu fonksiyonu F(x) fonksiyonunun çözümü için kullanabiliriz. G fonksiyonuna geçiş veya süreklilik fonksiyonu adı veriyoruz(λ=0 dan λ=1 e geçiş)

Böyle bir fonksiyonu oluşturabilmek için

G(λ,x)=0 alınır ve bu fonksiyonun λ ya göre kısmi türevi alınırsa.

$$0 = \frac{\partial G(\lambda, x)}{\partial \lambda} + \frac{\partial G(\lambda, x)}{\partial x} x'(\lambda)$$ buradan x'(λ) çekilirse

$$x'(\lambda) = -\left[\frac{\partial G(\lambda, x(\lambda)}{\partial x}\right]^{-1}\left[\frac{\partial G(\lambda, x(\lambda))}{\partial \lambda}\right] x'(\lambda)$$

G(λ,x) = λF(x) + (1-λ)[F(x) - F(x(0))] değerini bu denklemde yerine koya isek :

$$\left[\frac{\partial G(\lambda, x(\lambda)}{\partial x}\right] = \begin{bmatrix} \frac{\partial f_1(x(\lambda))}{\partial x_1} & \frac{\partial f_1(x(\lambda))}{\partial x_2} & \frac{\partial f_1(x(\lambda))}{\partial x_3} \\ \frac{\partial f_2(x(\lambda))}{\partial x_1} & \frac{\partial f_2(x(\lambda))}{\partial x_2} & \frac{\partial f_2(x(\lambda))}{\partial x_3} \\ \frac{\partial f_3(x(\lambda))}{\partial x_1} & \frac{\partial f_3(x(\lambda))}{\partial x_2} & \frac{\partial f_3(x(\lambda))}{\partial x_3} \end{bmatrix} = J(x(\lambda)$$

Jakobiyen matristir. Ve

$$\left[\frac{\partial G(\lambda, x(\lambda)}{\partial x}\right] = F(x(0))$$

diferansiyel denklem

$$x'(\lambda) = \frac{dx(\lambda)}{d\lambda} = -\left[J(x(\lambda))\right]^{-1} F(x(0)) \qquad 0 \le \lambda \le 1$$

formunu alır. Bu diferansiyel denklem x(0) başlangıç değeri için çözülürse, çözümden elde edilen x(1) değeri bize G(λ,x) = 0 denkleminin çözümünü verecektir. Bunun için herhangi bir diferansiyel denklem çözüm yöntemini kullanabiliriz. Örneğin

$\frac{dx(\lambda)}{d\lambda} = f(\lambda, x(\lambda))$ denkleminin çözümünü veren Dördüncü dereceden Runge-Kutta yöntemi :

$x_{i+1} = x_i + (1/6)*(k_1 + 2k_2 + 2k_3 + k_4)h$

$k_1 = f(\lambda_i, x_i)$

$k_2 = f(\lambda_i + 0.5h, x_i + 0.5k_1h)$

$k_3 = f(\lambda_i + 0.5h, x_i + 0.5k_2h)$

$k_4 = f(\lambda_i + h, x_i + k_3h)$

formüllerini kullanarak diferansiyel denklemi step step çözebilir. Buradaki h step boyutudur. Çözüm λ=0 , $x_0(0)$ ilk değerini kullanarak hesaplamaya başlar, ve her stepte λ değerine h ekleyerek λ=1 değerine kadar devam eder. $x_0(1)$ değeri çözüm setimizdir. Diferansiyel denklemlerin çözümü daha detaylı olarak diferansiyel denklemler konusunda görülecektir. Diferansiyel denklemin çözümü bize çözüm fonksiyonunu verdiği için aynı zamanda denklemin köklerini de bulmuş oluruz. Program 4.3-1 de süreklilik denklemini 4 dereceden ve 6ıncı dereceden Runge-Kutta formülleri yardımı ile çözen süreklilik lineer olmıyan denklem çözme programı verilmiştir.

Program 4.3.1 Süreklilik yöntemi kullanarak çok boyutlu lineer olmayan denklem çözümü (köklerini bulma)

```
// SCO5C Süreklilik(Continuation-homotopy) metoduyla
// birden fazla değişkenli denklem sisteminin köklerini bulmak
// Metod Referansı : Numerical Analysis, 8th edition Richard L. Burden, J. Douglas Faires 3
// sayfa 637
// bu versiyon fonksiyonların türevini sayısal olarak almaktadır.
// kullanıcının türev fonksiyonu tanımlaması gerekmez
// süreklilik denklemleri bir önceki SCO5 ile aynı formda kalmıştır.
// kullanıcının sadece fa fonksiyonunu oluşturması gerekmektedir.

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
```

```
//============== Tanımı gereken fonksiyon ================
abstract class fi_xi
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x) returns the value of f[0] and f[1]
  // as a two dimensional vector

  abstract double[] func(double x[]);
}

abstract class fij_xi
{
  // multifunction multi independent variable
  // for n independent variable n*n matrix of
  // dependent variables are returned
  // example f[0][0]=x[0]+Math.sin(x[1])   f[0][1]=x[0]-x[1]
  //         f[1][0]=x[0]*x[0]-x[1]        f[1][1]=Math.exp(x[0]+x[1]*x[1]
  // func(x) returns the value of f[0][0], f[0][1]
  //                              f[1][0], f[1][1]
  // as a two dimensional matrix
  abstract double[][] func(double x[]);
}

class fa extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyon
double ff[]=new double[3];
ff[0]=3.0*x[0]-Math.cos(x[1]*x[2])-0.5;
ff[1]=x[0]*x[0]-81.0*(x[1]+0.1)*(x[1]+0.1)+Math.sin(x[2])+1.06;
ff[2]=Math.exp(-x[0]*x[1])+20.0*x[2]+(10.0*Math.PI-3.0)/3.0;
return ff;
}
}

class fb extends fij_xi
{
fi_xi f;

public fb(fi_xi fi)
{f=fi;}

public double[][] func(double x[])
{
  int n=x.length;
  double a[][]=new double[n][n];
  for(int i=0;i<n;i++)
  {
```

```
    for(int j=0;j<n;j++)
    {
       a[i][j]=turev(x,i,j);
    }
  }
return a;
}

public double turev(double x[],int denklem_ref,int x_ref)
{
// fonksiyon f in denklem_ref sayılı fonksiyonunun(sayılar 0 dan başlar)
// x[x_ref} değişkenine göre türevi (sayılar 0 dan başlar)
// df_denklem_ref(x)/d_x_ref
// bu metod newtond metodu içinde kullanılmak içindir.
double h0=0.0256;
int i,m;
int n=7;
double f1[];
f1=new double[x.length];
double f2[];
f2=new double[x.length];
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
x2[i]=x[i];
}
//derivative of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
```

```
x2[x_ref]-=h[i];
f1=f.func(x1);
f2=f.func(x2);
T[i][0]=( f1[denklem_ref] - f2[denklem_ref])/(2.0*h[i]);
x1[x_ref]=x[x_ref];
x2[x_ref]=x[x_ref];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}

}

public class FSCO5C
{
// Süreklilik(Continuity-homotopy) metodu kullanarak denklem sisteminin köklerini bulmak
// Bu metod denklem sistemi çözümü için Matrix.java programını
// kullanmaktadır.


public static double[] multiply(double left,double[] right)
{
//multiplying a vector with a constant
int i;
int n=right.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {b[i]=left*right[i];}
return b;
}

public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right.length;
double[] b;
b=new double[m2];
 if(n1 != m2)
 {
 System.out.println("inner matrix dimensions must agree");
```

```
 for(ii=0;ii<n1;ii++)
   {
    b[ii]=0;
   }
  return b;
 }
 for(i=0;i<m1;i++)
   {
    b[i]=0;
    for(k=0;k<n1;k++)
          b[i]+=left[i][k]*right[k];
   }
return b;
//end of multiply of a matrix and a vector
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
   b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
   b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
}

public static double[][] inv(double[][] a)
{
// INVERSION OF A MATRIX
// inversion by using gaussian elimination
// with full pivoting
int n=a.length;
int m=a[0].length;
double b[][];
b=new double[n][n];
int indxc[];
int indxr[];
double ipiv[];
indxc=new int[n];
```

```
indxr=new int[n];
ipiv=new double[n];
int i,j,k,l,ll,ii,jj;
int icol=0;
int irow=0;
double big,dum,pivinv,temp;
if(n!=m)
{
  System.out.println("Matrix must be square ");
            for(ii=0;ii<n;ii++)
              for(jj=0;jj<n;jj++)
                    b[ii][jj]=0.0;
            return b;
}
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
     b[i][j]=a[i][j];
for(i=0;i<n;i++)
{
 big=0.0;
 for(j=0;j<n;j++)
 {
   if(ipiv[j] != 1)
     for(k=0;k<n;k++)
     {
           if(ipiv[k] == 0)
           {
             if(Math.abs(b[j][k]) >= big)
             {
               big=Math.abs(b[j][k]);
               irow=j;
               icol=k;
             }
           }
           else if(ipiv[k] > 1 )
             {
             System.out.println("error : inverse of the matrix : singular matrix-1");
             for(ii=0;ii<n;ii++)
               for(jj=0;jj<n;jj++)
                     b[ii][jj]=0.0;
             return b;
             }
     }
 }
 ++ ipiv[icol];
 if(irow != icol)
   for(l=0;l<n;l++)
   {
   temp=b[irow][l];
   b[irow][l]=b[icol][l];
   b[icol][l]=temp;
   }
```

```
  indxr[i]=irow;
  indxc[i]=icol;
  if(b[icol][icol] == 0.0)
  {
    System.out.println("error : inverse of the matrix : singular matrix-2");
              for(ii=0;ii<n;ii++)
                for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
    return b;
  }
  pivinv=1.0/b[icol][icol];
  b[icol][icol]=1.0;
  for(l=0;l<n;l++) b[icol][l] *=pivinv;
  for(ll=0;ll<n;ll++)
  if(ll != icol)
  {
    dum=b[ll][icol];
    b[ll][icol]=0.0;
    for(l=0;l<n;l++) b[ll][l]-= b[icol][l]*dum;
  }
}
for(l=n-1;l>=0;l--)
{
  if(indxr[l] != indxc[l])
     for(k=0;k<n;k++)
     {
       temp=b[k][indxc[l]];
       b[k][indxc[l]]=b[k][indxr[l]];
       b[k][indxr[l]]=temp;
     }
}
return b;
}

public static double[] continuationRK4(fi_xi f,fij_xi J,double x[],int N)
{
//=====================================================
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi J de tanımlanacaktır
// Homotopy, continuation (devamlılık) metodu ile lineer olmıyan denklem sistemi çözümü
// 4.derece Runge-Kutta denklemi ile
// yi+1 = yi + (1/6)*( k1 + 2k2 +2k3+k4)h
// k1=f(xi,yi)
// k2=f(xi+0.5h,yi+0.5k1h)
// k3=f(xi+0.5h,yi+0.5k2h)
// k4=f(xi+h,yi +k3h)
//=====================================================
// x  : bağımsız değişken vektörü
// y  : bağımlı değişken vektörü
// dy : bağımlı değişkenin türev fonksiyonları vektörü
// N  : Runge-Kutta diferansiyel denklem noktasayısı
```

```
int i;
int nmax=400;
double tolerance=1.0e-20;
int n=x.length;
double h=1.0/(double)N;
double b[]=new double[n];
double x1[]=new double[n];
double k[][]=new double[4][n];
double A[][]=new double[n][n];
b=multiply(-h,f.func(x));
for(i=0;i<N;i++)
{
x1=x;
A=J.func(x1);
k[0]=multiply(inv(A),b);
x1=add(x,multiply(0.5,k[0]));
A=J.func(x1);
k[1]=multiply(inv(A),b);
x1=add(x,multiply(0.5,k[1]));
A=J.func(x1);
k[2]=multiply(inv(A),b);
x1=add(x,k[2]);
A=J.func(x1);
k[3]=multiply(inv(A),b);
for(int j=0;j<n;j++) {x[j]=x[j]+1.0/6.0*(k[0][j]+2.0*k[1][j]+2.0*k[2][j]+k[3][j]);}
}
return x;
}

public static double[] continuationRK6(fi_xi f,fij_xi J,double x[],int N)
{
//=====================================================
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi J de tanımlanacaktır
// Homotopy, continuation (devamlılık) metodu ile lineer olmıyan denklem sistemi çözümü
// 6. derece Runge-Kutta denklemi ile
// yi+1 = yi + (1/90)*( 7k1 + 32k3 +12k4+32k5+7k6)h
// k1=f(xi,yi)
// k2=f(xi+0.25h , yi+0.25k1h)
// k3=f(xi+0.25h , yi+0.125k1h+0.125k2h)
// k4=f(xi+0.5h , yi - 0.5k2h+k3h)
// k5=f(xi+0.75h , yi + (3/16)k1h+(9/16)k4h)
// k6=f(xi+h , yi - (3/7)k1h+(2/7)k2h+(12/7)k3h - (12/7)k4h+(8/7)k5h)
//=====================================================
// x  : bağımsız değişken vektörü
// y  : bağımlı değişken vektörü
// dy : bağımlı değişkenin türev fonksiyonları vektörü
// N  : Runge-Kutta diferansiyel denklem noktasayısı
int i;
int nmax=400;
double tolerance=1.0e-20;
```

```
int n=x.length;
double h=1.0/(double)N;
double b[]=new double[n];
double x1[]=new double[n];
double k[][]=new double[6][n];
double A[][]=new double[n][n];
b=multiply(-h,f.func(x));
for(i=0;i<N;i++)
{
A=J.func(x);
// k1=f(xi,yi)
k[0]=multiply(inv(A),b);
x1=add(x,multiply(0.25,k[0]));
A=J.func(x1);
// k2=f(xi+0.25h , yi+0.25k1h)
k[1]=multiply(inv(A),b);
x1=add(x,add(multiply(0.125,k[0]),multiply(0.125,k[1])));
A=J.func(x1);
// k3=f(xi+0.25h , yi+0.125k1h+0.125k2h)
k[2]=multiply(inv(A),b);
x1=add(x,add(multiply(-0.5,k[1]),k[2]));
A=J.func(x1);
// k4=f(xi+0.5h , yi - 0.5k2h+k3h)
k[3]=multiply(inv(A),b);
x1=add(x,add(multiply((3.0/16.0),k[0]),multiply((9.0/16.0),k[3])));
A=J.func(x1);
// k5=f(xi+0.75h , yi + (3/16)k1h+(9/16)k4h)
k[4]=multiply(inv(A),b);
x1=add(x,
      add(multiply((-3.0/7.0),k[0]),add(multiply((2.0/7.0),k[1]),
      add(multiply((12.0/7.0),k[2]),
         add(multiply((-12.0/7.0),k[3]),multiply((8.0/7.0),k[4])))));
A=J.func(x1);
// k6=f(xi+h , yi - (3/7)k1h+(2/7)k2h+(12/7)k3h - (12/7)k4h+(8/7)k5h)
k[5]=multiply(inv(A),b);
// yi+1 = yi + (1/90)*( 7k1 + 32k3 +12k4+32k5+7k6)h
for(int j=0;j<n;j++) {x[j]=x[j]+1.0/90.0*(7.0*k[0][j]+32.0*k[2][j]+12.0*k[3][j]+32.0*k[4][j]+7.0*k[5][j]);}
}
return x;
}
public static double[] newton_continuationRK4(fi_xi f,fij_xi df,double x[])
{
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi dfj/dxi de tanımlanacaktır
// Newton-Raphson metodu ile lineer olmıyan denklem sistemi çözümü
//ti :önem katsayısı
//x bağımsız değişken vektörü
//y bağımlı değişken vektörü
//dy bağımlı değişkenin türev fonksiyonları vektörü
x=continuationRK4(f,df,x,4);
double ti=1.0;
```

```
int i;
int nmax=400;
double tolerance=1.0e-10;
int n=x.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
{
b[i]=1.0;
}
i=0;
while( i++ < nmax && Matrix.abs(b) > tolerance )
{ b=Matrix.multiply(Matrix.divide(f.func(x),df.func(x)),-ti);
  x=Matrix.add(x,b);
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
            " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
System.out.println();
return x;
}
public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
   Double ax=new Double(token.nextToken());
   a[j++]=ax.doubleValue();
   }
   return a;
}
    public static void main (String args[])
    {
            fa f =new fa();
    fb df=new fb(f);
    double [] x0=verigir();
    double [] r= continuationRK4(f,df,x0,4);
    double [] r1= continuationRK6(f,df,x0,4);
    double [] r2= newton_continuationRK4(f,df,x0);
    String s=" kök değerleri süreklilik RK4 : \n"+Matrix.toStringT(r);
    s+=" kök değerleri süreklilik RK6 : \n"+Matrix.toStringT(r1);
    s+=" kök değerleri newton-süreklilik RK4 : \n"+Matrix.toStringT(r2);
    String s1="çok değişkenli süreklilik metodu ile kök bulma : \n";
    System.out.println(s1+s);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO5C

çok değişkenli süreklilik metodu ile kök bulma :
 kök değerleri süreklilik RK4 :
      0.50000000000000000000000000
     -0.00000000000000000052804416
     -0.52359877559829870000000000
 kök değerleri süreklilik RK6 :
      0.50000000000000000000000000
     -0.00000000000000000052804416
     -0.52359877559829870000000000
 kök değerleri newton-süreklilik RK4 :
      0.50000000000000000000000000
     -0.00000000000000000010644621
     -0.52359877559829890000000000

> Terminated with exit code 0.
```

Süreklilik denklem çözüm yöntemini Newton yöntemi gibi bir yöntem için ilk değer tayini için de kullanabiliriz.

4.4 Optimizasyon yöntemleri : Nelder-Mead amip yöntemi

Bir adaptasyon fonksiyonu kullanarak kök çözüm problemini optimizasyon problemi haline getirebiliriz, ve optimizasyon problemini çözerek köklere ulaşabiliriz. Bu alt bölümde temelde bu yöntemin çeşitli optimizasyon metodlarına adapte ederek uygulanmasını inceleyeceğiz.

$f_i(x_j)=0$ denklem sistemini çözmek için

$$g(x_j) = \sum_{i=0}^{n\_denklem} f_i(x_j) * f_i(x_j)$$

denkleminin minimum değeri bulunmuştur. Aslında bir çok non-lineer denklem sisteminin geometrik yöntemlerle optimizasyon probleminin çözümü ile kök bulma metodları iç içedir. Çünki optimizasyon (minimum veya maksimum bulma işlemi) fonksiyonun türevinin kökünü bulma işlemi ile aynıdır. Nelder -Mead ya da amip yöntemi detaylarıyla optimizasyon konusunda (bölüm 5) verilmiştir. Lütfen detayları için ilgili bölümümüze bakınız. Burada verilen program da Nelder mead programının sadece çıktı kısmını içermektedir. Program detayları için lütfen OPO7B.java programını inceleyiniz**(Program 5.9.1)**

Program 4.4.1 Nelder-Mead optimizasyon yöntemi kullanarak çok boyutlu lineer olmayan denklem çözümü (köklerini bulma)

¶

```
 // Nelder mead metodu ile denklem sistemi çözümü
//
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;



class f1 extends f_xj
```

```
{//adaptör sınıfı
 //optimizasyon problemleri kullanarak lineer olmıyan denklem sistemi kökü çözer
fi_xi ff1;
public f1(fi_xi ffi) {ff1=ffi;}

public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0.0;
double fa[]=ff1.func(x);
for(int i=0;i<fa.length;i++)  ff+=fa[i]*fa[i];
return ff;
}
}


class fa extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyon
double ff[]=new double[3];
ff[0]=3.0*x[0]-Math.cos(x[1]*x[2])-0.5;
ff[1]=x[0]*x[0]-81.0*(x[1]+0.1)*(x[1]+0.1)+Math.sin(x[2])+1.06;
ff[2]=Math.exp(-x[0]*x[1])+20.0*x[2]+(10.0*Math.PI-3.0)/3.0;
return ff;
}
}

public class OPO7BDS
{
        public static void main (String args[])
    {
        String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=OPO7B.verigir(s);
    double [] dx0=new double[x0.length];
    for(int i=0;i<x0.length;i++) dx0[i]=0.2*x0[i];
    fa ff=new fa();
    f1 f=new f1(ff);
    double [] r1= OPO7B.nelder(f,x0,dx0);
    s=" çözüm seti  : \n"+Matrix.toStringT(r1);
    String s2="Nelder-Mead metodu ile lineer olmıyan çok değişkenli fonksiyon optimizasyonu:\n";
    System.out.println(s2+s);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO7BDS
Nelder-Mead metodu ile lineer olmıyan çok değişkenli fonksiyon optimizasyonu:
 çözüm seti  :
      0.50000000000000000000000000
```

```
      0.00000000000000000000000000
     -0.52359877559829890000000000

> Terminated with exit code 0.
```

4.5 Optimizasyon yöntemleri : genetik algoritma yöntemi

Genetik algoritmalar bir optimizasyon yöntemi olarak detaylı bir şekilde anlatılmıştır. Bu detayları bölüm 5.24 de bulabilirsiniz. Burada sadece bu yöntemin istatistik tabanlı bir yöntem olduğunu ve çözüm setinde istatistiksel sapmalar bekleyebileceğimizi belirtmekle yetineceğiz. Önceki bölümlerimizde kullandığımız aynı örnek problemi bu bölümde de kullandık, çözüm setinin (0.499502180107804, -0.000026702982723, -0.523405164356859) çok kötü olmadığını söyleyebiliriz. Orijinal fonksiyonu – ile çarpıp 10 ilave ettiğimize dikkatinizi çekeriz. Genetik algoritmalar fonksiyonun maksimimunu bulurlar ve fonksiyonun positif değerleri için sonuç verirler.

Program 4.20-1 Genetik algoritma optimizasyon yöntemi kullanarak çok boyutlu lineer olmayan denklem çözümü (köklerini bulma)

```
import java.io.*;

class f1 extends f_xj
{//adaptör sınıfı
 //optimizasyon problemleri kullanarak lineer olmıyan denklem sistemi kökü çözer
fi_xi ff1;
public f1(fi_xi ffi) {ff1=ffi;}

public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0.0;
double fa[]=ff1.func(x);
for(int i=0;i<fa.length;i++)  ff+=fa[i]*fa[i];
return 10-ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}



class fa extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyon
double ff[]=new double[3];
ff[0]=3.0*x[0]-Math.cos(x[1]*x[2])-0.5;
ff[1]=x[0]*x[0]-81.0*(x[1]+0.1)*(x[1]+0.1)+Math.sin(x[2])+1.06;
ff[2]=Math.exp(-x[0]*x[1])+20.0*x[2]+(10.0*Math.PI-3.0)/3.0;
return ff;
}
}
```

```
class OPO17DS
{
   public static void main(String args[]) throws IOException
   {
   int N=18;
  int N=24;
   int POPSIZE=500;
   int MAXGENS=1000;
   int NVARS=3;    double PMUTATION=0.05;
    fa ff=new fa();
    f1 f_x=new f1(ff);
   Genetic1 xx=new Genetic1(POPSIZE,MAXGENS,NVARS,N,PXOVER,PMUTATION);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=-1.0;high[0]=1.0;
   low[1]=-1.0;high[1]=1.0;
   low[2]=-1.0;high[2]=1.0;
   xx.setPopulation(low,high,f_x);
   System.out.println("best=\n"+Matrix.toStringT(xx.calculate(f_x,false)));
   }
}
```

Çıktı 4.20-1 Genetik algoritma optimizasyon yöntemi kullanarak çok boyutlu lineer olmayan denklem çözümü (köklerini bulma)

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO17DS
best=
       0.50002321601052380000000000
      -0.00004500150948771875000000
      -0.52358243009939380000000000
       9.99999285947870000000000000


> Terminated with exit code 0.
```

4.6 Optimizasyon yöntemleri : en dik yamaç metodu

Optimizasyon metodlarından olan en dik yamaç metodunun detayları optimizasyon konusunda bölüm 5.20 de verilmiştir. Burada detaylarına girmeyeceğiz. Bir önceki bölümde de belirttiğimiz gibi bu metod da lineer olmıyan denklem sistemlerinin köklerini bulmada kullanılabilir.

Program 4.21-1 en dik yamaç optimizasyon yöntemi kullanarak çok boyutlu lineer olmayan denklem çözümü (köklerini bulma)

```
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import java.io.*;
```

```
class f1 extends f_xj
{//adaptör sınıfı
 //optimizasyon problemleri kullanarak lineer olmıyan denklem sistemi kökü çözer
fi_xi ff1;
public f1(fi_xi ffi) {ff1=ffi;}

public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0.0;
double fa[]=ff1.func(x);
for(int i=0;i<fa.length;i++)  ff+=fa[i]*fa[i];
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}


class fa extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyon
double ff[]=new double[3];
ff[0]=3.0*x[0]-Math.cos(x[1]*x[2])-0.5;
ff[1]=x[0]*x[0]-81.0*(x[1]+0.1)*(x[1]+0.1)+Math.sin(x[2])+1.06;
ff[2]=Math.exp(-x[0]*x[1])+20.0*x[2]+(10.0*Math.PI-3.0)/3.0;
return ff;
}
}

class f1dim extends f_x
{
  // iki boyutlu fonksiyonu tek boyutlu fonksiyona çeviren
  // pseudo fonksiyon
  f_xj ff1;
  double pc[];
  double xc[];
  double xt[];
  int n;
  public f1dim(f_xj ff,double pi[],double xi[])
  { n=pi.length;
    ff1=ff;
          pc=new double[n];
    xc=new double[n];
    xt=new double[n];
          for(int i=0;i<n;i++)
            {pc[i]=pi[i];xc[i]=xi[i];}
  }

  public double func(double x)
  {
          for (int j=0;j<n;j++) xt[j]=pc[j]+x*xc[j];
```

```
            return ff1.func(xt);
   }
}

public class OPO8DS
{
    public static void main (String args[])
    {
            String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=OPO8.verigir(s);
    fa ff=new fa();
    f1 f_x=new f1(ff);
    double [] r1= OPO8.endikyamac(f_x,x0);
    s=" çözüm seti  : \n"+Matrix.toStringT(r1);
    String s2="En dik yamaç lineer olmıyan çok değişkenli fonksiyon optimizasyonu:";
    JOptionPane.showMessageDialog(null,s,s2,JOptionPane.PLAIN_MESSAGE);
    System.exit(0);
    }
}
```

4.7 Optimizasyon yöntemleri : Davidon-Fletcher-Powell metodu

Davidon-Fletcher-Powell metodu optimizasyon metodundan çok lineer olmıyan denklem sistemi çözüm metodu olarak düşünülebilir,ancak burada da tek boyutlu optimizasyondan yararlanıldığından bu metodu da bu bölümde guruplayabiliriz. Metod detaylarını bölüm 5.23 de bulabilirsiniz. Bu örnek problemde de Bu bölümdeki Program 5.23.1 deki OPO12F.java probleminden adapte edilmiştir. Burada okuyucu haklı olarak böyle bir bölümün optimizasyon bölümünden sonra sunulmasını talep edebilir, ancak aralarındaki zincirleme ilişki hangisinin daha temel olduğunu sınıflamada zorluk yaratmaktadır. Örneğin Newton lineer olmıyan kök bulma metodu bire bir Newton lineer olmıyan optimizasyon metodunun temelini oluşturmaktadır.

Program 4.7.1 Davidon-Fletcher-Powell optimizasyon yöntemi kullanarak çok boyutlu lineer olmayan denklem çözümü (köklerini bulma)

```
 import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class f1b extends f_xj
{//adaptör sınıfı
 //optimizasyon problemleri kullanarak lineer olmıyan denklem sistemi kökü çözer
fi_xi ff1;
public f1b(fi_xi ffi) {ff1=ffi;}

public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0.0;
```

```
double fa[]=ff1.func(x);
for(int i=0;i<fa.length;i++)  ff+=fa[i]*fa[i];
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}


class fa extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyon
double ff[]=new double[3];
ff[0]=3.0*x[0]-Math.cos(x[1]*x[2])-0.5;
ff[1]=x[0]*x[0]-81.0*(x[1]+0.1)*(x[1]+0.1)+Math.sin(x[2])+1.06;
ff[2]=Math.exp(-x[0]*x[1])+20.0*x[2]+(10.0*Math.PI-3.0)/3.0;
return ff;
}
}

public class OPO12FDS
{

        public static void main (String args[])
    {
        String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=OPO12F.verigir(s);
     fa ff=new fa();
    f1b f=new f1b(ff);
    double [] r1= OPO12F.davidon_fletcher_powell(f,x0);
    double [] r2= OPO12F.davidon_fletcher_powell_V1(f,x0);
    s=" çözüm seti DFP : \n"+Matrix.toStringT(r1);
    s+=" çözüm seti düzeltilmiş DFP : \n"+Matrix.toStringT(r2);
    String s2="Davidon - Fletcher - Powell çok değişkenli fonksiyon optimizasyonu:";
    System.out.println(s2+s);
    System.exit(0);
    }
}
```

```
Davidon - Fletcher - Powell çok değişkenli fonksiyon optimizasyonu: çözüm seti DFP :
       0.50000000000000000000000000
      -0.00000000000000000243896983
      -0.52359877559829890000000000
 çözüm seti düzeltilmiş DFP :
       0.50000000000000000000000000
      -0.00000000000000000243896983
      -0.52359877559829890000000000

> Terminated with exit code 0.
```

4.8 Jacobi, Gauss-Seidel ve rahatlamalı interpolasyon iteratif metodları

İteratif methodları lineer denklem sistemleri için görmüştük. Temel olarak aynı iteratif metodların lineer olmıyan denklem sistemleri için de kulanılmaları mümkündür. Ancak non linear denklem sistemlerinin bu yöntemle yakınsama garantisi mevcut değildir. Bu metodda önce denklem

$f_j(x_i) = 0 \quad i = 0..n \quad j = 0..n$ formundan

$x_j = g_j(x_i) \quad i = 0..n \quad j = 0..n$ or in an open form:

$x_0 = g_0(x_0, x_1, x_2, .., x_n)$

$x_1 = g_1(x_0, x_1, x_2, .., x_n)$

...

$x_n = g_n(x_0, x_1, x_2, .., x_n)$

formuna dönüştürülür. Bu formdan da iteratif form olan

$x_0^{k+1} = (1-\alpha)x_0^k + \alpha(g_0(x_0^k, x_1^k, x_2^k, .., x_n^k)$

$x_1^{k+1} = (1-\alpha)x_1^k + \alpha(g_1(x_0^{k+1}, x_1^k, x_2^k, .., x_n^k)$

$x_2^{k+1} = (1-\alpha)x_1^k + \alpha(g_2(x_0^{k+1}, x_1^{k+1}, x_2^k, .., x_n^k)$

$x_3^{k+1} = (1-\alpha)x_3^k + \alpha(g_3(x_0^{k+1}, x_1^{k+1}, x_2^{k+1}, x_3^k, .., x_n^k)$

..

$x_j^{k+1} = (1-\alpha)x_j^k + \alpha(g_j(x_0^{k+1}, x_1^{k+1}, x_2^{k+1}, .., x_{j-1}^{k+1}, x_j^k, .., x_n^k)$

.....

$x_n^{k+1} = (1-\alpha)x_n^k + \alpha(g_j(x_0^{k+1}, x_1^{k+1}, x_2^{k+1}, .., x_{n-1}^{k+1}, x_n^k)$

Denklemdeki α değeri 0 ile arasında olabileceği gibi 1 den büyük de olabilir. Örnek problem olarak

$3x_0 - \cos(x_1 x_2) - 0.5 = 0;$

$x_0^2 - 81(x_1 + 0.1) + \sin(x_2) + 1.06 = 0$

$-\exp(-x_0 x_1) + 20x_2 + (10\pi - 3)/3) = 0$

denklem sisteminin köklerini bulmaya çalışalım. İlk önce her denklemden x değerlerini çekerek yeni formuna dönüştürelim.

$x_0 = (\cos(x_1 x_2) + 0.5)/3.0;$

$x_1 = (x_0^2 + \sin(x_2) + 1.06)/81)^2 + 0.01$

$x_2 = (-\exp(-x_0 x_1) - (10\pi - 3)/3))/20$

Aşağıda bu metodun java dilinde yazılmış programı görülmektedir.

Program 3.7-1 Jacobi-Gauss_seidel ve rahatlamalı interpolasyon formülünün lineer olmıyan fonksiyon versiyonu

```
 import java.io.*;
class fa extends fi_xi
{
public double[] func(double x[])
{
// çözümü istenen fonksiyon
double g[]=new double[3];
g[0]=(Math.cos(x[1]*x[2])+0.5)/3.0;
```

```
g[1]=Math.sqrt((x[0]*x[0]+Math.sin(x[2]+1.06))/81.0)-0.1;
g[2]=(-Math.exp(-x[0]*x[1])-(10.0*Math.PI-3)/3.0)/20.0;
return g;
}
}

class SCO10X
{
  public static double [] gauss_seidel(fi_xi f,double x[],double alpha)
  {
  int n=x.length;
  double xx[]=new double[n];
  double xx1[]=new double[n];
  double delta=0;
  for(int i=0;i<n;i++) xx[i]=x[i];
  double g[]=new double[n];
  int iter=500;
  for(int k=0;k<iter;k++)
  {for(int i=0;i<n;i++) xx1[i]=xx[i];
   delta=0;
  for(int j=0;j<n;j++)
  {   g=f.func(xx);
                xx[j]=(1-alpha)*xx[j]+alpha*g[j];
                delta+=Math.abs(xx[j]-xx1[j]);
                System.out.print("xx["+j+"] = "+xx[j]);
  }
  System.out.println(" k = "+k+" delta = "+delta);
  if(delta<1e-15) break;
  }
  return xx;
  }


  public static void main(String args[]) throws IOException
  {
  double x[]={1.01,0.1,1.0121};
  fa ff=new fa();
  double alpha=0.1;
  Text.print(gauss_seidel(ff,x,alpha));
  }
}
```

Çıktı 3.7-1 Gauss_seidel rahatlamalı interpolasyon formülü

```
 ---------- Capture Output ----------
> "C:\java\bin\java.exe" SCO10X
xx[0] = 0.9588294212826883xx[1] = 0.09489183495398086xx[2] = 0.8589649657886227 k = 0 delta = 0.20941377797470812
xx[0] = 0.9128358124735219xx[1] = 0.09019866496669672xx[2] = 0.7211037819655458 k = 1 delta = 0.1885479626195275
xx[0] = 0.8714817470867342xx[1] = 0.085824605234896xx[2] = 0.5969938557799842 k = 2 delta = 0.16983805130414997
xx[0] = 0.8342898285193656xx[1] = 0.08169652937067469xx[2] = 0.4852640310069166 k = 3 delta = 0.1530498192046575
xx[0] = 0.8008346544741505xx[1] = 0.07776042319429945xx[2] = 0.3846796198127338 k = 4 delta = 0.1379756914157732
xx[0] = 0.7707362771620705xx[1] = 0.07397773129037596xx[2] = 0.2941288917316395 k = 5 delta = 0.12443179729709779
xx[0] = 0.743654758864458xx[1] = 0.07032203418833874xx[2] = 0.2126108822145333 k = 6 delta = 0.11225522491675591
xx[0] = 0.7192855573929079xx[1] = 0.06677618815104167xx[2] = 0.1392243959352476 k = 7 delta = 0.10130153378813286
xx[0] = 0.6973555611327479xx[1] = 0.06332995056219265xx[2] = 0.07315809120795819 k = 8 delta = 0.09144253857629844
xx[0] = 0.6776196472601156xx[1] = 0.059978059978701904xx[2] = 0.013681541973005897 k = 9 delta = 0.08256435369107533
xx[0] = 0.6598576713112414xx[1] = 0.05671871668388067xx[2] = -0.039862816917781055 k = 10 delta = 0.07456567813448237
…….
xx[0] = 0.4999995683317302xx[1] = -0.003073182441921277xx[2] = -0.5236756641509748 k = 388 delta = 8.673617379884035E-19
xx[0] = 0.4999995683317302xx[1] = -0.003073182441921278xx[2] = -0.5236756641509748 k = 389 delta = 8.673617379884035E-19
xx[0] = 0.4999995683317302xx[1] = -0.003073182441921279xx[2] = -0.5236756641509748 k = 390 delta = 8.673617379884035E-19
xx[0] = 0.4999995683317302xx[1] = -0.003073182441921279 8xx[2] = -0.5236756641509748 k = 391 delta = 8.673617379884035E-19
xx[0] = 0.4999995683317302xx[1] = -0.003073182441921280 6xx[2] = -0.5236756641509748 k = 392 delta = 8.673617379884035E-19
xx[0] = 0.4999995683317302xx[1] = -0.003073182441921281 5xx[2] = -0.5236756641509748 k = 393 delta = 8.673617379884035E-19
xx[0] = 0.4999995683317302xx[1] = -0.003073182441921282 4xx[2] = -0.5236756641509748 k = 394 delta = 8.673617379884035E-19
xx[0] = 0.4999995683317302xx[1] = -0.003073182441921282 4xx[2] = -0.5236756641509748 k = 395 delta = 0.0
```

4.9 Problemler

PROBLEM 1

$4x_1^2-20x_1+0.25x_2^2+8=0$
$0.5x_1x_2^2+2x_1+8=0$

denklem sisteminin köklerini
a) Newton raphson yöntemi
b) Süreklilik yöntemi
c) Nelder-Mead yöntemi
d) en dik yamaç yöntemi
e) Genetik algoritma yöntemlerini kullanarak bulunuz, sonuçları karşılaştırınız.

PROBLEM 2

$3x_1^2-\cos(x_2x_2) - 0.5=0$
$4x_1^2-6.25x_2^2+2x_3-1=0$
$e^{x1\ x2} +20x_3+(10\pi-3)/3=0$

denklem sisteminin köklerini
a) Newton raphson yöntemi
b) Broyden kiriş yöntemi
c) Süreklilik yöntemi
d) Nelder-Mead yöntemi
e) en dik yamaç yöntem i
f) Genetik algoritma yöntemlerini kullanarak bulunuz, sonuçları karşılaştırınız.

PROBLEM 3

$x_1(1- x_1) + 4x_3 – 12 =0$
$(x_1-2)^2+(2x_2-3)^2-25 =0$

lineer olmıyan denklem sisteminin köklerini
a) Newton raphson yöntemi
b) Süreklilik yöntemi
c) Nelder-Mead yöntemi
d) en dik yamaç yöntemi
e) Genetik algoritma yöntemlerini kullanarak bulunuz, sonuçları karşılaştırınız.

PROBLEM 4

$\sin(4\pi x_1x_2)-2x_2-x_1=0$

$$\left(\frac{4\pi-1}{4\pi}\right)(e^{2x_1}-e)+4ex_2^2-2ex_1=0$$

lineer olmıyan denklem sisteminin köklerini
a) Newton raphson yöntemi
b) Süreklilik yöntemi
c) Nelder-Mead yöntemi
d) en dik yamaç yöntemi
e) Genetik algoritma yöntemlerini kullanarak bulunuz, sonuçları karşılaştırınız.

PROBLEM 5

$10x_1-2x_2^2+x_2-2x_3-5=0$
$8x_2^2+4x_3^2-9=0$
$8x_2x_3+4=0$

lineer olmıyan denklem sisteminin köklerini
a) Newton raphson yöntemi
b) Süreklilik yöntemi
c) Nelder-Mead yöntemi
d) en dik yamaç yöntemi
e) Genetik algoritma yöntemlerini kullanarak bulunuz, sonuçları karşılaştırınız.

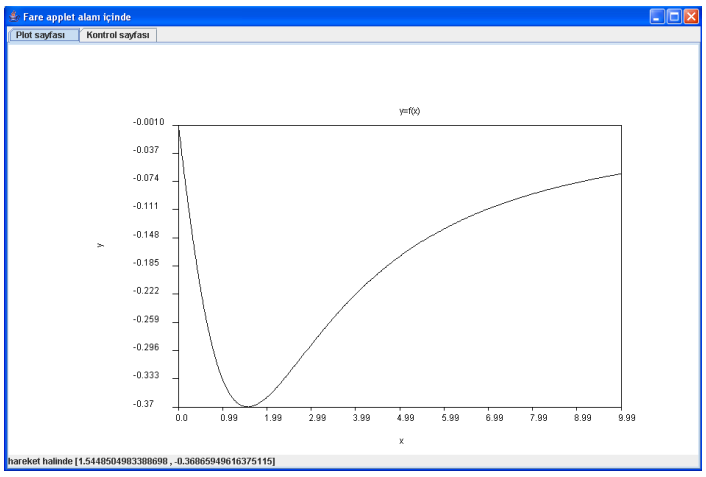
Bölüm 5 Optimizasyon(minimizasyon ve maksimizasyon)

6.0 Optimizasyonun tanımlanması ve grafik yöntemle tek boyutlu optimizasyon

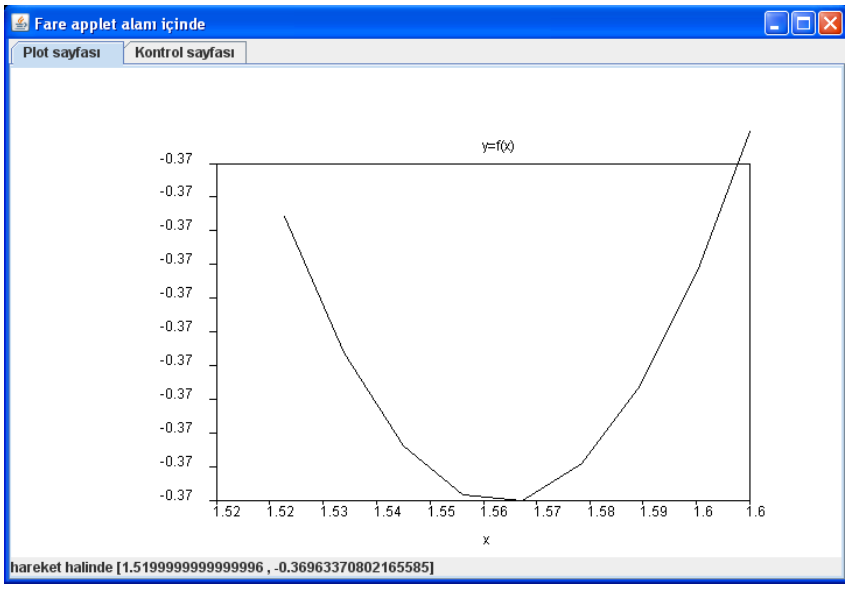
Optimizasyon minimizasyon ve maksimizasyon işlemlerine verilen genel bir isimdir. Temel matematik olarak baktığımızda tek boyutlu f(x) fonksiyonu için
df(x)/dx = 0 f(x) fonksiyonunun minimum veya maksimumunu tanımlar. Temel tanım olarak minimum veya maksimum arasında işlemsel olarak bir fark yoktur, çünki
maksimum (f(x)) = minimum(-f(x)) olarak yazılabilir. Bu yüzden sadece fonksiyonu -1 ile çarparak maksimum veya minimum bulma işlemleri arasına yer değiştirmek mümkündür. Tek boyutlu minimizasyon işlemine ilk örnek olarak grafik çizme yöntemini verelim. Bir fonksiyonun grafiğini çizerek (sabit aralıklarla fonksiyon değerini hesaplayarak) minimumun yeri çizilen grafiknoktalarının arasındaki boşluk hassasiyeti ile saptanabilir. Minimum noktaya yaklaştığınızda yeni daha küçük bir aralık saptıyarak fonksiyonu tekrar bu aralık için tarayabiliriz.

Program 5.0.1 2.0*c/(4.0+0.8*c+c*c+0.2*c*c*c fonksiyonu maksimumu veya –(2.0*c/(4.0+0.8*c+c*c+0.2*c*c*c) fonksiyonu minimumunun grafik yöntemle bulunması

```
import java.io.*;
class fa1 extends f_x
{
  double func (double c)
  { return -(2.0*c/(4.0+0.8*c+c*c+0.2*c*c*c)); }}

class PlotT02
{ public static void main (String args[]) throws IOException
    {
    fa1 ff1=new fa1();
    Plot pp=new Plot(ff1,0.0,10.0,400,0);
    pp.plot();
    }}
```



(bolum5_000.jpg,Resimaltı: **1 2.0*c/(4.0+0.8*c+c*c+0.2*c*c*c fonksiyonu maksimumu veya –(2.0*c/(4.0+0.8*c+c*c+0.2*c*c*c) fonksiyonu minimumunun grafik yöntemle bulunması**)



(bolum5_001.jpg,Resimaltı: **2.0*c/(4.0+0.8*c+c*c+0.2*c*c*c fonksiyonu maksimumu veya –(2.0*c/(4.0+0.8*c+c*c+0.2*c*c*c) fonksiyonu minimumunun grafik yöntemle bulunmasında detaya inilmesi**)

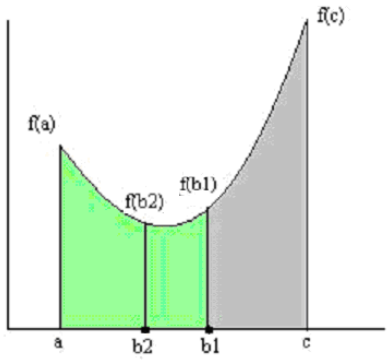
Program 5.0.1 de **-1 2.0*c/(4.0+0.8*c+c*c+0.2*c*c*c)** fonksiyonunun grafiği çizdirilmektedir. Grafik programımız farenin koordinatlarını da yansıttığından farereyi minimumun üzerine getirerek ekrandan değeri detaylı olarak okuyabiliriz. Grafik kontrol sayfasıdan x ve y eksenlerinin minumum ve maksimum değerlerini elle değiştirerek optimum noktası civarını daha detaylı çizdirerek noktayı okuma kolaylığı da elde edebiliriz. Grafik yöntemini çok daha büyük adımlarla bir tarama olarak kullanırsak optimumun olduğu bölgeyi saptamak için de kullanabiliriz.

5.1 Bir bilinmiyenli-lineer olmayan-geometrik-altın oran (Fibonacchi) arama

0,1,1,2,3,5,8,13,21,34,… serisine Fibonacci serisi ismi verilir. Genel yazılımı
F(n)=F(n-1)+F(n-2) şeklindedir. F(0)=0, F(1)=1 şeklide tanımlanmıştır. Fibonacci serisinin büyük oranlarında

$$R=\lim_{n\to\infty}\frac{F(n-1)}{F(n)}=\frac{\sqrt{5}-1}{2}=0.618033989$$

Peki bu oranı optimizasyonda nasıl kullanacağız? Verilen bir (a,c) bölgesinin hangi tarafında bir minimum(veya maksimum) olduğunu kestirebilmek için bu bölgedeki iki ara noktayı değerlendirmemiz gerekir. Minimum arıyorsak iki ara nokta bölgeyi 3 parçaya böleceği için minimum değerin saptandığı noktanın iki tarafındaki bölgeler kalır diğer bölgeyi atarız. Ancak yeni değerlendirmede yine iki ara noktaya ihtiyaç duyulur. Kendimize daha önce hesaplanan ara noktalardan biri tekrar yeni hesapta kullanılamazmı sorusunu sorarsak yanıtımız ara noktaların seçiminde altın oran kullanımı olacaktır. Şekil 5.1-1 de görüldüğü gibi a ve c noktaları arasında minimum değeri veren noktayı bulmaya çalışalım bölgenin uzunluğu d0=(c – a) olacaktır.
b1 noktasını b1=a+d0*R
b2 noktasını b2 = c – d0*R = a + d0*R^2
olarak alır isek ve minimumun b1 noktasında olduğunu kabul edersek (b2-c) aralığı iptal olacaktır.
yeni aralığımız d1=d0*R olacağından
b1 noktasını b1=a+d1*R=a+d0*R^2 olarak alırız bu bir önceki b2 noktasının aynısıdır.
b2 noktasını b2 = c – d1*R = a + d1*R^2 = a + d0*R^3 olarak alırız.
Bir nokta aynı olduğundan yeni eleme için sadece bir fonksiyon değerlendirmesine ihtiyacımız olacaktır. ve her yeni elemede noktaların biri (sağ veya soldan elememize göre değişerek) daima aynı kalacaktır.



(bolum5_002.jpg,Resimaltı:Altın oran minimum arama prensibi)

Program 5.1-1 Altın oran arama programı (minimizasyon için)

```
 /// OPTIMİZASYON
//  ÖDEV 1
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class fb extends f_x
{
public double func(double x)
   {return x+1/x;}
}

public class FOPO1
{
public static double golden(f_x f,double a,double b,double epsilon)
{return golden(f,a,b,epsilon,1.0e-10,0);}
public static double golden(f_x f,double a,double b)
{return golden(f,a,b,1.0e-10,1.0e-10,0);}

public static double golden(f_x f,double a,double b,double epsilon,double delta,int print)
{
//  f fonksiyonunun minimumunu bulur
// not  maximum f(x) = minimum (-f(x))
    double r1 = (Math.sqrt(5.0)-1.0)/2.0; // golden ratio
    double r2 = r1*r1;
    double h = b - a;
    double ya = f.func(a);
    double yb = f.func(b);
    double c = a + r2*h;
    double d = a + r1*h;
    double yc = f.func(c);
    double yd = f.func(d);
    int k = 1;
    double dp,dy,p,yp;
    while ((Math.abs(yb-ya)>epsilon) || (h>delta))
    {
      k++;
```

```
      if (yc<yd)
       {
       b = d;
       yb = yd;
       d = c;
       yd = yc;
       h = b - a;
       c = a + r2 * h;
       yc = f.func(c);
       }
      else
       {
        a = c;
        ya = yc;
        c = d;
        yc = yd;
        h = b - a;
        d = a + r1 * h;
        yd = f.func(d);
       }//end of if
    }//end of while
    dp = Math.abs(b-a);
    dy = Math.abs(yb-ya);
    p = a;
    yp = ya;
    if (yb<ya)
    {
     p = b;
     yp = yb;
    }
    if(print==1)
    {System.out.println("x min = "+p+"ymin = "+yp+"error of x ="+dp+"error of y"+dy); }
    return p;
}

     public static void main (String args[])
     {
             fb f=new fb();
     double a,b;
     a=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun alt sınırını giriniz a : "));
     b=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun üst sınırını giriniz b : "));
     double r;
     r= golden(f,a,b,1.0e-10,1.0e-8,0);
     System.out.println("Fibonacchi optimizasyon değeri  : "+r+"\nFonksiyon değeri : "+f.func(r));
     System.exit(0);
     }
}
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FOPO1
Fibonacchi optimizasyon değeri  : 1.0000000087402612
Fonksiyon değeri : 2.0
```

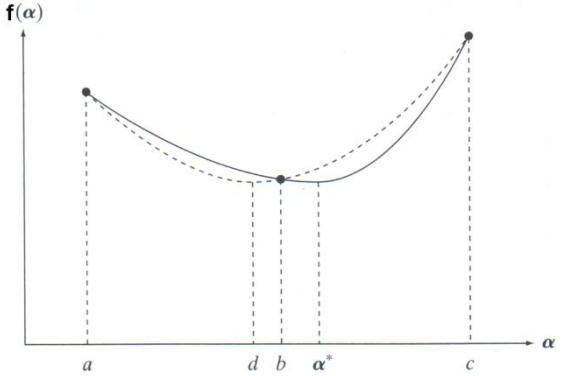
```
> Terminated with exit code 0.
```

5.2 Bir bilinmiyenli-lineer olmayan-geometrik-ikinci derece polinom metodları

Altın arama metodu lineer bir arama metodudur, her zaman işimize yarar ancak çözüme ulaşması göreceli olarak yavaştır. Çözüme ulaşma hızını arttırmak ister isek ikinci dereceden bir arama yöntemine başvurabiliriz. 3 noktadan ikinci dereceden bir polinom geçmesi kuralını kök bulma bölümünde de kullanmıştık. Benzer bir yaklaşımla verilen 3 noktadan (a b ve c noktaları olsun) geçen parabolün minimum(veya maksimum) noktasını parabol denkleminin çözümünden elde edebiliriz. Parabolün minimum noktası

$$d = b + 0.5\left(\frac{(c-b)^2[f(a)-f(b)]-(a-b)^2[f(c)-f(a)]}{(c-b)\,[f(a)-f(b)]-(a-b)\,[f(c)-f(a)]}\right)$$

denklemini kullanarak elde edilebilir. D nokasını bulduktan sonra b noktası yerine d noktasınıalarak yeni bir iterasyon için hesabımıza devam ederiz.



(bolum5_003.jpg,Resimaltı: ikinci derece polinom metodu ile minimum bulunması)

Program 5.2.1 ikinci derece polinom optimizasyonu

```
 /// OPTIMİZASYON
//  2 derece polinom interpolasyon kullanarak optimizasyon
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class fb extends f_x
{
   public double func (double x)
   { return x+1/x;}
}

public class FOPO2
{
public static double polinom2derece(f_x f,double x0,double x2)
{double x1=(x0+x2)/2.0;
 return polinom2derece(f,x0,x1,x2);
}
```

```
public static double polinom2derece(f_x f,double x0,double x1,double x2)
{
double epsilon=1.0e-10;
double delta=1.0e-5;
int print=0;
return polinom2derece(f,x0,x1,x2,epsilon,delta,print);
}
public static double polinom2derece(f_x f,double x0,double x1,double x2,double epsilon,double delta,int print)
{
int maxit=100;
double f0 = f.func(x0);
   double f1 = f.func(x1);
   double f2 = f.func(x2);
   double f3 = f2;
   double x3 = 0;
   double h = x1 - x0;
   double k=1;
   double dd=Math.abs(f1-f0);
   while ((dd >epsilon) || (h>delta))
   {
     k++;
     x3=(f0*(x1*x1-x2*x2)+f1*(x2*x2-x0*x0)+f2*(x0*x0-x1*x1))/
          (2*f0*(x1-x2)+2.0*f1*(x2-x0)+2.0*f2*(x0-x1));
     f3=f.func(x3);
     if(x3 >= x0 && x3<x1)
       {x2=x1;f2=f1;x1=x3;f1=f3;}
     else if(x3 >= x1 && x3 <x2)
       {x0=x1;f0=f1;x1=x3;f1=f3; }
     else if(x3 >x2)
       {x0=x1;f0=f1;x1=x2;f1=f2;x2=x3;f2=f3; }
     else if(x3 < x0)
       {x0=x3;f0=f3;x1=x0;f1=f0;x2=x1;f2=f1; }
     dd=Math.abs(f1-f0);
     h=Math.abs(x1-x0);
     if(k>maxit) break;
   }//end of while
   if(print==1)
   {System.out.println("x  = "+x3+"f = "+f3+"hata x ="+h+"hata f(x)=y "+dd); }
   return x3;
}

    public static void main (String args[])
    {
            fb f=new fb();
    double x0,x1,x2;
    x0=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun alt sınırını giriniz  x0 : "));
    x1=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun orta sınırını giriniz x1 : "));
    x2=Double.parseDouble(JOptionPane.showInputDialog(" Fonksiyonun üst sınırını giriniz  x2 : "));
    double r;
    r= polinom2derece(f,x0,x1,x2,1.0e-10,1.0e-5,1);
    System.out.println("Polinom 2 derece optimizasyon optimizasyon değeri  : "+r+"\nFonksiyon değeri :
"+f.func(r));
```

```
    System.exit(0);
    }
}
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FOPO2
x  = 0.7750000000000006f = 2.065322580645161hata x =0.0hata f(x)=y 0.0
Polinom 2 derece optimizasyon optimizasyon değeri  : 0.7750000000000006
Fonksiyon değeri : 2.065322580645161
> Terminated with exit code 0.
```

5.3 Bir bilinmiyenli-lineer olmayan kübik polinom arama yöntemi

Fonksiyonumuza üçüncü dereceden bir polinom olarak yaklaşarak, bu polinomun köklerini bulma işlemini bir önceki bölümdekine benzer bir şekilde gerçekleştirebiliriz. Hermite kübik polinomu
$P(x)=a(x-x_0)^3+b(x-x_0)^2+c(x-x_0)+d$ alınacak olursa, bu polinomun dönüş noktası (minimum maksimum veya ektremum noktası) $[x_0,x_1]$ aralığı için :

$x - x_0 = \frac{-b \pm \sqrt{b^2 - 4ac}}{3a}$ formülünden hesaplanabilir. İterasyon formülümüz, minimum sağlamak için

$x^* = -x_0 + \frac{-b + \sqrt{b^2 - 4ac}}{3a}$ şeklinde yazılabilir. Bu polinomun a b c d katsayılarını bilmiyoruz. Bu katsayıları bulmak için

$p(x_0)=d=f(x_0)$ $p'(x_0)=c=f'(x_0)$
$a=(G-2H)/h$ $h =x_1 – x_0$ $b=3H-G$
$F=[f(x_1) - f(x_0)]/h$
$G==[f'(x_1) – f'(x_0)]/h$
$H=[F – f'(x_0)]/h$

Formülleri ile hesaplanabilir. Bir önceki quadratik formülde olduğu gibi burada da her bulunan nokta yeni bir noktanın bulunması için kullanılabilir. Formüllerde fonksiyonu kendinin yanında türev değerlerinin de bulunması gerekmektedir. Bu değerler program 5.3-1 de sayısal türevler kullanılarak oluşturulmuştur. Sayısal türev kavramı için ilgili bölümümüzü inceleyebilirsiniz.

Program 5.3.1 üçüncü derece polinom optimizasyonu

```
 //
//
// Kübik arama yöntemi ile OPTİMİZASYON
//
//

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

//============== Tanımı gereken fonksiyon =================
```

```
class fb extends f_x
{
   public double func (double x)
   { return x*x*x-1.5*x*x;}
}
//============================================================

public class FSCO4E
{

public static double df(f_x f,double x,double h)
{ //Türev hesabı
  double hh=1.0/h;
  return (-f.func(x+2.0*h)+8.0*f.func(x+h)-8.0*f.func(x-h)+f.func(x-2.0*h))/12.0*hh;
}

public static double cubic_search_opt(f_x f,double x0,double x1)
{
int nmax=500;
double tolerance=1.0e-10;
double f0,f1,df0,df1,fxs,dfxs;
double h=0.0000001;
double xs;
int k=0;
double X,F,G,H,a,b,c;
  f0=f.func(x0);
  f1=f.func(x1);
  df0=df(f,x0,h);
  df1=df(f,x1,h);
while(Math.abs(x0-x1)>tolerance && k<nmax)
{ k++;
  X=1.0/(x1-x0);
  F=X*(f1-f0);
  G=X*(df1-df0);
  H=X*(F-df0);
  a=X*(G-2.0*H);
  b=3.0*H-G;
  c=df0;
  xs=x0+(Math.sqrt(b*b-3.0*a*c)-b)/(3.0*a);
  fxs=f.func(xs);
  dfxs=df(f,xs,h);
  if(dfxs > 0 ) {x1=xs;f1=fxs;df1=dfxs;}
  else          x0=xs;f0=fxs;df0=dfxs;
}
if(k>=nmax)  JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
   " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return (x0+x1)/2.0;1
}
    public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
```

```
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }

    public static void main (String args[])
    {
            fb f=new fb();
    double x[]=verigir("arama bölge sınırlarını giriniz: xo x1");
    double r= cubic_search_opt(f,x[0],x[1]);
    System.out.println("kübik arama yöntemi ile minimizasyon  değeri  : "+r+"\nFonksiyon değeri : "+f.func(r));
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO4E
kübik arama yöntemi ile minimizasyon  değeri  : 1.0000000003200153
Fonksiyon değeri : -0.5000000000000002
> Terminated with exit code 0.
```

5.4 Bir bilinmiyenli-lineer olmayan-geometrik-newton-raphson metodu

Kök bulma işlemleri sırasında Newton-Raphson yöntemini incelemiştik. Optimizasyonun temel tanımına geri dönersek Minimum (veya maksimum) f(x) demek df(x)/dx=0 anlamına geliyordu. Öyleyse Newton-Raphson metodu ile fonksiyonun türevinin kökünü bulursak burada fonksiyonun bir maksimum veya minimum veya extremun (dönme noktası) bulunacağı anlamına gelir.

Minimum f(x)=0

$$g(x) = \frac{df(x)}{dx} = 0$$

Newton raphson formülü

$$x_{n+1} = x_n - \frac{g(x)}{\frac{dg(x)}{dx}} = x_n - \frac{\frac{df(x)}{dx}}{\frac{d^2 f(x)}{dx^2}} = x_n - \frac{f'(x)}{f''(x)}$$

şeklinde yazılabilir. Program 5.4.1 de Newton-Raphson optimizasyon programı verilmiştir. Bu programda ikinci türevler sayısal olarak (fark denklemleri ile) bulunmaktadır.

Program 5.4.1 Newton-Raphson optimizasyon programı

```
 /// OPTIMİZASYON
//  Newton kök bulma yöntemi (ve türev ) kullanarak optimizasyon
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class fc extends f_x
{
   public double func(double x)
   { return x+1/x;}
}

class fb extends f_x
{
   public double func(double x)
   { return 1-1/(x*x);}
}

public class FOPO3
{

public static double derivative(f_x df,double x)
{
// türev alt programı
// df fonksiyonunun türevini hesaplar
double h0=0.256;
int i,m;
int n=7;
//derivative of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
```

```
{
T[i][0]=( df.func(x + h[i]) - df.func( x - h[i]))/(2.0*h[i]);
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/
         (h[i]*h[i]- h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}
public static double newton(f_x df,double x)
{
// df ile verilen türev fonksiyonunun köklerini bulur

int nmax=500;
double tolerance=1.0e-15;
double fx,dfx;
for(int i=0;i<nmax;i++)
{
fx=df.func(x);
dfx=derivative(df,x);
x-=fx/dfx;
if(Math.abs(fx)<tolerance)
{
return x;
}
}
return x;
}
public static double newton_2nd_derivative(f_x df,double x)
{
// df ile verilen türev fonksiyonunun köklerini bulur
int nmax=500;
double dx=1e-3;
double x1m;
double tolerance=1.0e-15;
double fx,fxm,dfx,dfxm,d2fx;
for(int i=0;i<nmax;i++)
{
fx=df.func(x);
fxm=df.func(x-dx);
dfx=derivative(df,x);
dfxm=derivative(df,(x-dx));
d2fx=-6.0/dx/dx*(fx-fxm)+2.0/dx*(2.0*dfx+dfxm);
x-=(fx/dfx+.5*fx*fx/(dfx*dfx*dfx)*d2fx);
if(Math.abs(fx)<tolerance)
{ return x; }
}
```

```
return x;
}

    public static void main (String args[])
    {
            fc  f=new fc();
            fb df=new fb();
    double x0;
    x0=Double.parseDouble(JOptionPane.showInputDialog(" optimum nokta tahmini giriniz  x0 : "));
    double r;
    r= newton(df,x0);
    System.out.println("Newton optimizasyon : "+r+"\nFonksiyon değeri : "+f.func(r));

    JOptionPane.showMessageDialog(null," optimizasyon değeri  : "+r+"\nFonksiyon değeri : "+f.func(r),
    " ",JOptionPane.PLAIN_MESSAGE);
    System.exit(0);
    }
}
```

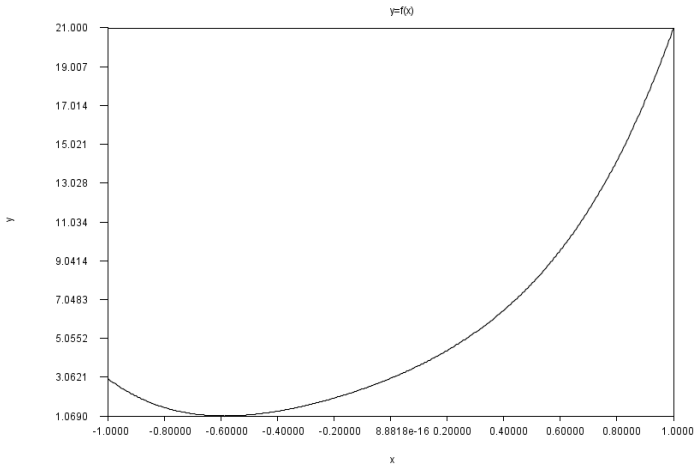
```
    ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FOPO3
Newton optimizasyon : 1.0
Fonksiyon değeri : 2.0
> Terminated with exit code 0.
```

Örnek problem :

Verilen fonksiyonun minimumunu Newton-Raphson metodu ile bulunuz (x_0=0)

$f(x) = 3 + 6x + 5x^2 + 3x^3 + 4x^4 \quad f'(x) = 6 + 10x + 9x^2 + 16x^3 \quad f''(x) = 10 + 18x + 48x^2$

x	f(x)	f'(x)	f''(x)
0	3	6	10
-0.6	1.0704	-0.216	16.48
-0.586893204	1.068969724	-0.003365379	15.9692167
-0.586682462	1.068969369	-8.51268E-07	15.9611386
-0.586682409	1.068969369	-5.4623E-14	15.9611366



5.5 Bir bilinmiyenli-lineer olmayan-geometrik brent metodu

$$x = b + 0.5(\frac{(b-a)^2[f(b)-f(a)]-(b-c)^2[f(b)-f(a)]}{(b-a)\ [f(b)-f(a)]-(b-c)\ [f(b)-f(c)]}$$

parabolik formülünü kullanarak minimuma ulaşabileceğimizi görmüştük. Ancak bu formülün temel bir hatası bulunmaktadır. Minimuma ulaşacağı yerde maksimuma ulaşması da mümkündür. Daha emniyetli bir yol izlemek istersek fonksiyon köke yeterince yaklaşmadan önce altın oran gibi daha emniyetli bir metodla gittikten sonra fonksiyonun minimum değere yaklaştığını gördüğümüzde ikinci derece polinom yaklaşımına geçebiliriz. Brent metodu bu işlevi sağlamak için geliştirilmiştir. Brent metodu bir iterasyonda 6 nokta kullanır. Bu noktalara a,b,u,v,w ve x dersek, tanımları şu şekilde yapılabilir. A ve b minimumun içinde bulunduğu bölgeyi gösterir. x o ana kadar bulunan minimuma en yakın noktadır. w, x den sonra gelen minimuma en yakın noktadır ve v w'dan sonra gelen minimuma en yakın noktadır. ayrıca a b bölgesinin orta noktasını x_m olarak tanımlayabiliriz, ancak bu noktada fonksiyon hesaplanmaz. Önce x v ve w noktalarından bir parabolik eğri geçiririz. Yeni değerin kabul edilebilmesi için önce a – b bölgesinin içine düşmesi şarttır. Eğer bu şart gerçekleşiyor ise eğer bu gerçekleşmiş ise yeni hesaplanan değerin o ana kadar bulunan en iyi minimum değerinin altına düşmüş olması şartı aranır. Bu ikinci şart iterasyonun minimuma yaklaştığı güvencesini sağlar. Eğer minimum nokta bu kriteri sağlıyor ama yine de yeterince minimuma yaklaşamıyorsa araya bir altın oran arama stepi yerleştirilir. Yalız tek bir kötü adım için bu yapılmaz algoritma ikinci adımda kendini toparlayabilir, bu yüzden altın arama stepine geçilmeden önce bir-kaç adım beklenir. Aynı zamanda değerlendirilen iki nokta arasındaki mesafe daha önceden verilen toleransın altına düşmemelidir. Program 5.5.1 de brent minimizasyon algoritması verilmiştir. Şekil 5-4-1 de bu programdaki örnek fonksiyonun grafiği görülmektedir.

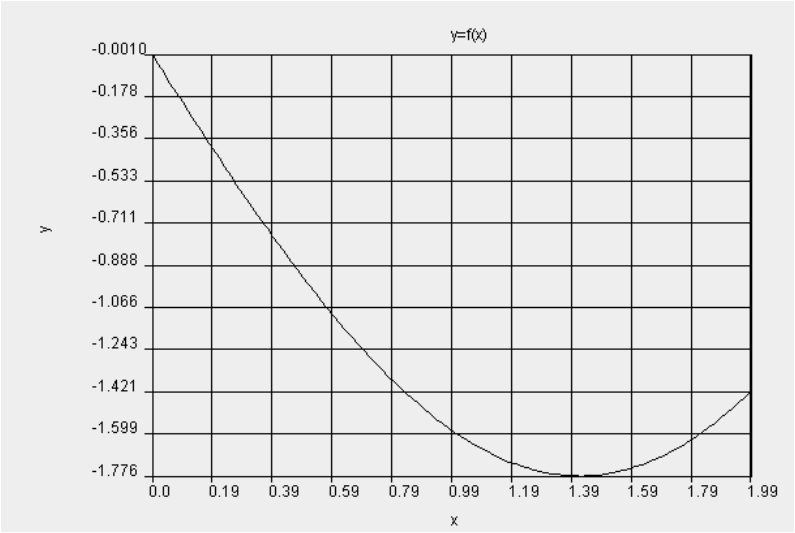
Program 5.5-1 Brent optimizasyon programı

```
 /// OPTIMİZASYON
//  bir boyutlu lineer olmıyan fonksiyonlar
//  brent metodu
//
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

//============= Tanımı gereken fonksiyon ================
class fb extends f_x
{
public double func (double c)
  {
   return -2.0*c/(4.0+0.8*c+c*c+0.2*c*c*c);
  }
    //- değeri maksimumunu elde etmek için verilmiştir
}
//=========================================================

public class OPO9
{    // 3. brent metodu
//_____________________________________________________________
// golden (altın) oran - Fibonacchi metodu ve ikinci derece polinom metodunu birlikte kullanır
public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
```

```
public static double MAX(double a,double b) {return (a > b ? a : b);}
public  static double[] brentf(f_x f,double ax,double bx,double cx,double tol)
{
// ax,bx,cx üç adet tahmin noktası (bxax ile cx arasında yer almalıdır)
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double CGOLD=(3.0-Math.sqrt(5))/2.0; //altın oran
double ZEPS=1.0e-10;
double xmin;
double aa[]=new double[2];
// SHFT(double a,double b,double c,d) (a)=(b);(b)=(c);(c)=(d);
        int iter;
        double a,b,d,etemp,fu,fv,fw,fx,p,q,r,tol1,tol2,u,v,w,x,xm;
        double e=0.0;
        d=0.0;
        a=((ax < cx) ? ax : cx);
        b=((ax > cx) ? ax : cx);
        x=w=v=bx;
        fw=fv=fx=f.func(x);
        for (iter=1;iter<=ITMAX;iter++) {
                xm=0.5*(a+b);
                tol2=2.0*(tol1=tol*Math.abs(x)+ZEPS);
                if (Math.abs(x-xm) <= (tol2-0.5*(b-a))) {
                        xmin=x;
            aa[0]=xmin;
            aa[1]=fx;
            return aa;
                }
                if (Math.abs(e) > tol1) {
                        r=(x-w)*(fx-fv);
                        q=(x-v)*(fx-fw);
                        p=(x-v)*q-(x-w)*r;
                        q=2.0*(q-r);
                        if (q > 0.0) p = -p;
                        q=Math.abs(q);
                        etemp=e;
                        e=d;
                        if (Math.abs(p) >= Math.abs(0.5*q*etemp) || p <= q*(a-x) || p >= q*(b-x))
                                d=CGOLD*(e=(x >= xm ? a-x : b-x));
                        else {
                                d=p/q;
                                u=x+d;
                                if (u-a < tol2 || b-u < tol2)
                                        d=SIGN(tol1,xm-x);
                        }
                } else {
                        d=CGOLD*(e=(x >= xm ? a-x : b-x));
                }
                u=(Math.abs(d) >= tol1 ? x+d : x+SIGN(tol1,d));
                fu=f.func(u);
                if (fu <= fx) {
```

```
                        if (u >= x) a=x; else b=x;
                        {v=w;w=x;x=u;}
                        {fv=fw;fw=fx;fx=fu;}
                } else {
                        if (u < x) a=u; else b=u;
                        if (fu <= fw || w == x) {
                                v=w;
                                w=u;
                                fv=fw;
                                fw=fu;
                        } else if (fu <= fv || v == x || v == w) {
                                v=u;
                                fv=fu;
                        }
                }
        }
        System.out.println("BRENT metodunda maksimum iterasyon sayısı aşıldı");
        xmin=x;
        //minumum değer a[0] ile fonksiyon değeri a[1] ile geridöndürülmektedir
        aa[0]=xmin;
        aa[1]=fx;
        return aa;
}

public  static double brent(f_x f,double ax,double bx,double cx,double tol)
{
 double aa[]=brentf(f,ax,bx,cx,tol);
 return aa[0];
}
    public static void main (String args[])
    {
            String s="üç adetkök tahmin değerini değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=opt.verigir(s);
    fb f=new fb();
    double r1= brent(f,x0[0],x0[1],x0[2],1.0e-8);
    s=" çözüm seti  : "+r1+"\n fonksiyon değeri : "+(-f.func(r1));
    //fonksiyonun gerçek değerini elde etmek için -1 ile çarpılmıştır
    String s2="Brent metodu ile lineer olmıyan bir değişkenli fonksiyon optimizasyonu:\n";
    System.out.println(s2+s);
    }
}
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO9
Brent metodu ile lineer olmıyan bir değişkenli fonksiyon optimizasyonu:
 çözüm seti  : 1.5678894229681264
 fonksiyon değeri : 0.3696348522522553
> Terminated with exit code 0.
```



(bolum5_004.jpg,Resimaltı: Brent optimizasyon programında kullanılan fonksiyonun grafiği)

5.6 Bir bilinmiyenli-lineer olmayan-geometrik-türevli brent metodu

Türevli Brent metodu brent metoduna benzer, temel farkı burada a,b,c gibi 3 nokta alınması ve her yeni noktada fonksiyon değerinin yanında türev değerinin de kontrol edilmesidir. a b intervalinin ortasında yer alan c noktasının türevi bize minimumun sol tarafa mı yoksa sağ tarafamı düştüğü hakında bilgi verir. Böylece bir sonraki arama bölgesinin a-c mi yoksa c-b mi olduğunu bilebiliriz. c noktasın türevi ve ikinci derece interpolasyonla bulunan en iyi noktanın türev değerleri kiriş(sekant) metodu kullanılarak interpole edilir. Eğer yeterli dönüşüm olmuyorsa tekrar altın arama metodunun uygulanmasına gidilir.

Program 5.6.1 Türevli Brent optimizasyon programı

```
 /// OPTIMİZASYON
//  bir boyutlu lineer olmayan fonksiyonlar
//  türevli brent metodu
//
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

//============== Tanımı gereken fonksiyon =================
class fb extends f_x
{
  public double func (double x)
  { return -(2.0*Math.sin(x)-x*x/10.0);}
}
//=========================================================

public class OPO10
{
        public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
    public static double MAX(double a,double b) {return (a > b ? a : b);}
    public static double df(f_x f,double x)
```

```
{
// türev alt programı
// df fonksiyonunun türevini hesaplar
double h0=0.0256;
int i,m;
int n=7;
//turev of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
T[i][0]=( f.func(x + h[i]) - f.func( x - h[i]))/(2.0*h[i]);
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/
         (h[i]*h[i]- h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}

     public  static double dbrent(f_x f,double ax,double bx,double cx,double tol)
{
// ax,bx,cx üç adet tahmin noktası (bxax ile cx arasında yer almalıdır)
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double CGOLD=(3.0-Math.sqrt(5))/2.0; //altın oran
double ZEPS=1.0e-10;
// MOV3(a,b,c, d,e,f) (a)=(d);(b)=(e);(c)=(f);
// {a=d;b=e;c=f}
             int iter;
```

```
boolean ok1,ok2;
double a,b,d,d1,d2,du,dv,dw,dx,e=0.0;
d=0.0;
double fu,fv,fw,fx,olde,tol1,tol2,u,u1,u2,v,w,x,xm;
double xmin=bx;
a=(ax < cx ? ax : cx);
b=(ax > cx ? ax : cx);
x=w=v=bx;
fw=fv=fx=f.func(x);
dw=dv=dx=df(f,x);
for (iter=1;iter<=ITMAX;iter++) {
        xm=0.5*(a+b);
        tol1=tol*Math.abs(x)+ZEPS;
        tol2=2.0*tol1;
        if (Math.abs(x-xm) <= (tol2-0.5*(b-a))) {
                xmin=x;
                return xmin;
        }
        if (Math.abs(e) > tol1) {
                d1=2.0*(b-a);
                d2=d1;
                if (dw != dx)  d1=(w-x)*dx/(dx-dw);
                if (dv != dx)  d2=(v-x)*dx/(dx-dv);
                u1=x+d1;
                u2=x+d2;
                ok1 = (a-u1)*(u1-b) > 0.0 && dx*d1 <= 0.0;
                ok2 = (a-u2)*(u2-b) > 0.0 && dx*d2 <= 0.0;
                olde=e;
                e=d;
                if (ok1 || ok2) {
                        if (ok1 && ok2)
                                d=(Math.abs(d1) < Math.abs(d2) ? d1 : d2);
                        else if (ok1)
                                d=d1;
                        else
                                d=d2;
                        if (Math.abs(d) <= Math.abs(0.5*olde)) {
                                u=x+d;
                                if (u-a < tol2 || b-u < tol2)
                                        d=SIGN(tol1,xm-x);
                        } else {
                                d=0.5*(e=(dx >= 0.0 ? a-x : b-x));
                        }
                } else {
                        d=0.5*(e=(dx >= 0.0 ? a-x : b-x));
                }
        } else {
                d=0.5*(e=(dx >= 0.0 ? a-x : b-x));
        }
        if (Math.abs(d) >= tol1) {
                u=x+d;
                fu=f.func(u);
```

```
                } else {
                        u=x+SIGN(tol1,d);
                        fu=f.func(u);
                        if (fu > fx) {
                                xmin=x;
                                return xmin;
                        }
                }
                du=df(f,u);
                if (fu <= fx) {
                        if (u >= x) a=x; else b=x;
                        //MOV3(v,fv,dv, w,fw,dw)
                        {v=w;fv=fw;dv=dw;}
                        //MOV3(w,fw,dw, x,fx,dx)
                        {w=x;fw=fx;dw=dx;}
                        //MOV3(x,fx,dx, u,fu,du)
                        {x=u;fx=fu;dx=du;}
                } else {
                        if (u < x) a=u; else b=u;
                        if (fu <= fw || w == x) {
                                //MOV3(v,fv,dv, w,fw,dw)
                                {v=w;fv=fw;dv=dw;}
                                //MOV3(w,fw,dw, u,fu,du)
                                {w=u;fw=fu;dw=du;}
                        } else if (fu < fv || v == x || v == w) {
                                //MOV3(v,fv,dv, u,fu,du)
                                {v=u;fv=fu;dv=du;}
                        }
                }
        }
        System.out.println("DBRENT maximum iterasyon sayısını aştı sonuç geçerli olmıyabilir");
        return xmin;
}
    public static void main (String args[])
    {
         String s="üç adetkök tahmin değerini değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=opt.verigir(s);
    fb f=new fb();
    double r1= dbrent(f,x0[0],x0[1],x0[2],1.0e-8);
    s=" çözüm seti  : "+r1;
    String s2="türevli Brent metodu ile lineer olmayan bir değişkenli fonksiyon optimizasyonu:\n";
    System.out.println(s2+s);
    }
}
```

```
   ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO10
türevli Brent metodu ile lineer olmayan bir değişkenli fonksiyon optimizasyonu:
 çözüm seti  : 1.4275517787636531
> Terminated with exit code 0.
```

5.7 Optimizasyon sınır tahmin değerlerinin iyileştirilmesi

Optimizasyon değerlerini herhangi bir optimizasyon yöntemi ile aramaya başlamadan önce sınır değerlerinin iyileştirilmesi yapılabilir. Böylece daha yakın bir tahmin değerinden interpolasyon işlemine geçerken minimum olan bir alan seçilmiş olduğuna emin olabiliriz. Mnbrak metodu 3 noktayla verilen arama bölgesini içinde bir minimum değer bulunana kadar genişletir. Bu işle için altın oran arama tekniğini kullanır.

Program 5.7-1 mnbrak optimizasyon braket tayin programı

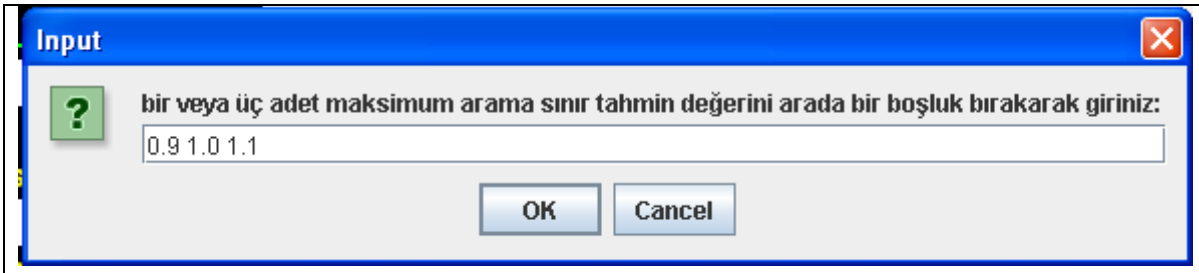
```
 /// OPTIMİZASYON
//  bir boyutlu lineer olmayan fonksiyonlar
//  brent metodu
//
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

//============= Tanımı gereken fonksiyon ================
class fb extends f_x
{
   public double func (double x)
   { return -(2.0*Math.sin(x)-x*x/10.0);}
   //- değeri maksimumunu elde etmek için verilmiştir
}
//=========================================================
public class OPO11
{
// 5. minumumu brakete alma (sınırlarını bulma) arama metodu
//    altın oran araması kullanır
public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}


public  static double[] mnbrak(f_x f,double ax,double dx)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçilen uzaklık
return mnbrak(f,ax,(ax-dx),(ax+dx));
}

public  static double[] mnbrak(f_x f,double ax)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçiln uzaklık
return mnbrak(f,ax,(ax-0.1),(ax+0.1));
}

public  static double[] mnbrak(f_x f,double ax,double bx,double cx)
{
```

```
// ax,bx,cx üç adet tahmin noktası verildiğinde içinde minimum bulunan ax,bx,cx üç adetnoktayı bize arama
// sonucu bulur
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double xmin;
double GOLD=(Math.sqrt(5.0)+1.0)/2.0;;
double GLIMIT=100.0;
double TINY=1.0e-20;
double fa,fb,fc;
fa=0;
fb=0;
fc=0;
        double ulim,u,r,q,fu,dum;
        double aa[]=new double[3];
        fa=f.func(ax);
        fb=f.func(bx);
        if (fb > fa) {
                //SHFT(dum,*ax,*bx,dum)
                {dum=ax;ax=bx;bx=dum;}
                //SHFT(dum,*fb,*fa,dum)
                {dum=fb;fb=fa;fa=dum;}
        }
        cx=(bx)+GOLD*(bx-ax);
        fc=f.func(cx);
                aa[0]=ax;
                aa[1]=bx;
                aa[2]=cx;
        while (fb > fc) {
                r=(bx-ax)*(fb-fc);
                q=(bx-cx)*(fb-fa);
                u=(bx)-((bx-cx)*q-(bx-ax)*r)/
                        (2.0*SIGN(MAX(Math.abs(q-r),TINY),q-r));
                ulim=(bx)+GLIMIT*(cx-bx);
                if ((bx-u)*(u-cx) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                ax=bx;
                                bx=u;
                                fa=fb;
                                fb=fu;
                                return aa;
                        } else if (fu > fb) {
                                cx=u;
                                fc=fu;
                                return aa;
                        }
                        u=(cx)+GOLD*(cx-bx);
                        fu=f.func(u);
                } else if ((cx-u)*(u-ulim) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
```

```
                                        //SHFT(bx,cx,u,cx+GOLD*(cx-bx))
                                        {bx=cx;cx=u;u=cx+GOLD*(cx-bx);}
                                        //SHFT(fb,fc,fu,f.func(u))
                                        {fb=fc;fc=fu;fu=f.func(u);}
                                }
                        } else if ((u-ulim)*(ulim-cx) >= 0.0) {
                                u=ulim;
                                fu=f.func(u);
                        } else {
                                u=cx+GOLD*(cx-bx);
                                fu=f.func(u);
                        }
                        //SHFT(ax,bx,cx,u)
                        {ax=bx;bx=cx;cx=u;}
                        //SHFT(fa,fb,fc,fu)
                        {fa=fb;fb=fc;fc=fu;}
                        aa[0]=ax;
                        aa[1]=bx;
                        aa[2]=cx;
                }
                        return aa;
}
//
// veri giriş çıkış metodu
//_____________________________________________________________________
    public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }
    public static void main (String args[])
    {
            String s="bir veya üç adet maksimum arama sınır tahmin değerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=verigir(s);
    fb f=new fb();
    double r1[];
    if(x0.length==3)      {r1= opt.mnbrak(f,x0[0],x0[1],x0[2]);}
    else if(x0.length==1) {r1= opt.mnbrak(f,x0[0]);}
    else{System.out.println("Yanlış maksimum raama sınır tahmin değeri verdiniz, ilk değer üzerinden hesap
yapılacaktır");
             r1= mnbrak(f,x0[0]);}
    s=" yeni kök tahmin değerleri  : \n"+Matrix.toString(r1);
```

```
    //fonksiyonun gerçek değerini elde etmek için -1 ile çarpılmıştır
    String s2="bir boyutta içinde optimum olan bölgeyi sağlama\n";
    System.out.println(s2+s);
    }
}
```

Çıktı 5.7-1 mnbrak optimizasyon braket tayin programı



(bolum5_005.jpg,Resimaltı:)

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO11
bir boyutta içinde optimum olan bölgeyi sağlama
 yeni kök tahmin değerleri  :
      1.16180339887498940000000000        1.46140749717968420000000000
1.94617711140544510000000000

> Terminated with exit code 0.
```

5.8 Çok bilinmiyenli-lineer olmayan-geometrik-newton-raphson metodu

$f(x_1,x_2,x_3,\ldots,x_n)$ fonksiyon seti verildiğinde, birinci, ikinci türevler ve x değerleri iterasyon farkları

$$\nabla f = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{Bmatrix} \text{ ve } \delta^{m+1} = \begin{Bmatrix} (x_1^{m+1} - x_1^m) \\ (x_2^{m+1} - x_2^m) \\ (x_3^{m+1} - x_3^m) \\ \cdots \\ (x_n^{m+1} - x_n^m) \end{Bmatrix} \text{ ve } [H] = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_{21} \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_3} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \frac{\partial^2 f}{\partial x_2 \partial x_n} & \frac{\partial^2 f}{\partial x_3 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

şeklinde verilebilir. (m+1) inci iterasyon için Newton-Raphson denklemi

$$\left[H^m\right]\{\delta^{m+1}\} = \{\nabla f^m\}$$

şeklinde yazılabilir. Görüldüğü gibi lineer bir denklem sistemi oluşmuştur ve her iterasyonda bu denklem sisteminin çözülmesi gerekir. Sistem çözüldüğünde tüm x değerlerinin yeni iteratif değerleri bulunur . ikinci türev H hessian matrisi ismini alır. Birinci ve ikinci türevler sayısal metodlarla da hesaplanabilir, ancak bu durumda hata olasılıkları doğal olarak artar. Bu denklem

$\{\delta^{m+1}\} = \left[H^m\right]^{-1}\{\nabla f^m\}$ olarak ta yazılabilir. Burada $\left[H^m\right]^{-1}$ Hesian matrisinin ters (inverse) matrisidir.

Program 5.8-1 Çok bilinmiyenli lineer olmayan geometrik Newton-Raphson optimizasyon programı (Birinci ve ikinci türevler ayrı fonksiyonlar olarak verilmiş)

```
 // SCO5 Newton Raphson metoduyla
// birden fazla değişkenli denklem sisteminin köklerini bulmak

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============= Tanımı gereken fonksiyon ================
abstract class fij_xi extends Mathd
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0][0]=x[0]+sin(x[1])  f[0][1]=x[0]*x[1]
  //         f[1][0]=x[0]*x[0]-x[1]  f[1][1]=x[0]+x[1]*x[0]
  // func(x) returns the value of f[0][0] and f[0][1]
  //                          f[0][1] and f[1][1]
  // as a two dimensional vector
  //
  abstract double[][] func(double x[]);
}

abstract class fi_xi extends Mathd
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x) returns the value of f[0][0]
  //                          f[0][1]
  // as a two dimensional vector
  //
  abstract double[] func(double x[]);
}

abstract class f_xj extends Mathd
{
  // single function  multi independent variable
  // a single value is returned indiced to equation_ref
  // example f=x[0]+sin(x[1])
  //
  // func(x,1) returns the value of f[1]
  // func(x,0) returns the value of f[0]

  abstract double func(double x[]);
}

class fxa extends f_xj
{
public double func(double x[])
```

```
{
//çözümü istenen fonksiyon
double ff;
ff=3.0*x[0]*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

class fxb extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyonun türev fonksiyonları
double ff[]=new double[2];
ff[0]=6.0*x[0]-4.0*x[1]-1.0;
ff[1]=-4.0*x[0]+4.0*x[1]-1.0;
return ff;
}
}

class fxc extends fij_xi
{
public double[][] func(double x[])
{
//çözümü istenen fonksiyonun türev fonksiyonları
double ff[][]=new double[2][2];
ff[0][0]=6.0;ff[0][1]=-4.0;
ff[1][0]=-4.0;ff[1][1]=4.0;
return ff;
}
}

public class FSCO5G
{
// Newton Raphson Metodunu kullanarak denklem sisteminin köklerini bulmak
// Bu metod denklem sistemi çözümü için Matrix.java programını
// kullanmaktadır.


public static double[] newtond( fi_xi f,fij_xi f1,double x[])
{
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi dfj/dxi turev alt programıyla hesaplanır.
// Newton-Raphson metodu ile lineer olmıyan denklem sistemi çözümü
//ti :önem katsayısı
//x bağımsız değişken vektörü
//y bağımlı değişken vektörü
//dy bağımlı değişkenin türev fonksiyonları vektörü
double ti=1.0;
int i,ii,jj;
int nmax=500;
```

```
double tolerance=1.0e-15;
int n=x.length;
double b[];
b=new double[n];
double dy[][];
dy=new double[n][n];
i=0;
for(i=0;i<n;i++)
{
b[i]=1.0;
}
while( i++ < nmax && Matrix.abs(b) > tolerance )
{
  for(ii=0;ii<n;ii++)
  {
    for(jj=0;jj<n;jj++)
    {
       dy[ii][jj]=dfx(f,x,ii,jj);
    }
  }
    b=multiply(divide(dy(f,x),H(f1,x)),-ti);
    x=add(x,b);
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}

public static double[][]H(fij_xi f,double x[])
{
  //Hessian matris
  int n=x.length;
  double dy[][]=f.func(x);
  System.out.println(Matrix.toString(dy));
  return dy;
}

public static double[] dy(fi_xi f,double x[])
{
//birinci türev
  int n=x.length;

  double deltay[]=f.func(x);
  System.out.println(Matrix.toStringT(deltay));
  return deltay;
}



public static double dfx(fi_xi f,double x[],int denklem_ref,int x_ref)
{ // Türev hesabı
double h=0.001;
```

```
double hh=1.0/h;
int i,m;
int n=x.length;
double f1[];
f1=new double[n];
double f2[];
f2=new double[n];
double f3[];
f3=new double[n];
double f4[];
f4=new double[n];

double x1[];
x1=new double[n];
double x2[];
x2=new double[n];
double x3[];
x3=new double[n];
double x4[];
x4=new double[n];

for(i=0;i<n;i++)
{
x1[i]=x[i];
x2[i]=x[i];
x3[i]=x[i];
x4[i]=x[i];
}
x1[x_ref]+=2.0*h;
x2[x_ref]+=h;
x3[x_ref]-=h;
x4[x_ref]-=2.0*h;

f1=f.func(x1);
f2=f.func(x2);
f3=f.func(x3);
f4=f.func(x4);
double fx=(f1[denklem_ref]+8.0*f2[denklem_ref]-8.0*f3[denklem_ref]+f4[denklem_ref])/12.0*hh;
//System.out.println("f("+x_ref+","+denklem_ref+") = "+fx);
return fx;
}
public static double[] multiply(double[] left,double right)
{
//multiplying a vector with a constant
int i;
int n=left.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {
  b[i]=right*left[i];
  }
```

```
return b;
}

public static double abs(double[] left)
{
// absolute value of a vector
int i;
int n=left.length;
double b=0;
for(i=0;i<n;i++)
  {
  b=b+Math.abs(left[i]);
  }
return b;
}

public static double[] divide(double[] left,double[][] right)
{
return pivotlugauss(right,left);
}
  //en kucuk kareler metodu
  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
              if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
                a[k][jj]=dummy;
              }
              dummy=b[p];
              b[p]=b[k];
              b[k]=dummy;
            }
            //gauss elemeyi çözme
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
```

```
      a[i][k]=0;
      for(j=k+1;j<n;j++)
      {   a[i][j]-=carpan*a[k][j]; }
         b[i]   =b[i]   -carpan*b[k];
           }
  }
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
}

public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
   Double ax=new Double(token.nextToken());
```

```
    a[j++]=ax.doubleValue();
    }
    return a;
}

     public static void main (String args[])
     {
            fxa f1 =new fxa();
            fxb f2 =new fxb();
            fxc f3 =new fxc();
     double [] x0=verigir();
     String s2="Newton-Raphson kök bulma : ";
     double [] r1= newtond(f2,f3,x0);
     String s=" kök değerleri  : \n"+Matrix.toStringT(r1);
     System.out.println(s2+s);
     System.exit(0);
     }
}
```

Çıktı 5.8-1Çok bilinmiyenli lineer olmayan geometrik Newton-Raphson optimizasyon programı

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO5G
      -3.000000000000000000000000000
       3.000000000000000000000000000

       6.000000000000000       -4.000000000000000
      -4.000000000000000        4.000000000000000

       0.000000000000000000000000000
       0.000000000000000000000000000

       6.000000000000000       -4.000000000000000
      -4.000000000000000        4.000000000000000

Newton-Raphson kök bulma :  kök değerleri  :
       1.000000000000000000000000000
       1.250000000000000000000000000

> Terminated with exit code 0.
```

Program 5.8-2 Çok bilinmiyenli lineer olmayan geometrik Newton-Raphson optimizasyon programı , Hessian matrix H sayısal türev kullanılarak hesaplanmış.

```
 // FSCO5H Newton Raphson metoduyla
// birden fazla değişkenli denklem sisteminin köklerini bulmak
// ve optimizasyon
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============= Tanımı gereken fonksiyon ================
abstract class fij_xi extends Mathd
```

```
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0][0]=x[0]+sin(x[1])  f[0][1]=x[0]*x[1]
  //         f[1][0]=x[0]*x[0]-x[1]  f[1][1]=x[0]+x[1]*x[0]
  // func(x) returns the value of f[0][0] and f[0][1]
  //                          f[0][1] and f[1][1]
  // as a two dimensional vector
  //
  abstract double[][] func(double x[]);
}

abstract class fi_xi extends Mathd
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x) returns the value of f[0][0]
  //                          f[0][1]
  // as a two dimensional vector
  //
  abstract double[] func(double x[]);
}

abstract class f_xj extends Mathd
{
  // single function  multi independent variable
  // a single value is returned indiced to equation_ref
  // example f=x[0]+sin(x[1])
  //
  // func(x,1) returns the value of f[1]
  // func(x,0) returns the value of f[0]

  abstract double func(double x[]);
}

class fxa extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=3.0*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

class fxb extends fi_xi
{
public double[] func(double x[])
{
```

```
//çözümü istenen fonksiyonun türev fonksiyonları
double ff[]=new double[2];
ff[0]=6.0*x[0]-4.0*x[1]-1.0;
ff[1]=-4.0*x[0]+4.0*x[1]-1.0;
return ff;
}
}


public class FSCO5H
{
// Newton Raphson Metodunu kullanarak denklem sisteminin köklerini bulmak
// Bu metod denklem sistemi çözümü için Matrix.java programını
// kullanmaktadır.


public static double[] newtond( fi_xi f,double x[])
{
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi dfj/dxi turev alt programıyla hesaplanır.
// Newton-Raphson metodu ile lineer olmıyan denklem sistemi çözümü
//ti :önem katsayısı
//x bağımsız değişken vektörü
//y bağımlı değişken vektörü
//dy bağımlı değişkenin türev fonksiyonları vektörü
double ti=1.0;
int i,ii,jj;
int nmax=500;
double tolerance=1.0e-10;
int n=x.length;
double b[];
b=new double[n];
double dy[][];
dy=new double[n][n];
i=0;
for(i=0;i<n;i++)
{
b[i]=1.0;
}
while( i++ < nmax && Matrix.abs(b) > tolerance )
{
   b=multiply(divide(dy(f,x),H(f,x)),-ti);
   x=add(x,b);
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}

public static double dfx(fi_xi f,double x[],int denklem_ref,int x_ref)
```

```
{ // Türev hesabı
double h=0.001;
double hh=1.0/h;
int i,m;
int n=x.length;
double f1[];
f1=new double[n];
double f2[];
f2=new double[n];
double f3[];
f3=new double[n];
double f4[];
f4=new double[n];

double x1[];
x1=new double[n];
double x2[];
x2=new double[n];
double x3[];
x3=new double[n];
double x4[];
x4=new double[n];

for(i=0;i<n;i++)
{
x1[i]=x[i];
x2[i]=x[i];
x3[i]=x[i];
x4[i]=x[i];
}
x1[x_ref]+=2.0*h;
x2[x_ref]+=h;
x3[x_ref]-=h;
x4[x_ref]-=2.0*h;
f1=f.func(x1);
f2=f.func(x2);
f3=f.func(x3);
f4=f.func(x4);
double fx=(-f1[denklem_ref]+8.0*f2[denklem_ref]-8.0*f3[denklem_ref]+f4[denklem_ref])/12.0*hh;
return fx;
}

public static double[][] H(fi_xi f,double x[])
{
  int n=x.length;
  double dy[][]=new double[n][n];;
  for(int ii=0;ii<n;ii++)
  {
    for(int jj=0;jj<n;jj++)
    {   //Hessian matrisi
        dy[ii][jj]=dfx(f,x,ii,jj);
    }
```

```
  }
  System.out.println(Matrix.toString(dy));
  return dy;
}

public static double[] dy(fi_xi f,double x[])
{
//birinci türev
  int n=x.length;
  double deltay[]=f.func(x);
  System.out.println(Matrix.toStringT(deltay));
  return deltay;
}

public static double[] multiply(double[] left,double right)
{
//multiplying a vector with a constant
int i;
int n=left.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {
  b[i]=right*left[i];
  }
return b;
}


public static double[] divide(double[] left,double[][] right)
{
return pivotlugauss(right,left);
}
  //en kucuk kareler metodu
  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
              if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
```

```
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
                a[k][jj]=dummy;
              }
              dummy=b[p];
              b[p]=b[k];
              b[k]=dummy;
            }
            //gauss elemeyi çözme
   for(i=k+1;i<n;i++)
   {  carpan=a[i][k]/a[k][k];
      a[i][k]=0;
      for(j=k+1;j<n;j++)
      {   a[i][j]-=carpan*a[k][j]; }
         b[i]   =b[i]   -carpan*b[k];
           }
  }
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
```

```
}

public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
   Double ax=new Double(token.nextToken());
   a[j++]=ax.doubleValue();
   }
   return a;
}

    public static void main (String args[])
    {
            fxa f1 =new fxa();
            fxb f2 =new fxb();
    double [] x0=verigir();
    String s2="Newton-Raphson kök bulma : ";
    double [] r1= newtond(f2,x0);
    String s=" kök değerleri  : \n"+Matrix.toStringT(r1);
    System.out.println(s2+s);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO5H
      -7.000000000000000000000000
       7.000000000000000000000000

       5.999999999999635       -3.999999999999855
      -3.999999999999264        3.999999999999855

       0.000000000000025311308496145
      -0.000000000000025357493882444

       5.999999999999783       -3.999999999999411
      -4.000000000000041        3.999999999999411

Newton-Raphson kök bulma :  kök değerleri  :
       1.000000000000000200000000
       1.250000000000000200000000

> Terminated with exit code 0.
```

Program 5.8-3 Çok bilinmiyenli lineer olmayan geometrik Newton-Raphson optimizasyon programı (Birinci ve ikinci türevler sayısal olarak hesaplanmış)

```
// FSCO5I Newton Raphson metoduyla
// birden fazla değişkenli denklem sisteminin köklerini bulmak
// ve optimizasyon Hessian matris H sayısal olarak hesaplanmış, birinci türev de sayısal olarak hesaplanmış

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============= Tanımı gereken fonksiyon ================
abstract class fij_xi extends Mathd
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0][0]=x[0]+sin(x[1])  f[0][1]=x[0]*x[1]
  //         f[1][0]=x[0]*x[0]-x[1]  f[1][1]=x[0]+x[1]*x[0]
  // func(x) returns the value of f[0][0] and f[0][1]
  //                          f[0][1] and f[1][1]
  // as a two dimensional vector
  //
  abstract double[][] func(double x[]);
}

abstract class fi_xi extends Mathd
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x) returns the value of f[0][0]
  //                          f[0][1]
  // as a two dimensional vector
  //
  abstract double[] func(double x[]);
}

abstract class f_xj extends Mathd
{
  // single function  multi independent variable
  // a single value is returned indiced to equation_ref
  // example f=x[0]+sin(x[1])
  //
  // func(x,1) returns the value of f[1]
  // func(x,0) returns the value of f[0]

  abstract double func(double x[]);
}

class fxa extends f_xj
```

```
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=3.0*x[0]*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}
class fxb extends fi_xi
{
f_xj fxx;

fxb(f_xj fxi)
{fxx=fxi;}

public double[] func(double x[])
{
// Türev hesabı
double h=0.0013421432165;
double hh=1.0/h;
int i,m;
int n=x.length;
double f1;
double f2;
double f3;
double f4;
double x1[];
x1=new double[n];
double x2[];
x2=new double[n];
double x3[];
x3=new double[n];
double x4[];
x4=new double[n];
double fx[]=new double[n];

for(int j=0;j<n;j++)
{
for(i=0;i<n;i++)
{
x1[i]=x[i];
x2[i]=x[i];
x3[i]=x[i];
x4[i]=x[i];
}
x1[j]+=2.0*h;
x2[j]+=h;
x3[j]-=h;
x4[j]-=2.0*h;

f1=fxx.func(x1);
```

```
f2=fxx.func(x2);
f3=fxx.func(x3);
f4=fxx.func(x4);
fx[j]=(-f1+8.0*f2-8.0*f3+f4)/12.0*hh;
}
return fx;
}

}

public class FSCO5I
{
// Newton Raphson Metodunu kullanarak denklem sisteminin köklerini bulmak
// Bu metod denklem sistemi çözümü için Matrix.java programını
// kullanmaktadır.


public static double[] newtond( fi_xi f,double x[])
{
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi dfj/dxi turev alt programıyla hesaplanır.
// Newton-Raphson metodu ile lineer olmıyan denklem sistemi çözümü
//ti :önem katsayısı
//x bağımsız değişken vektörü
//y bağımlı değişken vektörü
//dy bağımlı değişkenin türev fonksiyonları vektörü
double ti=1.0;
int i,ii,jj;
int nmax=500;
double tolerance=1.0e-15;
int n=x.length;
double b[];
b=new double[n];
double dy[][];
dy=new double[n][n];
i=0;
for(i=0;i<n;i++)
{
b[i]=1.0;
}
while( i++ < nmax && Matrix.abs(b) > tolerance )
{
  for(ii=0;ii<n;ii++)
  {
    for(jj=0;jj<n;jj++)
    {
       dy[ii][jj]=dfx(f,x,ii,jj);
    }
  }
    b=multiply(divide(dy(f,x),H(f,x)),-ti);
    x=add(x,b);
```

```
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}

public static double dfx(fi_xi f,double x[],int denklem_ref,int x_ref)
{ // Türev hesabı
double h=0.00014354523465;
double hh=1.0/h;
int i,m;
int n=x.length;
double f1[];
f1=new double[n];
double f2[];
f2=new double[n];
double f3[];
f3=new double[n];
double f4[];
f4=new double[n];

double x1[];
x1=new double[n];
double x2[];
x2=new double[n];
double x3[];
x3=new double[n];
double x4[];
x4=new double[n];

for(i=0;i<n;i++)
{
x1[i]=x[i];
x2[i]=x[i];
x3[i]=x[i];
x4[i]=x[i];
}
x1[x_ref]+=2.0*h;
x2[x_ref]+=h;
x3[x_ref]-=h;
x4[x_ref]-=2.0*h;

f1=f.func(x1);
f2=f.func(x2);
f3=f.func(x3);
f4=f.func(x4);
double fx=(-f1[denklem_ref]+8.0*f2[denklem_ref]-8.0*f3[denklem_ref]+f4[denklem_ref])/12.0*hh;
//System.out.println("f("+x_ref+","+denklem_ref+") = "+fx);
return fx;
}
```

```
public static double[][] H(fi_xi f,double x[])
{
  int n=x.length;
  double dy[][]=new double[n][n];;
  for(int ii=0;ii<n;ii++)
  {
    for(int jj=0;jj<n;jj++)
    {   //Hessian matrisi
        dy[ii][jj]=dfx(f,x,ii,jj);
    }
  }
  System.out.println(Matrix.toString(dy));
  return dy;
}

public static double[] dy(fi_xi f,double x[])
{
//birinci türev
  int n=x.length;
  double deltay[]=f.func(x);
  System.out.println(Matrix.toStringT(deltay));
  return deltay;
}

public static double[] multiply(double[] left,double right)
{
//multiplying a vector with a constant
int i;
int n=left.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {
  b[i]=right*left[i];
  }
return b;
}


public static double[] divide(double[] left,double[][] right)
{
return pivotlugauss(right,left);
}
  //en kucuk kareler metodu
  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
```

```
  //gauss eleme
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
              if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
                a[k][jj]=dummy;
              }
              dummy=b[p];
              b[p]=b[k];
              b[k]=dummy;
            }
            //gauss elemeyi çözme
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
          b[i]   =b[i]   -carpan*b[k];
            }
  }
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
```

```
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
}

public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
   Double ax=new Double(token.nextToken());
   a[j++]=ax.doubleValue();
   }
   return a;
}

    public static void main (String args[])
    {
            fxa f1 =new fxa();
            fxb f2 =new fxb(f1);
    double [] x0=verigir();
    String s2="Newton-Raphson kök bulma : ";
    double [] r1= newtond(f2,x0);
    String s=" kök değerleri  : \n"+Matrix.toStringT(r1);
    JOptionPane.showMessageDialog(null,s,s2,JOptionPane.PLAIN_MESSAGE);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO5H
      -7.000000000000000000000000
       7.000000000000000000000000

       5.999999999999635       -3.999999999999855
      -3.999999999999264        3.999999999999855

       0.000000000000252531308496145
      -0.000000000000252535749388244
```

```
      5.999999999999783       -3.999999999999411
     -4.000000000000041        3.999999999999411

Newton-Raphson kök bulma :  kök değerleri  :
      1.000000000000000200000000
      1.250000000000000200000000

> Terminated with exit code 0.
```

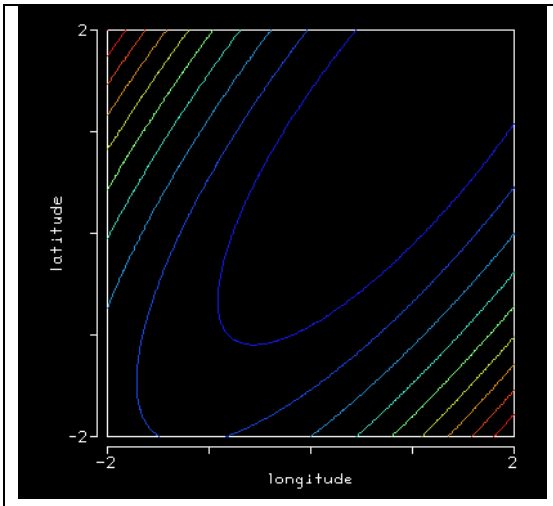
ÖRNEK PROBLEM 5.8-1

$f(x_0,x_1)=3x_0^2-4x_0x_1+2x_1^2-x_0-x_1$ fonksiyonunun $\{x\}=[-2,2]$ noktasından başlayarak minimum değerlerini bulunuz.

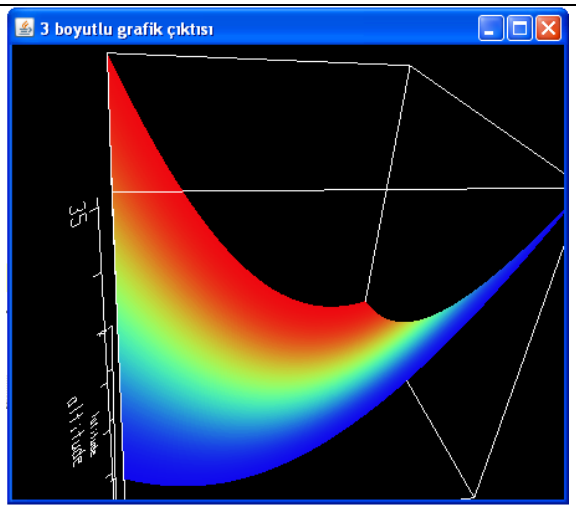
$$\nabla f=\begin{Bmatrix}6x_0-4x_1-1\\4x_1-4x_0-1\end{Bmatrix}=\begin{Bmatrix}-21\\15\end{Bmatrix}\quad [H]=\begin{bmatrix}6&-4\\-4&4\end{bmatrix}\quad [H]^{-1}=\begin{bmatrix}0.5&0.5\\0.5&0.75\end{bmatrix}$$

$$\{\delta^{m+1}\}=\begin{Bmatrix}x_0^1\\x_1^1\end{Bmatrix}-\begin{Bmatrix}-2\\2\end{Bmatrix}=[H^m]^{-1}\{\nabla f^m\}=\begin{Bmatrix}3\\-0.75\end{Bmatrix}$$

$$\begin{Bmatrix}x_0^1\\x_1^1\end{Bmatrix}=\begin{Bmatrix}1\\1.25\end{Bmatrix}$$



(bolum5_006.jpg,Resimaltı:)



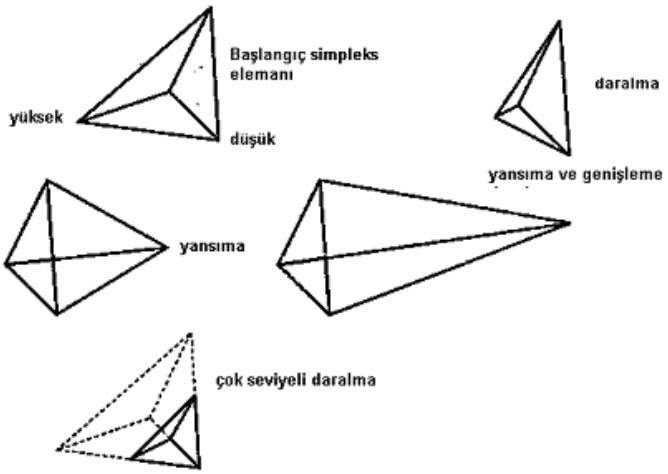
(bolum5_007.jpg,Resimaltı:)

5.9 Çok bilinmiyenli-lineer olmayan-geometrik-Nelder ve Mead simpleks metodu

Nelder ve Mead tarafından geliştirilen çok boyutlu simpleks metodu olarak adlandırabileceğimiz metod bir anlamda tek boyutta yaptığımız altın arama metodunu çok boyuta yansıtma prosesidir.

Simpleks n boyutlu bir geometrik şekildir. N boyutta (N+1) nokta içerir. İki boyuttaki simpleks bir üçgenden oluşur. Üç boyutta bir üçgenler prizması halini alır (4 yüzeyli). Bir boyutlu minimizasyon prosesinde minimumun sınırlarını saptamak olasıdır. Ancak çok boyuta geçtiğimizde bu olasılık ortadan kalkar. N boyutta en iyi yapabileceğimiz arama

prosesine başlamak için N+1 nokta vermektir. Bundan sonra program yüzey topolojisini inceleyerek minimuma doğru hareket etmeye başlar. Bu hareketlerin esası şu şekilde oluşur : program verilen tüm noktalarda fonksiyonu analiz ederek en yüksek ve en düşük değerleri bulur, diğer noktaları aynı bırakarak en yüksek değeri en yüksek ve en düşük elemanların üzerinde olduğu yüzeye göre yansıma noktasına taşır. Minimizasyon doğrultusunda yeterli değişim yok ise yansıtma 1 faktöründen daha büyük yada daha küçük bir faktör kullanılarak ta gerçekleştirilebilir. Minimum değere yaklaştığımızda simpleks daralma ile küçülür, bu daralma proseside bir seferde birden fazla nokta için uygulanabilir. Şekil 5.9-1 de bu prosesleri geometrik olarak 3 boyutlu uzay simpleksi için (4 nokta) görüyoruz.



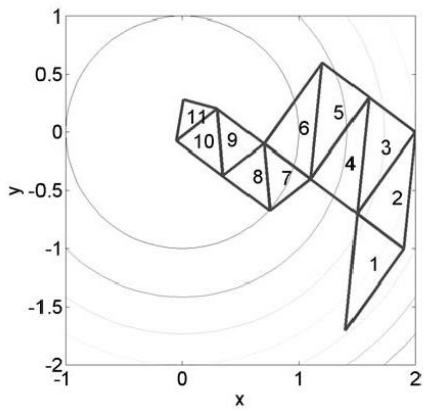
(bolum5_008.jpg,Resimaltı: Nelder-Mead simpleks amip hareketleri)

N+1 nokta vererek nelder mead iteratif araştırmasına başlamak yerine bir nokta vererek ve bu noktayı $x_{i-1} = P_i + \lambda dx_j$ denklemi ile N+1 noktaya genişleterek N+1 noktamızı elde edebiliriz. bu denklemde eğer i=j λ=1 eğer i ≠ j λ=0 değeri alır. Yani örneğin

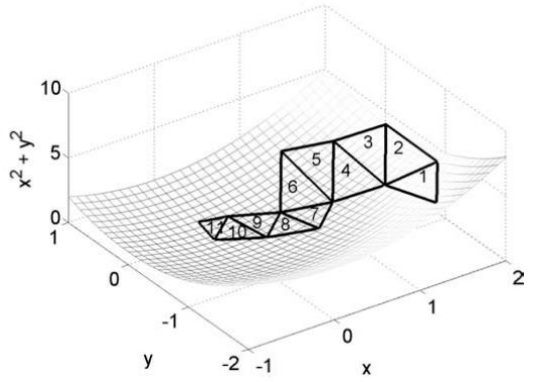
$P = \begin{Bmatrix} a \\ b \\ c \end{Bmatrix}$ ve $dx = \begin{Bmatrix} da \\ db \\ dc \end{Bmatrix}$ ise

$x_0 = \begin{Bmatrix} a \\ b \\ c \end{Bmatrix}$ $x_1 = \begin{Bmatrix} a+da \\ b \\ c \end{Bmatrix}$ $x_2 = \begin{Bmatrix} a \\ b+db \\ c \end{Bmatrix}$ $x_3 = \begin{Bmatrix} a \\ b \\ c+dc \end{Bmatrix}$ olacaktır. Böylece simplekse girdi olarak N+1 vektör yerine 2 vektör vererek nelder mead iterasyonuna başlayabiliriz. İterasyon stepleri şu şekilde gerçekleşir :

1. verilen N+1 Nokta için maksimum (en büyük), maksimumdan sonraki en büyük ve minimum fonksiyon değerlerini veren noktalar bulunur.
2. (yansıtma) maksimum değeri veren nokta tüm diğer noktaların ortalama değeri üzerinden yansıtılarak yeni bir nokta elde edilir. $x_{ortalama} = \frac{1}{n} \sum_{i \neq i-maksimum} x_i$ $x_{yeni1} = x_{ortalama} - x_{maksimum}$
3. (genişlemeli yansıtma) eğer elde edilen yeni noktanın değeri minimum noktsındaki değerden küçük ise yansıtma doğru yönde bir ilerleme sağlamış demektir, daha büyük ikinci bir yansıtma (genişleme) uygulanabilir: eğer $f(x_{yeni1}) < f(x_{minimum})$ $x_{yeni2} = \lambda x_{yeni1} - (1-\lambda)x_{ortalama}$ burada λ birden küçük bir sayıdır.
   a. eğer $f(x_{yeni2}) < f(x_{minimum})$ $x_{maksimum} = x_{yeni2}$ $x_{maksimum}$ yerine x_{yeni2} değeri alınır.
   b. Değilse) $x_{maksimum} = x_{yeni1}$ $x_{maksimum}$ yerine x_{yeni1} değeri alınır.
4. Şart 3 daha iyi bir değer vermemiş ise yansıtma prosesi bize iyi bir değer vermeyecek demektir, ancak $f(x_{yeni1})$ değeri $f(x_{maksimum})$ ve $f(x_{maksimumdan\ bir\ küçük})$ değerleri arasında ise yine
   a. $x_{maksimum} = x_{yeni1}$ alınabilir.
   b. Aynı zamanda yeni bir nokta araması 1 den küçük bir genleşme sayısı (büzülme) kullanılarak gerçekleştirilebilir. $x_{yeni2} = \beta x_{maksimum} - (1-\beta)x_{ortalama}$. Eğer bu proses başarılı ise $f(x_{yeni2}) < f(x_{maksimum})$ $x_{maksimum} = x_{yeni2}$ $x_{maksimum}$ yerine x_{yeni2} değeri alınır.
5. Eğer yukarıdaki şartların hiçbiri başarı sağlamamış ise tüm simpleks elemanları minimumun olduğu noktaya doğru büzülür. $x_i = \frac{1}{2}(x_i + x_{minimum})$
6. böylece bir çevrim prosesi bitirilmiş olur ve aynı işleme yeni bir çevrim için başlanılır.



(bolum5_009.jpg,Resimaltı:)



(bolum5_010.jpg,Resimaltı:)

Üstteki şekilde Nelder ve Mead amipinin maksimuma ne şekilde yaklaştığı görülmektedir. Program 5.9.1 de bu işlemin programlanmış formu verilmiştir.

Program 5.9.1 Nelder ve Mead simpleks optimizasyon arama programı

```
 // Newton metodu ile optimizasyon
//
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;


class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=(x[0]-2.0)*(x[0]-2.0)*(x[0]-2.0)*(x[0]-2.0)+(x[1]-1.0)*(x[1]-1.0)+
3.0*(x[2]-1.0)*(x[2]-1.0)*(x[2]-1.0)*(x[2]-1.0)*(x[2]-1.0)*(x[2]-1.0);
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

public class OPO7B
{
//2. Nelder ve Mead metodu (simpleks arama metodu)
//_____________________________________________________________________
public static double[] nelder(f_xj fnelder,double a[],double da[],int maxiteration,double tolerance,int printlist)
{
            int i,j;
double x[][]=new double[a.length+1][a.length];
double p[][]=new double[a.length+1][a.length+1];
for(i=0;i<x.length;i++)
  {for(j=0;j<x[0].length;j++)
    {if(i==j){x[i][j]=a[j]+da[j];p[i][j]=x[i][j];}
     else    {x[i][j]=a[j];p[i][j]=x[i][j]; }
```

```
   }
  p[i][j] = fnelder.func(p[i]);
 }
 //System.out.println("x=\n"+Matrix.toString(x));
 // Nelder mead çok boyutlu simplex minimizasyon metodu
 //  Nelder & Mead 1965 Computer J, v.7, 308-313.

 // Giriş değişkenleri tanımlaması
 // fnelder : abstract çok boyutlu fonksiyon f(x)
 // x : for n boyutlu n+1 simlex noktasını içerir bağımsız değişken seti
 // maxiteration : maximum iterasyon sayısı
 // tolerance :
      int NDIMS = x.length-1;
      int NPTS = x.length;
      int FUNC = NDIMS;
      int ncalls = 0;
        ////// başlangıç simplexini oluştur ///////////////////
        //double p[][]=new double[NPTS][NPTS]; // [row][col] = [whichvx][coord,FUNC]
        double z[]=new double[NDIMS];
        double best = 1E99;
        ///////////////// iilk fonksiyon değerlerini hesapla /////////////////
       /*
         for (int i=0; i<NPTS; i++)
        {
          for (int j=0; j<NDIMS; j++)
             {p[i][j] = x[i][j];}
          p[i][NDIMS] = fnelder.func(p[i]);
          System.out.println("i="+i+"p=\n"+Matrix.toString(p));
        }
        */
        int iter=0;

        for (iter=1; iter<maxiteration; iter++)
        {
           //System.out.println("iter="+iter+"p=\n"+Matrix.toString(p));
                   ////////////  lo, nhi, hi noktalarını tanımla ///////////////
           int  ilo=0, ihi=0, inhi = -1; // -1 means missing
           double flo = p[0][FUNC];
           double fhi = flo;
           double pavg,sterr;
           for (i=1; i<NPTS; i++)
           {
             if (p[i][FUNC] < flo)
               {flo=p[i][FUNC]; ilo=i;}
             if (p[i][FUNC] > fhi)
               {fhi=p[i][FUNC]; ihi=i;}
           }
           double fnhi = flo;
           inhi = ilo;
           for (i=0; i<NPTS; i++)
            if ((i != ihi) && (p[i][FUNC] > fnhi))
              {fnhi=p[i][FUNC]; inhi=i;}
```

```
            //System.out.println("büyük y i="+fhi+" "+ihi+" ikinci büyük y i="+fnhi+" "+inhi+" küçük y
i="+flo+" "+ilo);
            ////////// çıkış kriteri //////////////
            if ((iter % 4*NDIMS) == 0)
            {
               //yi nin standart hata kriteri set değerinden (tolerance)
               // küçük olmalı
               // ortalama değeri hesapla (en büyük değer de dahil olmak üzere)
               pavg=0;
               for(i=0;i<NPTS;i++)
                 pavg+=p[i][FUNC];
               pavg/=NPTS;
               double tot=0;
               if(printlist!=0)
               {  System.out.print(iter);
                  for (j=0; j<=NDIMS; j++)
                    { System.out.print(p[ilo][j]+" ");}
                  System.out.println("");
               }
               for(i=0;i<NPTS;i++)
               { tot=(p[i][FUNC]-pavg)*(p[i][FUNC]-pavg);}
               sterr=Math.sqrt(tot/NPTS);
               //if(sterr < tolerance)
                 { for (j=0; j<NDIMS; j++)
                     { z[j]=p[ilo][j];}
                   //break;
                 }
               best = p[ilo][FUNC];
            }

            ///// ave[] vektorünü en büyük değeri hariç tutarak hesapla //////

            double ave[] = new double[NDIMS];
            for (j=0; j<NDIMS; j++)
              ave[j] = 0;
            for (i=0; i<NPTS; i++)
              if (i != ihi)
                for (j=0; j<NDIMS; j++)
                  ave[j] += p[i][j];
            for (j=0; j<NDIMS; j++)
               ave[j] /= (NPTS-1);


            ///////// yansıt ////////////////

            double r[] = new double[NDIMS];
            for (j=0; j<NDIMS; j++)
              r[j] = 2*ave[j] - p[ihi][j];
            double fr = fnelder.func(r);

            if ((flo <= fr) && (fr < fnhi))  // in zone: accept
            {
```

```
    for (j=0; j<NDIMS; j++)
      p[ihi][j] = r[j];
    p[ihi][FUNC] = fr;
    continue;
  }

  if (fr < flo)  //// genişlet, else kabul et
  {
    double e[] = new double[NDIMS];
    for (j=0; j<NDIMS; j++)
      e[j] = 3*ave[j] - 2*p[ihi][j];
    double fe = fnelder.func(e);
    if (fe < fr)
    {
      for (j=0; j<NDIMS; j++)
        p[ihi][j] = e[j];
      p[ihi][FUNC] = fe;
      continue;
    }
    else
    {
      for (j=0; j<NDIMS; j++)
        p[ihi][j] = r[j];
      p[ihi][FUNC] = fr;
      continue;
    }
  }

  ////////////// daralt:

  if (fr < fhi)
  {
    double c[] = new double[NDIMS];
    for (j=0; j<NDIMS; j++)
      c[j] = 1.5*ave[j] - 0.5*p[ihi][j];
    double fc = fnelder.func(c);
    if (fc <= fr)
    {
      for (j=0; j<NDIMS; j++)
        p[ihi][j] = c[j];
      p[ihi][FUNC] = fc;
      continue;
    }
    else   /////// daralt
    {
      for (i=0; i<NPTS; i++)
        if (i != ilo)
        {
          for (j=0; j<NDIMS; j++)
            p[i][j] = 0.5*p[ilo][j] + 0.5*p[i][j];
          p[i][FUNC] = fnelder.func(p[i]);
        }
```

```
                continue;
              }
            }

            if (fr >= fhi)   ///
            {
              double cc[] = new double[NDIMS];
              for (j=0; j<NDIMS; j++)
                cc[j] = 0.5*ave[j] + 0.5*p[ihi][j];
              double fcc = fnelder.func(cc);
              if (fcc < fhi)
              {
                for (j=0; j<NDIMS; j++)
                  p[ihi][j] = cc[j];
                p[ihi][FUNC] = fcc;
                continue;
              }
              else    /////////
              {
                for (i=0; i<NPTS; i++)
                  if (i != ilo)
                  {
                    for (j=0; j<NDIMS; j++)
                      p[i][j] = 0.5*p[ilo][j] + 0.5*p[i][j];
                    p[i][FUNC] = fnelder.func(p[i]);
                  }
              }
            }
          }
      return z;
}
     public static double[] nelder(f_xj fnelder,double a[],double da[],double tolerance)
     {return nelder(fnelder,a,da,500,tolerance,0);}

     public static double[] nelder(f_xj fnelder,double a[],double da[])
     {return nelder(fnelder,a,da,500,1.0e-10,0);}
   // veri giriş çıkış metodu
   //_____________________________________________________________
     public static double[] verigir(String s)
     {
     String s1=JOptionPane.showInputDialog(s);
     StringTokenizer token=new StringTokenizer(s1);
     int n=token.countTokens()-1;
     int m=n+1;
     double a[]=new double[m];
     int j=0;
     while(token.hasMoreTokens())
     {
     Double ax=new Double(token.nextToken());
     a[j++]=ax.doubleValue();
     }
     return a;
```

```
    }
        public static void main (String args[])
    {
        String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=verigir(s);
    double [] dx0=new double[x0.length];
    for(int i=0;i<x0.length;i++) dx0[i]=0.2*x0[i];
    f1 f=new f1();
    double [] r1= nelder(f,x0,dx0);
    s=" çözüm seti  : \n"+Matrix.toStringT(r1);
    String s2="Nelder-Mead metodu ile lineer olmıyan çok değişkenli fonksiyon optimizasyonu:";
     System.out.println(s2+s);
    System.exit(0);
    }
}
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO7B
Nelder-Mead metodu ile lineer olmıyan çok değişkenli fonksiyon optimizasyonu: çözüm seti  :
       2.000000000000000000000000
       1.000000000000000000000000
       1.000000000000000000000000

> Terminated with exit code 0.
```

ÖRNEK PROBLEM

Bir örnek olarak işlemin detaylarına bakacak olursak :

$f(x_0,x_1)=3x_0^2-4x_0x_1+2x_1^2-x_0-x_1$ fonksiyonunun {x}=[-2,2] noktasından başlayarak minimum değerlerini bulalım

ilk 50 iterasyonun sonuçları nelder_mead_ornek_sonuc.txt dosyasında yer almaktadır.

Çıktı 5.9-2 Nelder ve Mead simpleks optimizasyon arama örnek çıktısı

```
iter=1p=
      -2.400000000000000       2.000000000000000      44.879999999999995
      -2.000000000000000       2.400000000000000      42.320000000000000
      -2.000000000000000       2.000000000000000      36.000000000000000

büyük y i=44.879999999999995 0 ikinci büyük y i=42.32 1 küçük y i=36.0 2
iter=2p=
      -1.200000000000000       2.600000000000000      28.920000000000005
      -2.000000000000000       2.400000000000000      42.320000000000000
      -2.000000000000000       2.000000000000000      36.000000000000000

büyük y i=42.32 1 ikinci büyük y i=36.0 2 küçük y i=28.920000000000005 0
iter=3p=
      -1.200000000000000       2.600000000000000      28.920000000000005
      -0.800000000000001       2.100000000000000      16.160000000000014
      -2.000000000000000       2.000000000000000      36.000000000000000

büyük y i=36.0 2 ikinci büyük y i=28.920000000000005 0 küçük y i=16.160000000000014 1
iter=4p=
      -1.200000000000000       2.600000000000000      28.920000000000005
      -0.800000000000001       2.100000000000000      16.160000000000014
```

```
    0.999999999999999    3.050000000000002    5.355000000000019

büyük y i=28.920000000000005 0 ikinci büyük y i=16.160000000000014 1 küçük y i=5.355000000000019 2
iter=5p=
    1.399999999999998    2.550000000000002    0.655000000000011
   -0.800000000000001    2.100000000000000   16.160000000000014
    0.999999999999999    3.050000000000002    5.355000000000019

büyük y i=16.160000000000014 1 ikinci büyük y i=5.355000000000019 2 küçük y i=0.6550000000000114 0
iter=6p=
    1.399999999999998    2.550000000000002    0.655000000000011
    3.199999999999998    3.500000000000003    3.719999999999994
    0.999999999999999    3.050000000000002    5.355000000000019

büyük y i=5.355000000000019 2 ikinci büyük y i=3.7199999999999944 1 küçük y i=0.6550000000000114 0
iter=7p=
    1.399999999999998    2.550000000000002    0.655000000000011
    3.199999999999998    3.500000000000003    3.719999999999994
    1.649999999999998    3.037500000000002    1.885312500000012

büyük y i=3.7199999999999944 1 ikinci büyük y i=1.8853125000000115 2 küçük y i=0.6550000000000114 0
iter=8p=
    1.399999999999998    2.550000000000002    0.655000000000011
    2.362499999999998    3.146875000000002    1.302519531250004
    1.649999999999998    3.037500000000002    1.885312500000012

büyük y i=1.8853125000000115 2 ikinci büyük y i=1.3025195312500042 1 küçük y i=0.6550000000000114 0
iter=9p=
    1.399999999999998    2.550000000000002    0.655000000000011
    2.362499999999998    3.146875000000002    1.302519531250004
    2.112499999999998    2.659375000000002    0.288925781250000

büyük y i=1.3025195312500042 1 ikinci büyük y i=0.6550000000000114 0 küçük y i=0.2889257812500001 2
iter=10p=
    1.399999999999998    2.550000000000002    0.655000000000011
    1.149999999999998    2.062500000000001   -0.224687499999993
    2.112499999999998    2.659375000000002    0.288925781250000

büyük y i=0.6550000000000114 0 ikinci büyük y i=0.2889257812500001 2 küçük y i=-0.22468749999999327
1
iter=11p=
    1.862499999999998    2.171875000000001   -0.374042968750002
    1.149999999999998    2.062500000000001   -0.224687499999993
    2.112499999999998    2.659375000000002    0.288925781250000

büyük y i=0.2889257812500001 2 ikinci büyük y i=-0.22468749999999327 1 küçük y i=-0.3740429687500022
0
iter=12p=
    1.862499999999998    2.171875000000001   -0.374042968750002
    1.149999999999998    2.062500000000001   -0.224687499999993
    0.899999999999998    1.575000000000000   -0.753749999999998
```

```
büyük y i=-0.22468749999999327 1 ikinci büyük y i=-0.3740429687500022 0 küçük y i=-
0.7537499999999975 2
iter=13p=
      1.862499999999998      2.171875000000001      -0.374042968750002
      1.612499999999998      1.684375000000000      -0.686386718750004
      0.899999999999998      1.575000000000000      -0.753749999999998

büyük y i=-0.3740429687500022 0 ikinci büyük y i=-0.6863867187500037 1 küçük y i=-0.7537499999999975
2
iter=14p=
      0.649999999999998      1.087499999999999      -0.932187499999998
      1.612499999999998      1.684375000000000      -0.686386718750004
      0.899999999999998      1.575000000000000      -0.753749999999998

büyük y i=-0.6863867187500037 1 ikinci büyük y i=-0.7537499999999975 2 küçük y i=-0.9321874999999984
0
iter=15p=
      0.649999999999998      1.087499999999999      -0.932187499999998
      1.193749999999998      1.507812500000000      -1.079252929687500
      0.899999999999998      1.575000000000000      -0.753749999999998

büyük y i=-0.7537499999999975 2 ikinci büyük y i=-0.9321874999999984 0 küçük y i=-1.0792529296874995
1
iter=16p=
      0.649999999999998      1.087499999999999      -0.932187499999998
      1.193749999999998      1.507812500000000      -1.079252929687500
      0.943749999999998      1.020312499999998      -1.061674804687500

büyük y i=-0.9321874999999984 0 ikinci büyük y i=-1.0616748046875 2 küçük y i=-1.0792529296874995 1
iter=17p=
      0.859374999999998      1.175781249999999      -1.096405029296875
      1.193749999999998      1.507812500000000      -1.079252929687500
      0.943749999999998      1.020312499999998      -1.061674804687500
……..
iter=49p=
      1.000000969021338      1.250005597923464      -1.124999999956208
      0.999993292813279      1.249984113824493      -1.124999999786506
      1.000019269216091      1.250017992060081      -1.124999999625434

büyük y i=-1.1249999996254345 2 ikinci büyük y i=-1.124999999786506 1 küçük y i=-1.1249999999562075
0
```

Amip hareketlerini grafik olarak görmek istersek :

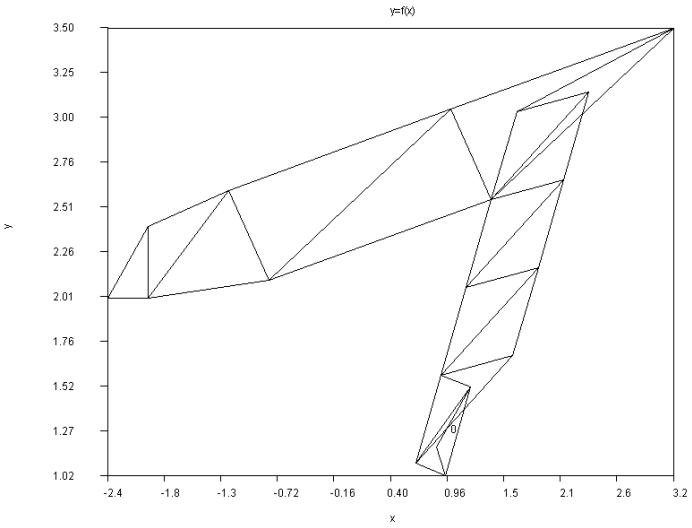
Program 5.9-2 Problem 5.9-1 deki Nelder ve Mead simpleks optimizasyon sonuçlarını görmek amacıyla olan grafik programı

```
import java.io.*;

class PlotT01
{
   public static void main (String args[]) throws java.io.IOException
   {
```

```
        double x1[]={-2.4,-2,-2,-2.4};
        double y1[]={2,2.4,2,2};
        double x2[]={-1.2,-2,-2,-1.2};
        double y2[]={2.6,2.4,2,2.6};
        double x3[]={-1.2,-0.8,-2,-1.2};
        double y3[]={2.6,2.1,2,2.6};
        double x4[]={-1.2,-0.8,1,-1.2};
        double y4[]={2.6,2.1,3.05,2.6};
        double x5[]={1.4,-0.8,1,1.4};
        double y5[]={2.55,2.1,3.05,2.55};
        double x6[]={1.4,3.2,1,1.4};
        double y6[]={2.55,3.5,3.05,2.55};
        double x7[]={1.4,3.2,1.65,1.4};
        double y7[]={2.55,3.5,3.0375,2.55};
        double x8[]={1.4,2.3625,1.65,1.4};
        double y8[]={2.55,3.146875,3.0375,2.55};
        double x9[]={1.4,2.3625,2.1125,1.4};
        double y9[]={2.55,3.146875,2.659375,2.55};
        double x10[]={1.4,1.15,2.1125,1.4};
        double y10[]={2.55,2.0625,2.659375,2.55};
        double x11[]={1.8625,1.15,2.1125,1.8625};
        double y11[]={2.171875,2.0625,2.659375,2.171875};
        double x12[]={1.8625,1.15,0.9,1.8625};
        double y12[]={2.171875,2.0625,1.575,2.171875};
        double x13[]={1.8625,1.6125,0.9,1.8625};
        double y13[]={2.171875,1.684375,1.575,2.171875};
        double x14[]={0.65,1.6125,0.9,0.65};
        double y14[]={1.0874,1.684375,1.575,1.0874};
        double x15[]={0.65,1.19375,0.9,0.65};
        double y15[]={1.0874,1.5078125,1.575,1.0874};
        double x16[]={0.65,1.19375,0.94375,0.65};
        double y16[]={1.0874,1.5078125,1.0203125,1.0874};
        double x17[]={0.859375,1.19375,0.94375,0.859375};
        double y17[]={1.17578125,1.5078125,1.0203125,1.17578125};
        Plot pp=new Plot(x1,y1);
   pp.addData(x2,y2);
   pp.addData(x3,y3);
   pp.addData(x4,y4);
   pp.addData(x5,y5);
   pp.addData(x6,y6);
   pp.addData(x7,y7);
   pp.addData(x8,y8);
   pp.addData(x9,y9);
   pp.addData(x10,y10);
   pp.addData(x11,y11);
   pp.addData(x12,y12);
   pp.addData(x13,y13);
   pp.addData(x14,y14);
   pp.addData(x15,y15);
   pp.addData(x16,y16);
   pp.addData(x17,y17);
    pp.plot();
```

```
    }
}
```

Çıktı 5.9-3 Nelder ve Mead simpleks optimizasyon arama örneği simpleks hareketi grafik programı çıktısı



(bolum5_011.jpg,Resimaltı:Nelder-Mead metodu amip hareketi)
Grafikte çözüm noktası 0 ile gösterilmiştir.

500 iterasyon sonunda sonuç :

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO7B
Nelder-Mead metodu ile lineer olmıyan çok değişkenli fonksiyon optimizasyonu: çözüm seti  :
      1.000000006670072000000000
      1.250000003839312900000000

> Terminated with exit code 0.
```

ÖRNEK PROBLEM
İkinci bir örnek olarak fabrika yeri problemini inceleyelim: Yeni bir ürün için fabrika yeri seçilmek isteniyor. Her şehrin koordinatları ve şehirdeki müşteri sayıları (şehir nüfusları olabilir) verildiğinde minimum mal taşıma maliyetini minimum toplam yol) veren fabrika yerini seçmeyi göz önüne alalım. Eğer N_i i şehrinin nüfusu, x_i ve y_i i şehrinin koordinatları ise ve fabrika koordinatları x,y olarak verilirse, Minimizasyon fonksiyonunu

$f = \sum_{i=1}^{n} N_i \left[(x - x_i)^2 + (y - y_i)^2 \right]$ olacaktır. Bir örnek test fonksiyonu olarak

Nüfus	X koordinatı	Y koordinatı
10	0	0
10	0	1
10	1	0
10	1	1

Bu veri seti için çözümün 0.5,0.5 noktası olacağı görülmektedir. Şimdi problemi oluşturalım ve çözümü inceleyelim.

Program 5.9-3 Fabrika yeri seçimi optimizasyonu problemi, Nelder ve Mead simpleks optimizasyonu

```
// Nelder-Mead metodu ile optimizasyon
//
import java.util.*;
```

```
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;


class ff1 extends f_xj
{
double a[][];

public ff1(String s)
{a=Text.readDoubleT(s);}

public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0;
int n=a[0].length;
for(int i=0;i<n;i++)
  ff+=a[0][i]*((x[0]-a[1][i])*(x[0]-a[1][i])+(x[1]-a[2][i])*(x[1]-a[2][i]));
return ff;
}
}

public class OPO7B1
{
        public static void main (String args[])
   {
        String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
   double [] x0=OPO7B.verigir(s);
   double [] dx0=new double[x0.length];
   for(int i=0;i<x0.length;i++) dx0[i]=0.2*x0[i];
   ff1 f=new ff1("sehir.txt");
   double [] r1= OPO7B.nelder(f,x0);
   s=" çözüm seti  : \n"+Matrix.toStringT(r1);
   String s2="Nelder-Mead metodu ile lineer olmıyan çok değişkenli fonksiyon optimizasyonu:";
   JOptionPane.showMessageDialog(null,s,s2,JOptionPane.PLAIN_MESSAGE);
   System.exit(0);
   }
}
```

Çıktı 5.9-4 Fabrika yeri seçimi optimizasyonu problemi, Nelder ve Mead simpleks optimizasyonu, step step nelder-Med simpleksinin aldığı değerlerin dökümü

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" OPO7B1
iter=1p=
      0.120000000000000      0.100000000000000      32.176000000000000
      0.100000000000000      0.120000000000000      32.176000000000000
      0.100000000000000      0.100000000000000      32.800000000000004

büyük y i=32.800000000000004 2 ikinci büyük y i=32.176 0 küçük y i=32.176 0
iter=2p=
```

0.120000000000000 0.100000000000000 32.176000000000000
0.100000000000000 0.120000000000000 32.176000000000000
0.130000000000000 0.130000000000000 30.951999999999990

büyük y i=32.176 0 ikinci büyük y i=32.176 1 küçük y i=30.95199999999999 2
iter=3p=
0.110000000000000 0.150000000000000 30.983999999999995
0.100000000000000 0.120000000000000 32.176000000000000
0.130000000000000 0.130000000000000 30.951999999999990

büyük y i=32.176 1 ikinci büyük y i=30.983999999999995 0 küçük y i=30.95199999999999 2
iter=4p=
0.110000000000000 0.150000000000000 30.983999999999995
0.160000000000000 0.180000000000000 28.719999999999995
0.130000000000000 0.130000000000000 30.951999999999990

büyük y i=30.983999999999995 0 ikinci büyük y i=30.95199999999999 2 küçük y i=28.719999999999995 1
iter=5p=
0.215000000000000 0.165000000000000 27.737999999999985
0.160000000000000 0.180000000000000 28.719999999999995
0.130000000000000 0.130000000000000 30.951999999999990

büyük y i=30.95199999999999 2 ikinci büyük y i=28.719999999999995 1 küçük y i=27.737999999999985 0
iter=6p=
0.215000000000000 0.165000000000000 27.737999999999985
0.160000000000000 0.180000000000000 28.719999999999995
0.302500000000000 0.257500000000001 23.912499999999984

büyük y i=28.719999999999995 1 ikinci büyük y i=27.737999999999985 0 küçük y i=23.912499999999984 2
iter=7p=
0.215000000000000 0.165000000000000 27.737999999999985
0.456250000000000 0.273750000000001 22.124124999999978
0.302500000000000 0.257500000000001 23.912499999999984

büyük y i=27.737999999999985 0 ikinci büyük y i=23.912499999999984 2 küçük y i=22.124124999999978 1
iter=8p=
0.543750000000001 0.366250000000001 20.792124999999988
0.456250000000000 0.273750000000001 22.124124999999978
0.302500000000000 0.257500000000001 23.912499999999984

büyük y i=23.912499999999984 2 ikinci büyük y i=22.124124999999978 1 küçük y i=20.792124999999988 0
iter=9p=
0.543750000000001 0.366250000000001 20.792124999999988
0.456250000000000 0.273750000000001 22.124124999999978
0.697500000000001 0.382500000000002 22.112499999999997

büyük y i=22.124124999999978 1 ikinci büyük y i=22.112499999999997 2 küçük y i=20.792124999999988 0
iter=10p=
0.543750000000001 0.366250000000001 20.792124999999988
0.538437500000001 0.324062500000001 21.297257812499986
0.697500000000001 0.382500000000002 22.112499999999997

büyük y i=22.112499999999997 2 ikinci büyük y i=21.297257812499986 1 küçük y i=20.792124999999988 0
iter=11p=
0.543750000000001 0.366250000000001 20.792124999999988
0.538437500000001 0.324062500000001 21.297257812499986
0.462890625000000 0.326484375000001 21.259391113281236

büyük y i=21.297257812499986 1 ikinci büyük y i=21.259391113281236 2 küçük y i=20.792124999999988 0
iter=12p=
0.543750000000001 0.366250000000001 20.792124999999988
0.433085937500000 0.390976562500001 20.654544067382805
0.462890625000000 0.326484375000001 21.259391113281236

büyük y i=21.259391113281236 2 ikinci büyük y i=20.792124999999988 0 küçük y i=20.654544067382805 1
iter=13p=
0.543750000000001 0.366250000000001 20.792124999999988
0.433085937500000 0.390976562500001 20.654544067382805
0.539472656250001 0.482871093750001 20.074059600830080

büyük y i=20.792124999999988 0 ikinci büyük y i=20.654544067382805 1 küçük y i=20.07405960083008 2
iter=14p=
0.428808593750000 0.507597656250001 20.205037628173827
0.433085937500000 0.390976562500001 20.654544067382805
0.539472656250001 0.482871093750001 20.074059600830080

büyük y i=20.654544067382805 1 ikinci büyük y i=20.205037628173827 0 küçük y i=20.07405960083008 2
iter=15p=
0.428808593750000 0.507597656250001 20.205037628173827
0.509667968750000 0.547363281250001 20.093470001220710
0.539472656250001 0.482871093750001 20.074059600830080

büyük y i=20.205037628173827 0 ikinci büyük y i=20.09347000122071 1 küçük y i=20.07405960083008 2
iter=16p=
0.476689453125000 0.511357421875001 20.026894905090334
0.509667968750000 0.547363281250001 20.093470001220710
0.539472656250001 0.482871093750001 20.074059600830080

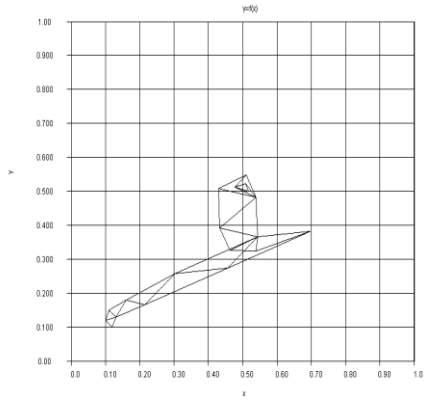
büyük y i=20.09347000122071 1 ikinci büyük y i=20.07405960083008 2 küçük y i=20.026894905090334 0
iter=17p=
0.476689453125000 0.511357421875001 20.026894905090334
0.508874511718750 0.522238769531251 20.022932793140413
0.539472656250001 0.482871093750001 20.074059600830080

büyük y i=20.07405960083008 2 ikinci büyük y i=20.026894905090334 0 küçük y i=20.022932793140413 1
iter=18p=
0.476689453125000 0.511357421875001 20.026894905090334
0.508874511718750 0.522238769531251 20.022932793140413
0.516127319335938 0.499834594726564 20.010404711514713

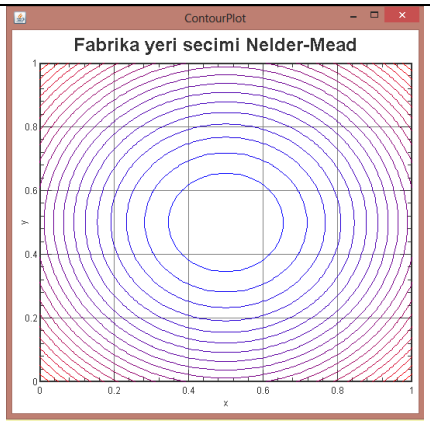
büyük y i=20.07405960083008 2 ikinci büyük y i=20.026894905090334 0 küçük y i=20.022932793140413 1
iter=52p=
0.500002657994614 0.499999541116936 20.000000000291020
0.499999313009312 0.500000148257349 20.000000000019753

```
      0.499999853685161       0.499998268210034       20.000000000120820

büyük y i=20.000000000029102 0 ikinci büyük y i=20.000000000012082 2 küçük y i=20.000000000019753 1
> Terminated with exit code 0.
```

Çıktı 5.9-5 Fabrika yeri seçimi optimizasyonu problemi, Nelder ve Mead simpleks optimizasyonu, step step nelder-Mead simpleksinin aldığı değerlerin ilk 18 inin grafik ortamda gösterilişi başlangıç değeri [0.1, 0.1] , çözüm değeri [0.5,0.5]



(bolum5_012.jpg,Resimaltı:Nelder-Mead metodu amip hareketi)



(bolum5_013.jpg,Resimaltı:Nelder-Mead metodu örnek fonksiyon contour grafiği)

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO7B1
Nelder-Mead metodu ile lineer olmıyan çok değişkenli fonksiyon optimizasyonu: çözüm seti  :
      0.500000001326383900000000
      0.499999997857373650000000

> Terminated with exit code 0.
```

5.10 Çok bilinmiyenli-lineer olmayan-geometrik-en dik yamaç metodu

N boyutlu sınırlamasız optimizasyon problemi Minimum : $f(x_0,x_1,x_2,\ldots,x_n)$ problemini çözme yöntemlerine devam ediyoruz. Denklemimizin türev fonksiyonu

$$\nabla f(x_1,x_2,x_3,...,x_n)=\begin{Bmatrix}\dfrac{\partial f(x_1,x_2,x_3,...,x_n)}{\partial x_1}\\ \dfrac{\partial f(x_1,x_2,x_3,...,x_n)}{\partial x_2}\\ \dfrac{\partial f(x_1,x_2,x_3,...,x_n)}{\partial x_3}\\ \cdots\\ \dfrac{\partial f(x_1,x_2,x_3,...,x_n)}{\partial x_n}\end{Bmatrix}$$

olarak verilmişti.

Bu metoda bir başlangıç noktası seçerek başlarız. Örneğin başlangıç noktası

$$P^0=f(x_1^0,x_2^0,x_3^0,...,x_n^0)$$

olsun. Önce her yeni step için türev denklemi

$d^k = -\nabla f(x_1^k, x_2^k, x_3^k, ..., x_n^k) \quad k = 0....n$

hesaplanır. Sonra bu değer kullanılarak α katsayısı cinsinden yeni değere geçilir.

$$f(\alpha^k) = f(x^k + \alpha^k d^k) = f([x_1^k - \alpha^k \frac{\partial f(x_1, x_2, x_3, ..., x_n)}{\partial x_1}], [x_2^k - \alpha^k \frac{\partial f(x_1, x_2, x_3, ..., x_n)}{\partial x2}], ..., [x_n^k - \alpha^k \frac{\partial f(x_1, x_2, x_3, ..., x_n)}{\partial x_n}])$$

$f(\alpha)$ tek boyutlu fonksiyon değerini minimum yapan α değeri tek boyutlu optimizasyon tekniklerinden birisi kullanılarak bulunur. Yeni x^{k+1} değeri

$x^{k+1} = x^k + \alpha^k d^k$ formülünü kullanarak hata küçülene kadar iteratif olarak tekrarlanır.

ÖRNEK PROBLEM 5.10-1

ÖRNEK PROBLEM

Genel formül : $\{x^{m+1}\} = \{x^m\} + \alpha^m \left[H^m\right]^{-1} \{\nabla f^m\}$

En dik yamaç formülü (genel formüldeki Hessian matrisinin değerini birim matisolarak kabul edersek):

$\{x^{m+1}\} = \{x^m\} + \alpha^m \{\nabla f^m\}$ veya yukarıda yazdığımız şekliyle $x^{m+1} = x^m + \alpha^m d^m$, $\quad d^m = \{\nabla f^m\}$

$f(x_0,x_1)=3x_0^2-4x_0x_1+2x_1^2-x_0-x_1$ fonksiyonunun $\{x\}=[-2,2]$ noktasından başlayarak minimum değerlerini bulunuz.

$$\nabla f = \begin{Bmatrix} 6x_0 - 4x_1 - 1 \\ 4x_1 - 4x_0 - 1 \end{Bmatrix} = \begin{Bmatrix} -21 \\ 15 \end{Bmatrix}$$

$$\begin{Bmatrix} x_0^0 \\ x_1^0 \end{Bmatrix} = \begin{Bmatrix} -2 \\ 2 \end{Bmatrix}$$

$$\begin{Bmatrix} x_0^1 \\ x_1^1 \end{Bmatrix} = \begin{Bmatrix} x_0^0 \\ x_1^0 \end{Bmatrix} + \alpha \begin{Bmatrix} 6x_0^0 - 4x_1^0 - 1 \\ -4x_0^0 - 4x_1^0 - 1 \end{Bmatrix}$$

$$\begin{Bmatrix} x_0^1 \\ x_1^1 \end{Bmatrix} = \begin{Bmatrix} -2 \\ 2 \end{Bmatrix} + \alpha \begin{Bmatrix} -21 \\ 15 \end{Bmatrix}$$

$x_0^1 = -2 -21\alpha$

$x_1^1 = 2 +15\alpha$

Bu değerleri

$f(x_0,x_1)=3x_0^2-4x_0x_1+2x_1^2-x_0-x_1$ fonksiyonunda yerine koyarsak

$y=3(-2-21\alpha)^2-4(-2-21\alpha)(2+15\alpha)+2(2+15\alpha)^2-(-2-21\alpha)-(2+15\alpha)$

$y = 36 +666\alpha+3033\alpha^2$ Birinci derece denklemini buluruz. Bizim optimal noktamız birinci dereceye dönüşmüş olan y fonksiyonunun minimum noktasıdır. Bu yüzden α'nın y yi minimum yapan değerini bulacağız. Programda bunun için brent metodu kullanılmıştır. Burada ise analitik olarak direk çözüm yapalım:

$y' = 666+6066\,\alpha=0 \quad \alpha=-0.109792284$

Bu değeri x_0^1 ve x_1^1 de yerine koyarsak

α^0=-0.109792284 için $\begin{Bmatrix} x_0^1 \\ x_1^1 \end{Bmatrix} = \begin{Bmatrix} 0.3056379 \\ 0.35311572 \end{Bmatrix}$ $\quad \alpha^1$=-1.121212 için $\begin{Bmatrix} x_0^2 \\ x_1^2 \end{Bmatrix} = \begin{Bmatrix} 0.9544105 \\ 1.2613973 \end{Bmatrix}$

α^2=-0.109792284 için $\begin{Bmatrix} x_0^3 \\ x_1^3 \end{Bmatrix} = \begin{Bmatrix} 0.9894 \\ 1.2363 \end{Bmatrix}$ α^3=-1.121212 için $\begin{Bmatrix} x_0^4 \\ x_1^4 \end{Bmatrix} = \begin{Bmatrix} 0.9993 \\ 1.250173 \end{Bmatrix}$

bundan sonra aynı iterasyona görüldüğü gibi devam ederiz. Programda bu işlemi yapan algorimalar mevcuttur, aynı problem çözülmektedir.

Program 5.10.1 en dik yamaç algoritması programı

```
 import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import java.io.*;
class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=3.0*x[0]*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

class f1dim extends f_x
{
  // iki boyutlu fonksiyonu tek boyutlu fonksiyona çeviren
  // pseudo fonksiyon
  f_xj ff1;
  double pc[];
  double xc[];
  double xt[];
  int n;
  public f1dim(f_xj ff,double pi[],double xi[])
  { n=pi.length;
    ff1=ff;
          pc=new double[n];
    xc=new double[n];
    xt=new double[n];
          for(int i=0;i<n;i++)
            {pc[i]=pi[i];xc[i]=xi[i];}
  }

  public double func(double x)
  {
          for (int j=0;j<n;j++) xt[j]=pc[j]+x*xc[j];
          return ff1.func(xt);
  }
}
```

```
public class OPO8
{
public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}
// 3. brent metodu
//_____________________________________________________________________
// golden (altın) oran - Fibonacchi metodu ve ikinci derece polinom metodunu birlikte kullanır

public  static double[] brentf(f_x f,double ax,double bx,double cx,double tol)
{
// ax,bx,cx üç adet tahmin noktası (bxax ile cx arasında yer almalıdır)
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double CGOLD=(3.0-Math.sqrt(5))/2.0; //altın oran
double ZEPS=1.0e-10;
double xmin;
double aa[]=new double[2];
// SHFT(double a,double b,double c,d) (a)=(b);(b)=(c);(c)=(d);
        int iter;
        double a,b,d,etemp,fu,fv,fw,fx,p,q,r,tol1,tol2,u,v,w,x,xm;
        double e=0.0;
        d=0.0;
        a=((ax < cx) ? ax : cx);
        b=((ax > cx) ? ax : cx);
        x=w=v=bx;
        fw=fv=fx=f.func(x);
        for (iter=1;iter<=ITMAX;iter++) {
                xm=0.5*(a+b);
                tol2=2.0*(tol1=tol*Math.abs(x)+ZEPS);
                if (Math.abs(x-xm) <= (tol2-0.5*(b-a))) {
                        xmin=x;
            aa[0]=xmin;
            aa[1]=fx;
            return aa;
                }
                if (Math.abs(e) > tol1) {
                        r=(x-w)*(fx-fv);
                        q=(x-v)*(fx-fw);
                        p=(x-v)*q-(x-w)*r;
                        q=2.0*(q-r);
                        if (q > 0.0) p = -p;
                        q=Math.abs(q);
                        etemp=e;
                        e=d;
                        if (Math.abs(p) >= Math.abs(0.5*q*etemp) || p <= q*(a-x) || p >= q*(b-x))
                                d=CGOLD*(e=(x >= xm ? a-x : b-x));
                        else {
                                d=p/q;
                                u=x+d;
                                if (u-a < tol2 || b-u < tol2)
                                        d=SIGN(tol1,xm-x);
```

```
                        }
                } else {
                        d=CGOLD*(e=(x >= xm ? a-x : b-x));
                }
                u=(Math.abs(d) >= tol1 ? x+d : x+SIGN(tol1,d));
                fu=f.func(u);
                if (fu <= fx) {
                        if (u >= x) a=x; else b=x;
                        {v=w;w=x;x=u;}
                        {fv=fw;fw=fx;fx=fu;}
                } else {
                        if (u < x) a=u; else b=u;
                        if (fu <= fw || w == x) {
                                v=w;
                                w=u;
                                fv=fw;
                                fw=fu;
                        } else if (fu <= fv || v == x || v == w) {
                                v=u;
                                fv=fu;
                        }
                }
        }
        System.out.println("BRENT metodunda maksimum iterasyon sayısı aşıldı");
        xmin=x;
        //minumum değer a[0] ile fonksiyon değeri a[1] ile geridöndürülmektedir
        aa[0]=xmin;
        aa[1]=fx;
        return aa;
}

public  static double brent(f_x f,double ax,double bx,double cx,double tol)
{
 double aa[]=brentf(f,ax,bx,cx,tol);
 return aa[0];
}

// 3. en dik yamac metodu
//_____________________________________________________________________
public static double[] endikyamac( f_xj f,double x[])
{
// en dik yamac metodu
// linmin ve brent metodunukullanır
double ti=1.0;
int i,ii,jj;
int nmax=5;
double tolerance=1.0e-10;
int n=x.length;
double b[];
b=new double[n];
double lambda;
double dy[]=new double[n];
```

```
//dy=new double[n][n];
i=0;
for(i=0;i<n;i++)
{
b[i]=1.0;
}
i=0;
while( i++ < nmax && Matrix.abs(b) > tolerance )
{
   dy=turev(f,x);
   lambda=linmin(f,x,dy);
   b=Matrix.multiply(dy,lambda);
   x=Matrix.add(x,b);
   System.out.print("i= "+i+"alfa="+lambda+"\n"+Matrix.toString(x));
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
public  static double[] mnbrak(f_x f,double ax,double dx)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçilen uzaklık
return mnbrak(f,ax,(ax-dx),(ax+dx));
}

public  static double[] mnbrak(f_x f,double ax)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçiln uzaklık
return mnbrak(f,ax,(ax-0.1),(ax+0.1));
}

public  static double[] mnbrak(f_x f,double ax,double bx,double cx)
{
// ax,bx,cx üç adet tahmin noktası verildiğinde içinde minimum bulunan ax,bx,cx üç adetnoktayı bize arama
// sonucu bulur
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double xmin;
double GOLD=(Math.sqrt(5.0)+1.0)/2.0;;
double GLIMIT=100.0;
double TINY=1.0e-20;
double fa,fb,fc;
fa=0;
fb=0;
fc=0;
            double ulim,u,r,q,fu,dum;
            double aa[]=new double[3];
            fa=f.func(ax);
```

```
fb=f.func(bx);
if (fb > fa) {
        //SHFT(dum,*ax,*bx,dum)
        {dum=ax;ax=bx;bx=dum;}
        //SHFT(dum,*fb,*fa,dum)
        {dum=fb;fb=fa;fa=dum;}
}
cx=(bx)+GOLD*(bx-ax);
fc=f.func(cx);
        aa[0]=ax;
        aa[1]=bx;
        aa[2]=cx;
while (fb > fc) {
        r=(bx-ax)*(fb-fc);
        q=(bx-cx)*(fb-fa);
        u=(bx)-((bx-cx)*q-(bx-ax)*r)/
                (2.0*SIGN(MAX(Math.abs(q-r),TINY),q-r));
        ulim=(bx)+GLIMIT*(cx-bx);
        if ((bx-u)*(u-cx) > 0.0) {
                fu=f.func(u);
                if (fu < fc) {
                        ax=bx;
                        bx=u;
                        fa=fb;
                        fb=fu;
                        return aa;
                } else if (fu > fb) {
                        cx=u;
                        fc=fu;
                        return aa;
                }
                u=(cx)+GOLD*(cx-bx);
                fu=f.func(u);
        } else if ((cx-u)*(u-ulim) > 0.0) {
                fu=f.func(u);
                if (fu < fc) {
                        //SHFT(bx,cx,u,cx+GOLD*(cx-bx))
                        {bx=cx;cx=u;u=cx+GOLD*(cx-bx);}
                        //SHFT(fb,fc,fu,f.func(u))
                        {fb=fc;fc=fu;fu=f.func(u);}
                }
        } else if ((u-ulim)*(ulim-cx) >= 0.0) {
                u=ulim;
                fu=f.func(u);
        } else {
                u=cx+GOLD*(cx-bx);
                fu=f.func(u);
        }
        //SHFT(ax,bx,cx,u)
        {ax=bx;bx=cx;cx=u;}
        //SHFT(fa,fb,fc,fu)
        {fa=fb;fb=fc;fc=fu;}
```

```
                        aa[0]=ax;
                        aa[1]=bx;
                        aa[2]=cx;
            }
                        return aa;
}
public static double linmin(f_xj f,double p[],double xi[])
{
            double tol=2.0e-4;
   int n=p.length;
   double pcom[],xicom[];
            int j;
            double xx,xmin,bx,ax;
            f1dim f1=new f1dim(f,p,xi);
            ax=0.0;
            xx=1.0;
            bx=2.0;
            double aa[]=mnbrak(f1,ax,xx,bx);
            ax=aa[0];
            xx=aa[1];
            bx=aa[2];
            xmin=brent(f1,ax,xx,bx,tol);
            return xmin;
}

public static double[] turev(f_xj f_deriv,double x[])
{
// df/dxj  j=0...x.length
// This method calculates turev of a function with more than one independent variable.
// Accuracy of method can be adjusted by changing variables h0 and n
// function input should be in the form given in abstract class
// f_xj,j=0...x.length = df/dx(x_ref)
double a[]=new double[x.length];
for(int x_ref=0;x_ref<x.length;x_ref++)
   {
   a[x_ref]=turev(f_deriv,x,x_ref);
   }
return a;
}

public static double turev(f_xj f,double xxi[],int x_ref)
{
// df/dxj
double h0=0.256808;
int i,m;
int n=7;
double x1[];
x1=new double[xxi.length];
double x2[];
x2=new double[xxi.length];
for(i=0;i<xxi.length;i++)
{
```

```
x1[i]=xxi[i];
x2[i]=xxi[i];
}
//turev of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
//h[i]=h0*Math.pow(r,i);
h0/=2.0;
h[i]=h0;
}
//first turev (difference formula)
for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i];
T[i][0]=(f.func(x1)-f.func(x2))/(2.0*h[i]);
x1[x_ref]=xxi[x_ref];
x2[x_ref]=xxi[x_ref];
}

for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] -
  h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i] - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}

// veri giriş çıkış metodu
//_____________________________________________________________
    public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
```

```
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }

    public static void main (String args[])
    {
            String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=verigir(s);
    f1 f_x=new f1();
    double [] r1= endikyamac(f_x,x0);
    s=" çözüm seti  : \n"+Matrix.toStringT(r1);
    String s2="En dik yamaç lineer olmıyan çok değişkenli fonksiyon optimizasyonu:";
    System.out.println(s2+s);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO8
i= 1alfa=-0.10979228486646384
      0.30563798219595041000000000        0.35311572700303606000000000
i= 2alfa=-1.121212121212094
      0.95441057458858570000000000        1.26139735635283900000000000
i= 3alfa=-0.10979228486646424
      0.98944814486025210000000000        1.23637052044449320000000000
i= 4alfa=-1.1212121212109205
      0.99930720143024550000000000        1.25017319964244830000000000
i= 5alfa=-0.10979228486540332
      0.99983964899571070000000000        1.24979287995280460000000000
En dik yamaç lineer olmıyan çok değişkenli fonksiyon optimizasyonu: çözüm seti  :
      0.99983964899571070000000000
      1.24979287995280460000000000

> Terminated with exit code 0.
```

ikinci bir örnek fonksiyona bakalım :

ÖRNEK PROBLEM 5.10-2

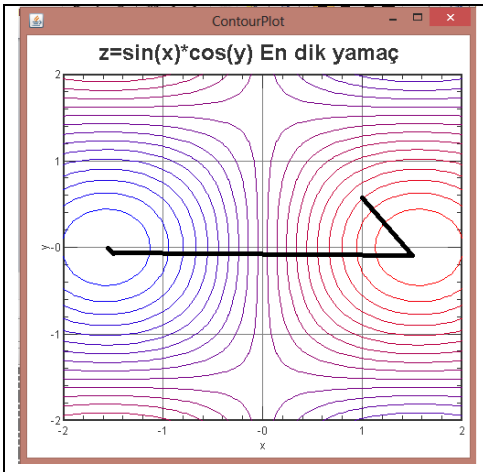
f(x0,x1)=sin(x0)cos(x1) fonksiyonunun {x}=[1,0.5] noktasından başlayarak minimum değerlerini bulunuz.
Burada sadece fonksiyon tanımını yeniden verelim, program aynı olacaktır. :

Program 5.10-2 en dik yamaç PROBLEM 5.10-2 örnek fonksiyonu

```
class f1 extends f_xj
{
public double func(double x[])
```

```
{
//çözümü istenen fonksiyon
double ff;
ff=Math.sin(x[0])*Math.cos(x[1]);
return ff;
}}
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO8
i= 1alfa=-2.3814248105049534
      -0.12917610661472700000000000        1.46072077953751900000000000
i= 2alfa=-12.174223408824403
      -1.45541326915791820000000000        -0.09803672080166970000000000
i= 3alfa=-1.0075832314505182
      -1.57085650848892660000000000        -0.00007048310398274915000000
i= 4alfa=-0.9999999999999999
      -1.57079632679509280000000000        -0.00000000000017315451373080
i= 5alfa=2.617371117109745
      -1.57079632679559130000000000        -0.00000000000065205948934140
i= 6alfa=2.617371117109745
      -1.57079632679744590000000000        -0.00000000000234541179790490
i= 7alfa=2.617371117109745
      -1.57079632680415250000000000        -0.00000000000845655140754880
i= 8alfa=2.617371117109745
      -1.57079632682836760000000000        -0.00000000003057180012360640
i= 9alfa=2.617371117109745
      -1.57079632691594880000000000        -0.00000000011060224934332880
i= 10alfa=2.617371117109745
      -1.57079632723282140000000000        -0.00000000040010169941733930
En dik yamaç lineer olmıyan çok değişkenli fonksiyon optimizasyonu: çözüm seti  :
      -1.57079632723282140000000000
      -0.00000000040010169941733930

> Terminated with exit code 0.
```



(bolum5_014.jpg,Resimaltı:En dik yamaç step ilerlemesi)

5.11Çok bilinmiyenli-lineer olmayan-geometrik-değiştirilmiş Newton metodu

Newton metodu köke yaklaştığı zaman çok iyi bir yakınsama sağlarken, kökten uzaktayken performansı daha kötüdür. Değiştirilmiş Newton metodu Newton_Raphson metoduyla En dik yamaç metodunu birleştirir.

Program 5.11-1 Değiştirilmiş Newton metodu- türevler sayısal hesaplanıyor

```
 /  Newton-Raphson metodu ile en dik yamaç metodunu birlikte kullanır.
// linmin ve brent metodunukullanır
double ti=1.0;
int i,ii,jj;
int nmax=500;
double tolerance=1.0e-12;
int n=x.length;
double b[];
b=new double[n];
double lambda;
double g[]=new double[n];
double G[][]=H(f,x);
double g1,g2,g3,g4,g5;
//dy=new double[n][n];
i=0;
for(i=0;i<n;i++)
{
b[i]=1.0;
}
while( i++ < nmax && Matrix.abs(b) > tolerance )
{
   g=turev(f,x);
   g1=Matrix.VT_V(g);
   g2=Matrix.VT_G_V(g,G);
   g1*=g1;
   g3=g1/g2;
   g4=Matrix.VT_G_V(g,Matrix.inverse(G));
   lambda=linmin(f,x,g);
   if(g4>Math.max(g3,0))
        b=Matrix.multiply(Matrix.divide(g,G),lambda);
   else
        b=Matrix.multiply(g,lambda);
   x=Matrix.add(x,b);
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
             public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
```

```
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }

            public static void main (String args[])
    {
            String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=verigir(s);
    f1 f_x=new f1();
    double [] r1= dampednewton(f_x,x0);
    s=" çözüm seti  : \n"+Matrix.toStringT(r1);
    String s2="Modifiye newton çok değişkenli fonksiyon optimizasyonu:";
     System.out.println(s2+s);
    ContourPlot pp = new ContourPlot(f_x,-2.0,2.0,100,-2.0,2.0,100,20, false, "Değiştirilmiş newton metodu",
"x", "y");
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO8A
Modifiye newton çok değişkenli fonksiyon optimizasyonu: çözüm seti  :
       0.99999999999996580000000000
       1.24999999768664160000000000

> Terminated with exit code 0.
```

5.12 Çok bilinmiyenli-lineer olmayan- geometrik Fletcher - Reeves ve Polak - Ribiere metodları (konjuge gradyen metodu)

Fletcher-Reeves metodu (konjuge gradyen metodu)

Genel iteratif optimizasyon denklemi :

$$\{x^{k+1}\}=\{x^k\}+\alpha^k\left[H^k\right]^{-1}[-\{\nabla f^k\}]=\{x^k\}+\alpha^k\left[H^k\right]^{-1}\{-g^k\}=\{x^k\}+\alpha^k\{d^k\}$$

Formundaydı. Burada değişkenlerin isimlerinde denklem yazımını kolaylaştırmak için bazı küçük değişiklikler yaparak başlayalım

$g^k=\nabla f(x_1^k,x_2^k,x_3^k,...,x_n^k)$, $k=0,...,n$ ve

$d^k=\left[H^k\right]^{-1}[-\{g^k\}]$ görüldüğü gibi bu durumda aynı genel denklem

$\{x^{k+1}\}=\{x^k\}+\alpha^k\{d^k\}$ formunda yazıldı

Formunda idi. Bu formülde en dik yamaç metodunda Hessian matrisin birim matris olduğunu kabul ederek çözüme gitmiştik. Newton-Raphson metodunda ise Hessian matrisi hesaplayım adım parametresi α'nın bir olduğunu kabul etmiştik. Konjge gradyen (birbirine dik vektörler) metodu birbirine dik iki vektör özelliklerini kullanarak Hessian matrise türev hesabı yapmadan iteratif yaklaşım yapan bir metoddur. Genellikle tüm iteratif Hessian matrisi hesaplama yaklaşım formüllerine ilk adım olarak en dik yamaç metodunun çözümüyle yaklaşılır.
(ilk adımda Hessian birim matris olarak alınır). BU durumda Fletcher-Reves metodu adımları şu şekilde tanımlanabilir.

$$d^0 = -g^0$$

$$g^k = -\nabla f(x_1^k, x_2^k, x_3^k, ..., x_n^k) \ , \ k = 0,...,n$$

k=0 için $x^{k+1} = x^k + \alpha^k d^k$ denklemini hesaplayabiliriz.

Bundan sonra $1 \le k \le n$ için

$$\beta_k = \frac{(g^k)^T (g^k)}{(g^{k-1})^T (g^{k-1})}$$

$$d^k = -g^k + \beta^k d^{k-1}$$

$$x^{k+1} = x^k + \alpha^k d^k$$

ÖRNEK PROBLEM 5.12.1

Genel formül : $\{x^{m+1}\} = \{x^m\} + \alpha^m \left[H^m \right]^{-1} \{\nabla f^m\}$

En dik yamaç formülü :

$\{x^{m+1}\} = \{x^m\} + \alpha^m \{\nabla f^m\}$ veya yukarıda yazdığımız şekliyle $x^{m+1} = x^m + \alpha^m d^m$, $d^m = \{\nabla f^m\}$

$f(x_0,x_1)=3x_0^2-4x_0x_1+2x_1^2-x_0-x_1$ fonksiyonunun {x}=[-2,2] noktasından başlayarak minimum değerlerini bulunuz.

$$\nabla f = \begin{Bmatrix} 6x_0 - 4x_1 - 1 \\ 4x_1 - 4x_0 - 1 \end{Bmatrix} = \begin{Bmatrix} -21 \\ 15 \end{Bmatrix}$$

$$\begin{Bmatrix} x_0^0 \\ x_1^0 \end{Bmatrix} = \begin{Bmatrix} -2 \\ 2 \end{Bmatrix}$$

$$\begin{Bmatrix} x_0^1 \\ x_1^1 \end{Bmatrix} = \begin{Bmatrix} x_0^0 \\ x_1^0 \end{Bmatrix} + \alpha \begin{Bmatrix} 6x_0^0 - 4x_1^0 - 1 \\ -4x_0^0 - 4x_1^0 - 1 \end{Bmatrix}$$

$$\begin{Bmatrix} x_0^1 \\ x_1^1 \end{Bmatrix} = \begin{Bmatrix} -2 \\ 2 \end{Bmatrix} + \alpha^0 \begin{Bmatrix} -21 \\ 15 \end{Bmatrix}$$

$x_0^1 = -2 -21\alpha$

$x_1^1 = 2 +15\alpha$

Bu değerleri

$f(x_0,x_1)=3x_0^2-4x_0x_1+2x_1^2-x_0-x_1$ fonksiyonunda yerine koyarsak

$y=3(-2-21\ \alpha)^2-4(-2\ -21\alpha)(2\ +15\alpha)+2(2\ +15\alpha)^2-(-2\ -21\alpha)-(2\ +15\alpha)$

$y = 36\ +666\alpha+3033\ \alpha^2$ Birinci derece denklemini buluruz. Bizim optimal noktamız birinci dereceye dönüşmüş olan y fonksiyonunun minimum noktasıdır. Bu yüzden α'nın y yi minimum yapan değerini bulacağız. Programda bunun için brent metodu kullanılmıştır. Burada ise analitik olarak direk çözüm yapalım:

$y' = 666+6066\ \alpha = 0 \quad \alpha=-0.109792284$

Bu değeri x_0^1 ve x_1^1 de yerine koyarsak

α^0=-0.109792284 için $\begin{Bmatrix} x_0^1 \\ x_1^1 \end{Bmatrix} = \begin{Bmatrix} 0.305637982195846 \\ 0.353115727002967 \end{Bmatrix}$ olur. Bu ilk stepimiz en dik yamaç stepi idi şimdi Fletcher-Reeves dik gradyan yöntemini hesaplayabiliriz.

$$g = \nabla f = \begin{Bmatrix} 6x_0 - 4x_1 - 1 \\ 4x_1 - 4x_0 - 1 \end{Bmatrix} = \begin{Bmatrix} -0.578635014836797 \\ -0.810089020771512 \end{Bmatrix}$$

$$\beta_k = \frac{(g^k)^T (g^k)}{(g^{k-1})^T (g^{k-1})} = \frac{0.9910627019697276}{666.0} = 0.001488082135089681$$

$$d^k = -g^k + \beta^k d^{k-1} = \begin{Bmatrix} 0.609884739673680 \\ 0.787767788745167 \end{Bmatrix}$$

$$x^{k+1} = x^k + \alpha^k d^k = \begin{Bmatrix} 0.3056379 \\ 0.35311572 \end{Bmatrix} + \alpha \begin{Bmatrix} 0.609884739673680 \\ 0.787767788745167 \end{Bmatrix}$$

α^1=1.138513513513513

$\begin{Bmatrix} x_0^2 \\ x_1^2 \end{Bmatrix} = \begin{Bmatrix} 1.000000000000001 \\ 1.249999999999998 \end{Bmatrix}$ olarak elde ederiz.

Problem 5.12.1 Fletcher – Reeves dik gradyan metodu ile optimizasyon, fonksiyon ve türevi verilmiş formül

```
 import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
//ff=2.0*x[0]*x[1]+2.0*x[0]-x[0]*x[0]-2.0*x[1]*x[1];
ff=3.0*x[0]*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

class fxb extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyonun türev fonksiyonları
```

```
double ff[]=new double[2];
ff[0]=6.0*x[0]-4.0*x[1]-1.0;
ff[1]=-4.0*x[0]+4.0*x[1]-1.0;
return ff;
}
}

public class OPO12E1
{
public static double VT_V(double [] left)
{
//multiplies a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*left[i];
  }
System.out.print("v = "+Matrix.toString(left)+"carpim="+tot+"\n");
return tot;
}

public static double[][] V_VT(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
for(int i=0;i<n;i++)
  { for(int j=0;j<n;j++)
     {aa[i][j]=left[i]*left[j];}
  }
return aa;
}

public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}
public static double[] df(f_xj f,double xxi[])
{
//Gredyen vektörü
// f(x0,x1,x2,...xn) founksiyonunun türev vektörü
// [df/dx0, df/dx1,df/dx2,....,df/dxn]
// df/dxj  j=0...x.length
//
// bu fonksiyon bilgisayar tarafından hesaplanmaktadır
// kullanıcı tarafından tanımlanan f(x) fonksiyonunun
// sayısal olarak alınan türevidir.
```

```
//
double a[]=new double[xxi.length];
for(int i=0;i<xxi.length;i++)
   {
   a[i]=dfdx(f,xxi,i);
   }
return a;
}


public static double dfdx(f_xj f,double xxi[],int xref)
{ int i;
            double h=1.0e-3;
  double hh=1.0/h;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
  for(i=0;i<xxi.length;i++)
  {
  if(i!=xref)
    {xp2h[i]=xxi[i];xph[i]=xxi[i];xm2h[i]=xxi[i];xmh[i]=xxi[i];}
  else {xp2h[i]=xxi[i]+2.0*h;xph[i]=xxi[i]+h;xm2h[i]=xxi[i]-2.0*h;xmh[i]=xxi[i]-h;}
  }
  fp2h=f.func(xp2h);
  fph=f.func(xph);
  fm2h=f.func(xm2h);
  fmh=f.func(xmh);
  return (-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
}

public static double d2fdx2(f_xj f,double xxi[],int xiref,int xjref)
{ //ikincitürev matrisi
  int i;
  double h=1.0e-4;
  double hh=1.0/h;
  int n=xxi.length;
  double ff;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
     for(int j=0;j<n;j++)
     {if(j!=xjref) {xp2h[j]=xxi[j];xph[j]=xxi[j];xm2h[j]=xxi[j];xmh[j]=xxi[j];}
      else {xp2h[j]=xxi[j]+2.0*h;xph[j]=xxi[j]+h;xm2h[j]=xxi[j]-2.0*h;xmh[j]=xxi[j]-h;}
     }
     fp2h=dfdx(f,xp2h,xiref);
     fph=dfdx(f,xph,xiref);
     fm2h=dfdx(f,xm2h,xiref);
```

```
    fmh=dfdx(f,xmh,xiref);
    ff=(-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
  return ff;
}



public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}


public static double linmin(f_xj f,double p[],double xi[])
{
        double tol=2.0e-4;
   int n=p.length;
   double pcom[],xicom[];
        int j;
        double xx,xmin,bx,ax;
        f1dim f1=new f1dim(f,p,xi);
        ax=0.0;
        xx=1.0;
        bx=2.0;
        double aa[]=mnbrak(f1,ax,xx,bx);
        ax=aa[0];
        xx=aa[1];
        bx=aa[2];
        xmin=brent(f1,ax,xx,bx,tol);
        return xmin;
}

public static double[][] linminiter(f_xj f,double p[],double xi[])
{   // yeni iterasyon noktasını verir
   int n=p.length;
   double xmin=linmin(f,p,xi);
   double aa[][]=new double[2][n];
        for (int j=0;j<n;j++)
        {   xi[j] *= xmin;
                p[j] += xi[j];
                aa[0][j]=p[j];
                aa[1][j]=xi[j];
        }
   return aa;
}

public  static double[] brentf(f_x f,double ax,double bx,double cx,double tol)
{
// ax,bx,cx üç adet tahmin noktası (bxax ile cx arasında yer almalıdır)
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double CGOLD=(3.0-Math.sqrt(5))/2.0; //altın oran
double ZEPS=1.0e-10;
```

```
double xmin;
double aa[]=new double[2];
// SHFT(double a,double b,double c,d) (a)=(b);(b)=(c);(c)=(d);
        int iter;
        double a,b,d,etemp,fu,fv,fw,fx,p,q,r,tol1,tol2,u,v,w,x,xm;
        double e=0.0;
        d=0.0;
        a=((ax < cx) ? ax : cx);
        b=((ax > cx) ? ax : cx);
        x=w=v=bx;
        fw=fv=fx=f.func(x);
        for (iter=1;iter<=ITMAX;iter++) {
                xm=0.5*(a+b);
                tol2=2.0*(tol1=tol*Math.abs(x)+ZEPS);
                if (Math.abs(x-xm) <= (tol2-0.5*(b-a))) {
                        xmin=x;
            aa[0]=xmin;
            aa[1]=fx;
            return aa;
                }
                if (Math.abs(e) > tol1) {
                        r=(x-w)*(fx-fv);
                        q=(x-v)*(fx-fw);
                        p=(x-v)*q-(x-w)*r;
                        q=2.0*(q-r);
                        if (q > 0.0) p = -p;
                        q=Math.abs(q);
                        etemp=e;
                        e=d;
                        if (Math.abs(p) >= Math.abs(0.5*q*etemp) || p <= q*(a-x) || p >= q*(b-x))
                                d=CGOLD*(e=(x >= xm ? a-x : b-x));
                        else {
                                d=p/q;
                                u=x+d;
                                if (u-a < tol2 || b-u < tol2)
                                        d=SIGN(tol1,xm-x);
                        }
                } else {
                        d=CGOLD*(e=(x >= xm ? a-x : b-x));
                }
                u=(Math.abs(d) >= tol1 ? x+d : x+SIGN(tol1,d));
                fu=f.func(u);
                if (fu <= fx) {
                        if (u >= x) a=x; else b=x;
                        {v=w;w=x;x=u;}
                        {fv=fw;fw=fx;fx=fu;}
                } else {
                        if (u < x) a=u; else b=u;
                        if (fu <= fw || w == x) {
                                v=w;
                                w=u;
                                fv=fw;
```

```
                                        fw=fu;
                                } else if (fu <= fv || v == x || v == w) {
                                        v=u;
                                        fv=fu;
                                }
                        }
                }
                System.out.println("BRENT metodunda maksimum iterasyon sayısı aşıldı");
                xmin=x;
                //minumum değer a[0] ile fonksiyon değeri a[1] ile geridöndürülmektedir
                aa[0]=xmin;
                aa[1]=fx;
                return aa;
}
public  static double[] mnbrak(f_x f,double ax,double dx)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçilen uzaklık
return mnbrak(f,ax,(ax-dx),(ax+dx));
}

public  static double[] mnbrak(f_x f,double ax)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçiln uzaklık
return mnbrak(f,ax,(ax-0.1),(ax+0.1));
}

public  static double[] mnbrak(f_x f,double ax,double bx,double cx)
{
// ax,bx,cx üç adet tahmin noktası verildiğinde içinde minimum bulunan ax,bx,cx üç adetnoktayı bize arama
// sonucu bulur
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double xmin;
double GOLD=(Math.sqrt(5.0)+1.0)/2.0;;
double GLIMIT=100.0;
double TINY=1.0e-20;
double fa,fb,fc;
fa=0;
fb=0;
fc=0;
                double ulim,u,r,q,fu,dum;
                double aa[]=new double[3];
                fa=f.func(ax);
                fb=f.func(bx);
                if (fb > fa) {
                        //SHFT(dum,*ax,*bx,dum)
                        {dum=ax;ax=bx;bx=dum;}
                        //SHFT(dum,*fb,*fa,dum)
                        {dum=fb;fb=fa;fa=dum;}
```

```
        }
        cx=(bx)+GOLD*(bx-ax);
        fc=f.func(cx);
                aa[0]=ax;
                aa[1]=bx;
                aa[2]=cx;
        while (fb > fc) {
                r=(bx-ax)*(fb-fc);
                q=(bx-cx)*(fb-fa);
                u=(bx)-((bx-cx)*q-(bx-ax)*r)/
                        (2.0*SIGN(MAX(Math.abs(q-r),TINY),q-r));
                ulim=(bx)+GLIMIT*(cx-bx);
                if ((bx-u)*(u-cx) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                ax=bx;
                                bx=u;
                                fa=fb;
                                fb=fu;
                                return aa;
                        } else if (fu > fb) {
                                cx=u;
                                fc=fu;
                                return aa;
                        }
                        u=(cx)+GOLD*(cx-bx);
                        fu=f.func(u);
                } else if ((cx-u)*(u-ulim) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                //SHFT(bx,cx,u,cx+GOLD*(cx-bx))
                                {bx=cx;cx=u;u=cx+GOLD*(cx-bx);}
                                //SHFT(fb,fc,fu,f.func(u))
                                {fb=fc;fc=fu;fu=f.func(u);}
                        }
                } else if ((u-ulim)*(ulim-cx) >= 0.0) {
                        u=ulim;
                        fu=f.func(u);
                } else {
                        u=cx+GOLD*(cx-bx);
                        fu=f.func(u);
                }
                //SHFT(ax,bx,cx,u)
                {ax=bx;bx=cx;cx=u;}
                //SHFT(fa,fb,fc,fu)
                {fa=fb;fb=fc;fc=fu;}
                aa[0]=ax;
                aa[1]=bx;
                aa[2]=cx;
        }
                return aa;
}
```

```
public  static double brent(f_x f,double ax,double bx,double cx,double tol)
{
 double aa[]=brentf(f,ax,bx,cx,tol);
 return aa[0];
}
// 4. fletcher Reeves metodu
//_____________________________________________________________________
public static double[] fletcher_reeves( f_xj f,fi_xi df,double x[])
{
int i,j,ii,jj,k;
int nmax=500;
double gamma=1.0;
double tolerance=1.0e-11;
int n=x.length;
double alpha;
double g[]=new double[n];
double ge[]=new double[n];
double d[]=new double[n];
double beta;
double alpha_d;
double Q[]=new double[n];
int nn=15;
i=0;
j=0;k=0;
while( j < nmax && gamma > tolerance )
{
   g=df.func(x);
   gamma=norm(g);
   if(k<1) {for(i=0;i<n;i++){d[i]=-g[i];};ge=g;}
   else
   {beta=VT_V(g)/VT_V(ge);
    for(i=0;i<n;i++){Q[i]=beta*d[i];};
   }
   for(i=0;i<n;i++){d[i]=-g[i]+Q[i];}
   alpha=linmin(f,x,d);

   for(i=0;i<n;i++){x[i]+=alpha*d[i];}
   System.out.print("alpha = "+alpha+"\nx = "+Matrix.toString(x)+"d = "+Matrix.toString(d));
   ge=g;
   j++;k++;
   if(k>=nn) k=0; //tekrar bir en dik yamaç metodu uygulaması
}
if(j >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
            public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
```

```
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }

            public static void main (String args[])
    {
            String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=verigir(s);
    f1 f_x=new f1();
    fxb df_x=new fxb();
    double [] r1= fletcher_reeves(f_x,df_x,x0);
    s=" çözüm seti  : \n"+Matrix.toStringT(r1);
    String s2="Fletcher-Reeves çok değişkenli fonksiyon optimizasyonu:";
    System.out.println(s2+s);
    }
}
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO12E1
alpha = 0.10979228486646883
x =         0.30563798219584550000000000        0.35311572700296745000000000
d =        21.00000000000000000000000000       -15.00000000000000000000000000
v =        -0.57863501483679690000000000       -0.81008902077151210000000000
carpim=0.9910627019697276
v =       -21.00000000000000000000000000        15.00000000000000000000000000
carpim=666.0
alpha = 1.138513513513513
x =         1.00000000000000130000000000        1.24999999999999820000000000
d =         0.60988473967368020000000000        0.78776778874516700000000000
v =         0.00000000000001509903313490       -0.00000000000001243449787580
carpim=3.8259753903219073E-28
v =        -0.57863501483679690000000000       -0.81008902077151210000000000
carpim=0.9910627019697276
alpha = 3.8548836452020043
x =         0.99999999999994320000000000        1.25000000000004620000000000
d =        -0.00000000000001509903313490        0.00000000000001243449787580
Fletcher-Reeves çok değişkenli fonksiyon optimizasyonu: çözüm seti  :
       0.99999999999994320000000000
       1.25000000000004620000000000

> Terminated with exit code 0.
```

Problem 5.12.2 Fletcher – Reeves dik gradyan metodu ile optimizasyon, fonksiyon verilmiş ve türevi sayısal olarak hesaplanıyor

```
 import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
//ff=2.0*x[0]*x[1]+2.0*x[0]-x[0]*x[0]-2.0*x[1]*x[1];
ff=3.0*x[0]*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

public class OPO12E
{

public static double VT_V(double [] left)
{
//multiplies a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*left[i];
  }
System.out.println("v=\n"+Matrix.toString(left)+"carpim="+tot);
return tot;
}

public static double[][] V_VT(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
for(int i=0;i<n;i++)
  {
  for(int j=0;j<n;j++)
    {
            aa[i][j]=left[i]*left[j];
            }
  }
return aa;
}

public static double norm(double v[])
{
// vector norm
```

```
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}
//====================

public static double[] df(f_xj f,double xxi[])
{
//Gredyen vektörü
// f(x0,x1,x2,...xn) founksiyonunun türev vektörü
// [df/dx0, df/dx1,df/dx2,....,df/dxn]
// df/dxj  j=0...x.length
//
// bu fonksiyon bilgisayar tarafından hesaplanmaktadır
// kullanıcı tarafından tanımlanan f(x) fonksiyonunun
// sayısal olarak alınan türevidir.

//
double a[]=new double[xxi.length];
for(int i=0;i<xxi.length;i++)
   {
   a[i]=dfdx(f,xxi,i);
   }
return a;
}


public static double dfdx(f_xj f,double xxi[],int xref)
{ int i;
            double h=1.0e-3;
  double hh=1.0/h;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
  for(i=0;i<xxi.length;i++)
  {
  if(i!=xref)
    {xp2h[i]=xxi[i];xph[i]=xxi[i];xm2h[i]=xxi[i];xmh[i]=xxi[i];}
  else {xp2h[i]=xxi[i]+2.0*h;xph[i]=xxi[i]+h;xm2h[i]=xxi[i]-2.0*h;xmh[i]=xxi[i]-h;}
  }
  fp2h=f.func(xp2h);
  fph=f.func(xph);
  fm2h=f.func(xm2h);
  fmh=f.func(xmh);
  return (-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
}

public static double d2fdx2(f_xj f,double xxi[],int xiref,int xjref)
{ //ikincitürev matrisi
```

```
  int i;
  double h=1.0e-4;
  double hh=1.0/h;
  int n=xxi.length;
  double ff;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
     for(int j=0;j<n;j++)
     {if(j!=xjref) {xp2h[j]=xxi[j];xph[j]=xxi[j];xm2h[j]=xxi[j];xmh[j]=xxi[j];}
      else {xp2h[j]=xxi[j]+2.0*h;xph[j]=xxi[j]+h;xm2h[j]=xxi[j]-2.0*h;xmh[j]=xxi[j]-h;}
     }
     fp2h=dfdx(f,xp2h,xiref);
     fph=dfdx(f,xph,xiref);
     fm2h=dfdx(f,xm2h,xiref);
     fmh=dfdx(f,xmh,xiref);
     ff=(-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
  return ff;
}
//===================
public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}


public static double linmin(f_xj f,double p[],double xi[])
{
            double tol=2.0e-4;
    int n=p.length;
    double pcom[],xicom[];
            int j;
            double xx,xmin,bx,ax;
            f1dim f1=new f1dim(f,p,xi);
            ax=0.0;
            xx=1.0;
            bx=2.0;
            double aa[]=mnbrak(f1,ax,xx,bx);
            ax=aa[0];
            xx=aa[1];
            bx=aa[2];
            xmin=brent(f1,ax,xx,bx,tol);
            return xmin;
}

public static double[][] linminiter(f_xj f,double p[],double xi[])
{   // yeni iterasyon noktasını verir
    int n=p.length;
    double xmin=linmin(f,p,xi);
    double aa[][]=new double[2][n];
            for (int j=0;j<n;j++)
            {   xi[j] *= xmin;
```

```
                    p[j] += xi[j];
                    aa[0][j]=p[j];
                    aa[1][j]=xi[j];
            }
   return aa;
}
public  static double[] brentf(f_x f,double ax,double bx,double cx,double tol)
{
// ax,bx,cx üç adet tahmin noktası (bxax ile cx arasında yer almalıdır)
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double CGOLD=(3.0-Math.sqrt(5))/2.0; //altın oran
double ZEPS=1.0e-10;
double xmin;
double aa[]=new double[2];
// SHFT(double a,double b,double c,d) (a)=(b);(b)=(c);(c)=(d);
        int iter;
        double a,b,d,etemp,fu,fv,fw,fx,p,q,r,tol1,tol2,u,v,w,x,xm;
        double e=0.0;
        d=0.0;
        a=((ax < cx) ? ax : cx);
        b=((ax > cx) ? ax : cx);
        x=w=v=bx;
        fw=fv=fx=f.func(x);
        for (iter=1;iter<=ITMAX;iter++) {
                xm=0.5*(a+b);
                tol2=2.0*(tol1=tol*Math.abs(x)+ZEPS);
                if (Math.abs(x-xm) <= (tol2-0.5*(b-a))) {
                        xmin=x;
            aa[0]=xmin;
            aa[1]=fx;
            return aa;
                }
                if (Math.abs(e) > tol1) {
                        r=(x-w)*(fx-fv);
                        q=(x-v)*(fx-fw);
                        p=(x-v)*q-(x-w)*r;
                        q=2.0*(q-r);
                        if (q > 0.0) p = -p;
                        q=Math.abs(q);
                        etemp=e;
                        e=d;
                        if (Math.abs(p) >= Math.abs(0.5*q*etemp) || p <= q*(a-x) || p >= q*(b-x))
                                d=CGOLD*(e=(x >= xm ? a-x : b-x));
                        else {
                                d=p/q;
                                u=x+d;
                                if (u-a < tol2 || b-u < tol2)
                                        d=SIGN(tol1,xm-x);
                        }
                } else {
```

```
                        d=CGOLD*(e=(x >= xm ? a-x : b-x));
                }
                u=(Math.abs(d) >= tol1 ? x+d : x+SIGN(tol1,d));
                fu=f.func(u);
                if (fu <= fx) {
                        if (u >= x) a=x; else b=x;
                        {v=w;w=x;x=u;}
                        {fv=fw;fw=fx;fx=fu;}
                } else {
                        if (u < x) a=u; else b=u;
                        if (fu <= fw || w == x) {
                                v=w;
                                w=u;
                                fv=fw;
                                fw=fu;
                        } else if (fu <= fv || v == x || v == w) {
                                v=u;
                                fv=fu;
                        }
                }
        }
        System.out.println("BRENT metodunda maksimum iterasyon sayısı aşıldı");
        xmin=x;
        //minumum değer a[0] ile fonksiyon değeri a[1] ile geridöndürülmektedir
        aa[0]=xmin;
        aa[1]=fx;
        return aa;
}
public  static double[] mnbrak(f_x f,double ax,double dx)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçilen uzaklık
return mnbrak(f,ax,(ax-dx),(ax+dx));
}

public  static double[] mnbrak(f_x f,double ax)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçiln uzaklık
return mnbrak(f,ax,(ax-0.1),(ax+0.1));
}

public  static double[] mnbrak(f_x f,double ax,double bx,double cx)
{
// ax,bx,cx üç adet tahmin noktası verildiğinde içinde minimum bulunan ax,bx,cx üç adetnoktayı bize arama
// sonucu bulur
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double xmin;
double GOLD=(Math.sqrt(5.0)+1.0)/2.0;;
double GLIMIT=100.0;
```

```
double TINY=1.0e-20;
double fa,fb,fc;
fa=0;
fb=0;
fc=0;
        double ulim,u,r,q,fu,dum;
        double aa[]=new double[3];
        fa=f.func(ax);
        fb=f.func(bx);
        if (fb > fa) {
                //SHFT(dum,*ax,*bx,dum)
                {dum=ax;ax=bx;bx=dum;}
                //SHFT(dum,*fb,*fa,dum)
                {dum=fb;fb=fa;fa=dum;}
        }
        cx=(bx)+GOLD*(bx-ax);
        fc=f.func(cx);
                aa[0]=ax;
                aa[1]=bx;
                aa[2]=cx;
        while (fb > fc) {
                r=(bx-ax)*(fb-fc);
                q=(bx-cx)*(fb-fa);
                u=(bx)-((bx-cx)*q-(bx-ax)*r)/
                        (2.0*SIGN(MAX(Math.abs(q-r),TINY),q-r));
                ulim=(bx)+GLIMIT*(cx-bx);
                if ((bx-u)*(u-cx) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                ax=bx;
                                bx=u;
                                fa=fb;
                                fb=fu;
                                return aa;
                        } else if (fu > fb) {
                                cx=u;
                                fc=fu;
                                return aa;
                        }
                        u=(cx)+GOLD*(cx-bx);
                        fu=f.func(u);
                } else if ((cx-u)*(u-ulim) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                //SHFT(bx,cx,u,cx+GOLD*(cx-bx))
                                {bx=cx;cx=u;u=cx+GOLD*(cx-bx);}
                                //SHFT(fb,fc,fu,f.func(u))
                                {fb=fc;fc=fu;fu=f.func(u);}
                        }
                } else if ((u-ulim)*(ulim-cx) >= 0.0) {
                        u=ulim;
                        fu=f.func(u);
```

```
                        } else {
                                u=cx+GOLD*(cx-bx);
                                fu=f.func(u);
                        }
                        //SHFT(ax,bx,cx,u)
                        {ax=bx;bx=cx;cx=u;}
                        //SHFT(fa,fb,fc,fu)
                        {fa=fb;fb=fc;fc=fu;}
                        aa[0]=ax;
                        aa[1]=bx;
                        aa[2]=cx;
                }
                        return aa;
}
public  static double brent(f_x f,double ax,double bx,double cx,double tol)
{
 double aa[]=brentf(f,ax,bx,cx,tol);
 return aa[0];
}
// 4. fletcher Reeves metodu
//_____________________________________________________________________
public static double[] fletcher_reeves( f_xj f,double x[])
{
int i,j,ii,jj,k;
int nmax=500;
double gamma=1.0;
double tolerance=1.0e-11;
int n=x.length;
double alpha;
double g[]=new double[n];
double ge[]=new double[n];
double d[]=new double[n];
double beta;
double alpha_d;
double Q[]=new double[n];
int nn=15;
i=0;
j=0;k=0;
while( j < nmax && gamma > tolerance )
{
   g=df(f,x);
   gamma=norm(g);
   if(k<1) {for(i=0;i<n;i++){d[i]=-g[i];};ge=g;}
   else
   {beta=VT_V(g)/VT_V(ge);
    for(i=0;i<n;i++){Q[i]=beta*d[i];};
   }
   for(i=0;i<n;i++){d[i]=-g[i]+Q[i];}
   alpha=linmin(f,x,d);
   for(i=0;i<n;i++){x[i]+=alpha*d[i];}
   ge=g;
   j++;k++;
```

```
    if(k>=nn) k=0; //tekrar bir en dik yaöaç metodu uygulaması
}
if(j >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
            public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }

     public static void main (String args[])
    {
    String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=verigir(s);
    f1 f_x=new f1();
    double [] r1= fletcher_reeves(f_x,x0);
        s=" çözüm seti  : \n"+Matrix.toStringT(r1);
    String s2="Fletcher-Reeves çok değişkenli fonksiyon optimizasyonu:";
    System.out.println(s2+s);
    System.exit(0);
    }

}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO12E
Fletcher-Reeves çok değişkenli fonksiyon optimizasyonu: çözüm seti  :
       0.99999999999998379000000000
       1.24999999999999707000000000

> Terminated with exit code 0.
```

POLAK-RİBİERE METODU (KONJUGE GRADYEN METODU)

Polak-Ribiere metodu Fletcher reeves metodu formülünde küçük bir değişikliği içeren benzer bir metoddur. Fletcher-Reeves metodunda β değeri

$\beta_k = \frac{(g^k)^T (g^k)}{(g^{k-1})^T (g^{k-1})}$ olarak tanımlanmıştı. Bu tanımı biraz değiştirirsek,

$\beta_k = \frac{(g^{k-1} - g^k)^T (g^k)}{(g^{k-1})^T (g^{k-1})}$ tanımıyla verecek olursak Polak-Ribiere metodunu elde etmiş oluruz.

Problem 5.12.2 Polak-Ribiere dik gradyan metodu ile optimizasyon

```
 import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=3.0*x[0]*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
return ff;
}
}

public class OPO12E4
{

public static double VT_V(double [] left)
{
//multiplies a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*left[i];
  }
//System.out.println("v=\n"+Matrix.toString(left)+"carpim="+tot);
return tot;
}

public static double[][] V_VT(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
for(int i=0;i<n;i++)
  {
  for(int j=0;j<n;j++)
    {
            aa[i][j]=left[i]*left[j];
            }
  }
return aa;
}
```

```
public static double  VT_X(double [] left,double [] right)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*right[i];
  }
return tot;
}

public static double[] substract(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]-right[i];
  }
return b;
//end of vector substraction method
}

public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}
//===================

public static double[] df(f_xj f,double xxi[])
{
//Gredyen vektörü
// f(x0,x1,x2,...xn) founksiyonunun türev vektörü
// [df/dx0, df/dx1,df/dx2,....,df/dxn]
// df/dxj  j=0...x.length
//
```

```
// bu fonksiyon bilgisayar tarafından hesaplanmaktadır
// kullanıcı tarafından tanımlanan f(x) fonksiyonunun
// sayısal olarak alınan türevidir.

//
double a[]=new double[xxi.length];
for(int i=0;i<xxi.length;i++)
   {
   a[i]=dfdx(f,xxi,i);
   }
return a;
}


public static double dfdx(f_xj f,double xxi[],int xref)
{ int i;
            double h=1.0e-3;
  double hh=1.0/h;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
  for(i=0;i<xxi.length;i++)
  {
  if(i!=xref)
    {xp2h[i]=xxi[i];xph[i]=xxi[i];xm2h[i]=xxi[i];xmh[i]=xxi[i];}
  else {xp2h[i]=xxi[i]+2.0*h;xph[i]=xxi[i]+h;xm2h[i]=xxi[i]-2.0*h;xmh[i]=xxi[i]-h;}
  }
  fp2h=f.func(xp2h);
  fph=f.func(xph);
  fm2h=f.func(xm2h);
  fmh=f.func(xmh);
  return (-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
}

public static double d2fdx2(f_xj f,double xxi[],int xiref,int xjref)
{ //ikincitürev matrisi
  int i;
  double h=1.0e-4;
  double hh=1.0/h;
  int n=xxi.length;
  double ff;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
     for(int j=0;j<n;j++)
     {if(j!=xjref) {xp2h[j]=xxi[j];xph[j]=xxi[j];xm2h[j]=xxi[j];xmh[j]=xxi[j];}
      else {xp2h[j]=xxi[j]+2.0*h;xph[j]=xxi[j]+h;xm2h[j]=xxi[j]-2.0*h;xmh[j]=xxi[j]-h;}
     }
```

```
    fp2h=dfdx(f,xp2h,xiref);
    fph=dfdx(f,xph,xiref);
    fm2h=dfdx(f,xm2h,xiref);
    fmh=dfdx(f,xmh,xiref);
    ff=(-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
  return ff;
}
//===================
public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}


public static double linmin(f_xj f,double p[],double xi[])
{
        double tol=2.0e-4;
   int n=p.length;
   double pcom[],xicom[];
        int j;
        double xx,xmin,bx,ax;
        f1dim f1=new f1dim(f,p,xi);
        ax=0.0;
        xx=1.0;
        bx=2.0;
        double aa[]=mnbrak(f1,ax,xx,bx);
        ax=aa[0];
        xx=aa[1];
        bx=aa[2];
        xmin=brent(f1,ax,xx,bx,tol);
        return xmin;
}

public static double[][] linminiter(f_xj f,double p[],double xi[])
{   // yeni iterasyon noktasını verir
   int n=p.length;
   double xmin=linmin(f,p,xi);
   double aa[][]=new double[2][n];
        for (int j=0;j<n;j++)
        {   xi[j] *= xmin;
                p[j] += xi[j];
                aa[0][j]=p[j];
                aa[1][j]=xi[j];
        }
   return aa;
}
public  static double[] brentf(f_x f,double ax,double bx,double cx,double tol)
{
// ax,bx,cx üç adet tahmin noktası (bxax ile cx arasında yer almalıdır)
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double CGOLD=(3.0-Math.sqrt(5))/2.0; //altın oran
double ZEPS=1.0e-10;
```

```
double xmin;
double aa[]=new double[2];
// SHFT(double a,double b,double c,d) (a)=(b);(b)=(c);(c)=(d);
        int iter;
        double a,b,d,etemp,fu,fv,fw,fx,p,q,r,tol1,tol2,u,v,w,x,xm;
        double e=0.0;
        d=0.0;
        a=((ax < cx) ? ax : cx);
        b=((ax > cx) ? ax : cx);
        x=w=v=bx;
        fw=fv=fx=f.func(x);
        for (iter=1;iter<=ITMAX;iter++) {
                xm=0.5*(a+b);
                tol2=2.0*(tol1=tol*Math.abs(x)+ZEPS);
                if (Math.abs(x-xm) <= (tol2-0.5*(b-a))) {
                        xmin=x;
            aa[0]=xmin;
            aa[1]=fx;
            return aa;
                }
                if (Math.abs(e) > tol1) {
                        r=(x-w)*(fx-fv);
                        q=(x-v)*(fx-fw);
                        p=(x-v)*q-(x-w)*r;
                        q=2.0*(q-r);
                        if (q > 0.0) p = -p;
                        q=Math.abs(q);
                        etemp=e;
                        e=d;
                        if (Math.abs(p) >= Math.abs(0.5*q*etemp) || p <= q*(a-x) || p >= q*(b-x))
                                d=CGOLD*(e=(x >= xm ? a-x : b-x));
                        else {
                                d=p/q;
                                u=x+d;
                                if (u-a < tol2 || b-u < tol2)
                                        d=SIGN(tol1,xm-x);
                        }
                } else {
                        d=CGOLD*(e=(x >= xm ? a-x : b-x));
                }
                u=(Math.abs(d) >= tol1 ? x+d : x+SIGN(tol1,d));
                fu=f.func(u);
                if (fu <= fx) {
                        if (u >= x) a=x; else b=x;
                        {v=w;w=x;x=u;}
                        {fv=fw;fw=fx;fx=fu;}
                } else {
                        if (u < x) a=u; else b=u;
                        if (fu <= fw || w == x) {
                                v=w;
                                w=u;
                                fv=fw;
```

```
                                        fw=fu;
                                } else if (fu <= fv || v == x || v == w) {
                                        v=u;
                                        fv=fu;
                                }
                        }
                }
                System.out.println("BRENT metodunda maksimum iterasyon sayısı aşıldı");
                xmin=x;
                //minumum değer a[0] ile fonksiyon değeri a[1] ile geridöndürülmektedir
                aa[0]=xmin;
                aa[1]=fx;
                return aa;
}
public  static double[] mnbrak(f_x f,double ax,double dx)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçilen uzaklık
return mnbrak(f,ax,(ax-dx),(ax+dx));
}

public  static double[] mnbrak(f_x f,double ax)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçiln uzaklık
return mnbrak(f,ax,(ax-0.1),(ax+0.1));
}

public  static double[] mnbrak(f_x f,double ax,double bx,double cx)
{
// ax,bx,cx üç adet tahmin noktası verildiğinde içinde minimum bulunan ax,bx,cx üç adetnoktayı bize arama
// sonucu bulur
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double xmin;
double GOLD=(Math.sqrt(5.0)+1.0)/2.0;;
double GLIMIT=100.0;
double TINY=1.0e-20;
double fa,fb,fc;
fa=0;
fb=0;
fc=0;
                double ulim,u,r,q,fu,dum;
                double aa[]=new double[3];
                fa=f.func(ax);
                fb=f.func(bx);
                if (fb > fa) {
                        //SHFT(dum,*ax,*bx,dum)
                        {dum=ax;ax=bx;bx=dum;}
                        //SHFT(dum,*fb,*fa,dum)
                        {dum=fb;fb=fa;fa=dum;}
```

```
	}
	cx=(bx)+GOLD*(bx-ax);
	fc=f.func(cx);
		aa[0]=ax;
		aa[1]=bx;
		aa[2]=cx;
	while (fb > fc) {
		r=(bx-ax)*(fb-fc);
		q=(bx-cx)*(fb-fa);
		u=(bx)-((bx-cx)*q-(bx-ax)*r)/
			(2.0*SIGN(MAX(Math.abs(q-r),TINY),q-r));
		ulim=(bx)+GLIMIT*(cx-bx);
		if ((bx-u)*(u-cx) > 0.0) {
			fu=f.func(u);
			if (fu < fc) {
				ax=bx;
				bx=u;
				fa=fb;
				fb=fu;
				return aa;
			} else if (fu > fb) {
				cx=u;
				fc=fu;
				return aa;
			}
			u=(cx)+GOLD*(cx-bx);
			fu=f.func(u);
		} else if ((cx-u)*(u-ulim) > 0.0) {
			fu=f.func(u);
			if (fu < fc) {
				//SHFT(bx,cx,u,cx+GOLD*(cx-bx))
				{bx=cx;cx=u;u=cx+GOLD*(cx-bx);}
				//SHFT(fb,fc,fu,f.func(u))
				{fb=fc;fc=fu;fu=f.func(u);}
			}
		} else if ((u-ulim)*(ulim-cx) >= 0.0) {
			u=ulim;
			fu=f.func(u);
		} else {
			u=cx+GOLD*(cx-bx);
			fu=f.func(u);
		}
		//SHFT(ax,bx,cx,u)
		{ax=bx;bx=cx;cx=u;}
		//SHFT(fa,fb,fc,fu)
		{fa=fb;fb=fc;fc=fu;}
		aa[0]=ax;
		aa[1]=bx;
		aa[2]=cx;
	}
		return aa;
}
```

```
public  static double brent(f_x f,double ax,double bx,double cx,double tol)
{
 double aa[]=brentf(f,ax,bx,cx,tol);
 return aa[0];
}
// 4. Polak Ribiere Metodu
//_____________________________________________________________________
public static double[] polak_ribiere( f_xj f,double x[])
{
int i,j,ii,jj,k;
int nmax=500;
double gamma=1.0;
double tolerance=1.0e-7;
int n=x.length;
double alpha;
double g[]=new double[n];
double ge[]=new double[n];
double d[]=new double[n];
double beta;
double alpha_d;
double Q[]=new double[n];
int nn=15;
i=0;
j=0;k=0;
while( j < nmax && gamma > tolerance )
{
   g=df(f,x);
   gamma=norm(g);
   if(k<1) {for(i=0;i<n;i++){d[i]=-g[i];};ge=g;}
   else
   {   // Polack-Ribiere conjugate gradient tanımı
              beta=VT_X(substract(ge,g),g)/VT_V(ge);
    for(i=0;i<n;i++){Q[i]=beta*d[i];};
   }
   for(i=0;i<n;i++){d[i]=-g[i]+Q[i];}
   alpha=linmin(f,x,d);
   for(i=0;i<n;i++){x[i]+=alpha*d[i];}
   ge=g;
   j++;k++;
   if(k>=nn) k=0; //tekrar bir en dik yaöaç metodu uygulaması
}
if(j >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
            public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
```

```
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }

        public static void main (String args[])
    {
        String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=verigir(s);
    f1 f_x=new f1();
    double [] r1= polak_ribiere(f_x,x0);
    s=" çözüm seti  : \n"+Matrix.toStringT(r1);
    String s2="Polak-Ribiere çok değişkenli fonksiyon optimizasyonu:";
    System.out.println(s2+s);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO12E4
Polak-Ribiere çok değişkenli fonksiyon optimizasyonu: çözüm seti  :
      1.000000037167308000000000
      1.250000055437962400000000

> Terminated with exit code 0.
```

5.13 Çok bilinmiyenli-lineer olmayan-geometrik Davidon-Fletcher-Powell metodu

Newton-Raphson metodunda

$$\{\delta^{k+1}\} = -\left[H^k\right]^{-1}\{\nabla f^k\}$$
$$\{x^{k+1}\} = \{x^{k+1}\} - \left[H^k\right]^{-1}\{\nabla f^k\}$$

şeklinde çözüme ulaşabiliyorduk. Bu denkleme adım miktarını ayarlayabileceğimiz (en dik yamaç metodunda olduğu gibi) α_k faktörünü eklersek, genel denklem

$\{x^{k+1}\} = \{x^{k+1}\} - \alpha_k \left[H^k\right]^{-1}\{\nabla f^k\}$ formunu alır.

$g^k = \nabla f(x_1^k, x_2^k, x_3^k, ..., x_n^k)$, $k = 0,...,n$ ve

$d^k = \left[H^k\right]^{-1} g^k$

$x^{k+1} = x^k - \alpha^k d^k$ şeklinde yazılabilir.

Bu metodumuzda ikinci türev, Hessian yerine türevi hesaplamadan onun yerine geçebilecek bir fonksiyon oluşturmaya çalışacağız.

$Q_k = \left[H^k\right]^{-1} \quad , k \geq 0$ olarak tanımlayalım.

$d^k = Q_k \, g^k$

$x^{k+1} = x^k - \alpha_k \, Q_k g^k = x^k - \alpha_k \, d_k$

şeklinde yazabiliriz. Davidon – Fletcher – Powell metodu Q yu hesaplamak için

$\Delta g^k = g^{k+1} - g^k$

$r^k = Q_k \Delta g^k$

$\Delta x^k = \{\delta^{k+1}\} = x^{k+1} - x^k$ olarak yazarsak

$$Q_{k+1} = Q_k - \frac{r^k (r^k)^T}{(r^k)^T \Delta g^k} + \frac{\Delta x^k (\Delta x^k)^T}{(\Delta x^k)^T \Delta g^k} \quad , \quad 0 \leq k < n$$

formülünü kullanır. Program 5.13-1 de Davidon-Fletcher-Powell metodu algoritması verilmiştir. Programımımızda Q hesaplanmasında ikinci bir versiyon daha kullanılmıştır ikinci denklemde yeni Q değeri bir önceki değer kullanılarak modifiye edilmektedir.

$$Q_{k+1} = \beta_k [Q_k - \frac{r^k (r^k)^T}{(r^k)^T \Delta g^k}] + \frac{\Delta x^k (\Delta x^k)^T}{(\Delta x^k)^T \Delta g^k} \quad , \quad 0 \leq k < n$$

$$\beta_k = \frac{(\Delta x^k)^T \Delta x^k}{(r^k)^T \Delta g^k} \quad , \quad 0 \leq k < n$$

İteratif çözüm yönteminde iterasyona Q_0 değeri birim matrise eşit alınarak başlanır. Q_0=I

$$x^{k+1} = x^k - \alpha_k \, Q_k g^k = x^k - \alpha_k \, d_k$$

denklemindeki α_k katsayısı birinci dereceden tek boyutlu optimizasyon metodlarından birisi kullanılarak bulunur.

Bizim programımızda brent yöntemi kullanılmıştır. Bu birinci step temel olarak tepe tırmanma metodudur. Daha sonra Q değerleri verilen denklemle hesaplanarak Davidon-Fletcher-Powell iterasyon stepleri hesaplanır. Her n stepte Q=I değerine geri dönülerek bir step tepe tırmanma metodu kullanılır ve tekrar Davidon-Fletcher-Powell iterasyon steplerine dönülür.

ÖRNEK PROBLEM 5.13.1

Genel formül : $\{x^{m+1}\} = \{x^m\} + \alpha^m \left[H^m\right]^{-1} \{\nabla f^m\}$

En dik yamaç formülü :

$\{x^{m+1}\} = \{x^m\} + \alpha^m \{\nabla f^m\}$ veya yukarıda yazdığımız şekliyle $x^{m+1} = x^m + \alpha^m d^m$, $\quad d^m = \{\nabla f^m\}$

$f(x_0,x_1)=3x_0^2-4x_0x_1+2x_1^2-x_0-x_1$ fonksiyonunun {x}=[-2,2] noktasından başlayarak minimum değerlerini bulunuz.

$$\nabla f = \begin{Bmatrix} 6x_0 - 4x_1 - 1 \\ 4x_1 - 4x_0 - 1 \end{Bmatrix} = \begin{Bmatrix} -21 \\ 15 \end{Bmatrix}$$

$$\begin{Bmatrix} x_0^0 \\ x_1^0 \end{Bmatrix} = \begin{Bmatrix} -2 \\ 2 \end{Bmatrix}$$

$$\begin{Bmatrix} x_0^1 \\ x_1^1 \end{Bmatrix} = \begin{Bmatrix} x_0^0 \\ x_1^0 \end{Bmatrix} + \alpha \begin{Bmatrix} 6x_0^0 - 4x_1^0 - 1 \\ -4x_0^0 - 4x_1^0 - 1 \end{Bmatrix}$$

$$\begin{Bmatrix} x_0^1 \\ x_1^1 \end{Bmatrix} = \begin{Bmatrix} -2 \\ 2 \end{Bmatrix} + \alpha^0 \begin{Bmatrix} -21 \\ 15 \end{Bmatrix}$$

x_0^1 = -2 -21α

x_1^1 = 2 +15α

Bu değerleri

$f(x_0,x_1)=3x_0^2-4x_0x_1+2x_1^2-x_0-x_1$ fonksiyonunda yerine koyarsak

$y=3(-2-21\ \alpha)^2-4(-2\ -21\alpha)(2\ +15\alpha)+2(2\ +15\alpha)^2-(-2\ -21\alpha)-(2\ +15\alpha)$

$y = 36 +666\alpha+3033\ \alpha^2$ Birinci derece denklemini buluruz. Bizim optimal noktamız birinci dereceye dönüşmüş olan y fonksiyonunun minimum noktasıdır. Bu yüzden α'nın y yi minimum yapan değerini bulacağız. Programda bunun için brent metodu kullanılmıştır. Burada ise analitik olarak direk çözüm yapalım:

$y' = 666+6066\ \alpha = 0 \quad \alpha=-0.109792284$

Bu değeri x_0^1 ve x_1^1 de yerine koyarsak

α^0=-0.109792284 için $\begin{Bmatrix} x_0^1 \\ x_1^1 \end{Bmatrix} = \begin{Bmatrix} 0.305637982195846 \\ 0.353115727002967 \end{Bmatrix}$ olur. Bu ilk stepimiz en dik yamaç stepi idi şimdi Davidon-Fletcher-Powell dik gradyan yöntemini hesaplayabiliriz.

$$g = \nabla f = \begin{Bmatrix} 6x_0 - 4x_1 - 1 \\ 4x_1 - 4x_0 - 1 \end{Bmatrix} = \begin{Bmatrix} -0.578635014836797 \\ -0.810089020771512 \end{Bmatrix}$$

$$\Delta g^k = g^{k+1} - g^k = = \begin{Bmatrix} 20.421364985163200 \\ -15.810089020771512 \end{Bmatrix}$$

$$Q = \begin{Bmatrix} 1 & 0 \\ 0 & 1 \end{Bmatrix}$$

$$r^k = Q_k \Delta g^k = \begin{Bmatrix} 20.421364985163200 \\ -15.810089020771512 \end{Bmatrix}$$

$$\Delta x^k = \{\delta^{k+1}\} = x^{k+1} - x^k = \begin{Bmatrix} 2.305637982195846 \\ -1.646884272997033 \end{Bmatrix}$$

$$Q_{k+1} = Q_k - \frac{r^k (r^k)^T}{(r^k)^T \Delta g^k} + \frac{\Delta x^k (\Delta x^k)^T}{(\Delta x^k)^T \Delta g^k} = \begin{Bmatrix} 0.447456314953212 & 0.432131073481232 \\ 0.432131073481232 & 0.662335969913257 \end{Bmatrix}$$

$$d^k = Q_k\, g^k = \begin{Bmatrix} 0.6089785296031 23 \\ 0.7865972674040 34 \end{Bmatrix}$$

$$x^{k+1} = x^k + \alpha^k d^k \quad = \begin{Bmatrix} 0.3056379821958 46 \\ 0.3531157270029 67 \end{Bmatrix} + \alpha \begin{Bmatrix} 0.6089785296031 23 \\ 0.7865972674040 34 \end{Bmatrix}$$

Bu değerleri Bu değerleri
$f(x_0,x_1)=3x_0^2-4x_0x_1+2x_1^2-x_0-x_1$ fonksiyonunda yerine koyarsak ve çözersek :

$\alpha^1=1.1402077151335304$ bulunur. Buradan da :

$$\begin{Bmatrix} x_0^2 \\ x_1^2 \end{Bmatrix} = \begin{Bmatrix} 1.000000000000000 \\ 1.250000000000000 \end{Bmatrix}$$

olarak ikinci iterasyon sonuçlarını elde ederiz.

Program 5.13-1 Davidon-Fletcher-Powell metodu (türev sayısal olarak hesaplanıyor)

```
 // OPO12F Davidon-Fletcher-Powell optimizasyonu
// Metod Referansı : Numerical Analysis, 8th edition Richard L. Burden, J. Douglas Faires 3
// sayfa 617
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
//ff=7.9+0.13*x[0]+0.21*x[1]-0.05*x[0]*x[0]-0.016*x[1]*x[1]-0.007*x[0]*x[1];
//ff=2.0*x[0]*x[1]+2.0*x[0]-x[0]*x[0]-2.0*x[1]*x[1];
ff=3.0*x[0]*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}



public class OPO12F
{
public static double VT_V(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*left[i];
  }
```

```
return tot;
}

public static double[][] V_VT(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
for(int i=0;i<n;i++)
  {
  for(int j=0;j<n;j++)
    {
            aa[i][j]=left[i]*left[j];
            }
  }
return aa;
}

public static double  VT_X(double [] left,double [] right)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*right[i];
  }
return tot;
}

public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}

public static double turev(f_xj f,double x[],int denklem_ref,int x_ref)
{
//
// verilen fonksiyonun ikinci türevlerinin matrisi
// fonksiyon f in denklem_ref sayılı fonksiyonunun(sayılar 0 dan başlar)
// x[x_ref] değişkenine göre türevi (sayılar 0 dan başlar)
// df_denklem_ref(x)/d_x_ref
// bu metod newtond metodu içinde kullanılmak içindir.
double h0=0.256808;
int i,m;
int n=7;
double f1[];
f1=new double[x.length];
```

```
double f2[];
f2=new double[x.length];
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
x2[i]=x[i];
}
//turev of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i]*1.0094847;
f1=df(f,x1);
f2=df(f,x2);
T[i][0]=( f1[denklem_ref] - f2[denklem_ref])/(2.0*h[i]);
x1[x_ref]=x[x_ref];
x2[x_ref]=x[x_ref];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}
public static double[] df(f_xj f,double xxi[])
```

```
{
//Gredyen vektörü
// f(x0,x1,x2,...xn) founksiyonunun türev vektörü
// [df/dx0, df/dx1,df/dx2,....,df/dxn]
// df/dxj  j=0...x.length
//
// bu fonksiyon bilgisayar tarafından hesaplanmaktadır
// kullanıcı tarafından tanımlanan f(x) fonksiyonunun
// sayısal olarak alınan türevidir.

//
double a[]=new double[xxi.length];
for(int i=0;i<xxi.length;i++)
   {
   a[i]=turev(f,xxi,i);
   }
return a;
}
public static double turev(f_xj f,double xxi[],int x_ref)
{
// df/dxj
double h0=0.256808;
int i,m;
int n=7;
double x1[];
x1=new double[xxi.length];
double x2[];
x2=new double[xxi.length];
for(i=0;i<xxi.length;i++)
{
x1[i]=xxi[i];
x2[i]=xxi[i];
}
//turev of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
//h[i]=h0*Math.pow(r,i);
h0/=2.0;
h[i]=h0;
}
```

```
//first turev (difference formula)
for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i];
T[i][0]=(f.func(x1)-f.func(x2))/(2.0*h[i]);
x1[x_ref]=xxi[x_ref];
x2[x_ref]=xxi[x_ref];
}

for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] -
  h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i] - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}
public static double[] turev(f_xj f_deriv,double x[])
{
// df/dxj  j=0...x.length
// This method calculates turev of a function with more than one independent variable.
// Accuracy of method can be adjusted by changing variables h0 and n
// function input should be in the form given in abstract class
// f_xj,j=0...x.length = df/dx(x_ref)
double a[]=new double[x.length];
for(int x_ref=0;x_ref<x.length;x_ref++)
   {
   a[x_ref]=turev(f_deriv,x,x_ref);
   }
return a;
}

public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}



public static double linmin(f_xj f,double p[],double xi[])
{
        double tol=2.0e-4;
    int n=p.length;
    double pcom[],xicom[];
        int j;
        double xx,xmin,bx,ax;
        f1dim f1=new f1dim(f,p,xi);
        ax=0.0;
        xx=1.0;
        bx=2.0;
        double aa[]=mnbrak(f1,ax,xx,bx);
```

```
            ax=aa[0];
            xx=aa[1];
            bx=aa[2];
            xmin=brent(f1,ax,xx,bx,tol);
            return xmin;
}

public static double[][] linminiter(f_xj f,double p[],double xi[])
{   // yeni iterasyon noktasını verir
    int n=p.length;
    double xmin=linmin(f,p,xi);
    double aa[][]=new double[2][n];
            for (int j=0;j<n;j++)
            {   xi[j] *= xmin;
                        p[j] += xi[j];
                        aa[0][j]=p[j];
                        aa[1][j]=xi[j];
            }
    return aa;
}
public  static double[] brentf(f_x f,double ax,double bx,double cx,double tol)
{
// ax,bx,cx üç adet tahmin noktası (bxax ile cx arasında yer almalıdır)
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double CGOLD=(3.0-Math.sqrt(5))/2.0; //altın oran
double ZEPS=1.0e-10;
double xmin;
double aa[]=new double[2];
// SHFT(double a,double b,double c,d) (a)=(b);(b)=(c);(c)=(d);
            int iter;
            double a,b,d,etemp,fu,fv,fw,fx,p,q,r,tol1,tol2,u,v,w,x,xm;
            double e=0.0;
            d=0.0;
            a=((ax < cx) ? ax : cx);
            b=((ax > cx) ? ax : cx);
            x=w=v=bx;
            fw=fv=fx=f.func(x);
            for (iter=1;iter<=ITMAX;iter++) {
                        xm=0.5*(a+b);
                        tol2=2.0*(tol1=tol*Math.abs(x)+ZEPS);
                        if (Math.abs(x-xm) <= (tol2-0.5*(b-a))) {
                                    xmin=x;
                    aa[0]=xmin;
                    aa[1]=fx;
                    return aa;
                        }
                        if (Math.abs(e) > tol1) {
                                    r=(x-w)*(fx-fv);
                                    q=(x-v)*(fx-fw);
                                    p=(x-v)*q-(x-w)*r;
```

```
                        q=2.0*(q-r);
                        if (q > 0.0) p = -p;
                        q=Math.abs(q);
                        etemp=e;
                        e=d;
                        if (Math.abs(p) >= Math.abs(0.5*q*etemp) || p <= q*(a-x) || p >= q*(b-x))
                                d=CGOLD*(e=(x >= xm ? a-x : b-x));
                        else {
                                d=p/q;
                                u=x+d;
                                if (u-a < tol2 || b-u < tol2)
                                        d=SIGN(tol1,xm-x);
                        }
                } else {
                        d=CGOLD*(e=(x >= xm ? a-x : b-x));
                }
                u=(Math.abs(d) >= tol1 ? x+d : x+SIGN(tol1,d));
                fu=f.func(u);
                if (fu <= fx) {
                        if (u >= x) a=x; else b=x;
                        {v=w;w=x;x=u;}
                        {fv=fw;fw=fx;fx=fu;}
                } else {
                        if (u < x) a=u; else b=u;
                        if (fu <= fw || w == x) {
                                v=w;
                                w=u;
                                fv=fw;
                                fw=fu;
                        } else if (fu <= fv || v == x || v == w) {
                                v=u;
                                fv=fu;
                        }
                }
        }
        System.out.println("BRENT metodunda maksimum iterasyon sayısı aşıldı");
        xmin=x;
        //minumum değer a[0] ile fonksiyon değeri a[1] ile geridöndürülmektedir
        aa[0]=xmin;
        aa[1]=fx;
        return aa;
}
public  static double[] mnbrak(f_x f,double ax,double dx)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçilen uzaklık
return mnbrak(f,ax,(ax-dx),(ax+dx));
}

public  static double[] mnbrak(f_x f,double ax)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
```

```
// ana noktadan seçiln uzaklık
return mnbrak(f,ax,(ax-0.1),(ax+0.1));
}

public  static double[] mnbrak(f_x f,double ax,double bx,double cx)
{
// ax,bx,cx üç adet tahmin noktası verildiğinde içinde minimum bulunan ax,bx,cx üç adetnoktayı bize arama
// sonucu bulur
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double xmin;
double GOLD=(Math.sqrt(5.0)+1.0)/2.0;;
double GLIMIT=100.0;
double TINY=1.0e-20;
double fa,fb,fc;
fa=0;
fb=0;
fc=0;
        double ulim,u,r,q,fu,dum;
        double aa[]=new double[3];
        fa=f.func(ax);
        fb=f.func(bx);
        if (fb > fa) {
                //SHFT(dum,*ax,*bx,dum)
                {dum=ax;ax=bx;bx=dum;}
                //SHFT(dum,*fb,*fa,dum)
                {dum=fb;fb=fa;fa=dum;}
        }
        cx=(bx)+GOLD*(bx-ax);
        fc=f.func(cx);
                aa[0]=ax;
                aa[1]=bx;
                aa[2]=cx;
        while (fb > fc) {
                r=(bx-ax)*(fb-fc);
                q=(bx-cx)*(fb-fa);
                u=(bx)-((bx-cx)*q-(bx-ax)*r)/
                        (2.0*SIGN(MAX(Math.abs(q-r),TINY),q-r));
                ulim=(bx)+GLIMIT*(cx-bx);
                if ((bx-u)*(u-cx) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                ax=bx;
                                bx=u;
                                fa=fb;
                                fb=fu;
                                return aa;
                        } else if (fu > fb) {
                                cx=u;
                                fc=fu;
                                return aa;
```

```
                        }
                        u=(cx)+GOLD*(cx-bx);
                        fu=f.func(u);
                } else if ((cx-u)*(u-ulim) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                //SHFT(bx,cx,u,cx+GOLD*(cx-bx))
                                {bx=cx;cx=u;u=cx+GOLD*(cx-bx);}
                                //SHFT(fb,fc,fu,f.func(u))
                                {fb=fc;fc=fu;fu=f.func(u);}
                        }
                } else if ((u-ulim)*(ulim-cx) >= 0.0) {
                        u=ulim;
                        fu=f.func(u);
                } else {
                        u=cx+GOLD*(cx-bx);
                        fu=f.func(u);
                }
                //SHFT(ax,bx,cx,u)
                {ax=bx;bx=cx;cx=u;}
                //SHFT(fa,fb,fc,fu)
                {fa=fb;fb=fc;fc=fu;}
                aa[0]=ax;
                aa[1]=bx;
                aa[2]=cx;
        }
                return aa;
}
public  static double brent(f_x f,double ax,double bx,double cx,double tol)
{
 double aa[]=brentf(f,ax,bx,cx,tol);
 return aa[0];
}

public static double[][] I(int n)
{
//unit matrix
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
   b[i][i]=1.0;
return b;
}
// Davidon - Fletcher-Powell metodu
//_____________________________________________________________________
public static double[] davidon_fletcher_powell( f_xj f,double x[])
{
int i,j,k;
int nmax=500;
double gamma=1.0;
double tolerance=1.0e-12 ;
int n=x.length;
```

```
double alpha;
double dx[]=new double[n];
double g[] =new double[n];
double ge[]=new double[n];
double dg[]=new double[n];
double d[] =new double[n];
double r[] =new double[n];
double rdg,dxdg;
double beta;
double alpha_d;
double Q[][]=I(n);
double Q1[][]=new double[n][n];
double Q2[][]=new double[n][n];
int nn=15;
i=0;
j=0;k=0;
g=turev(f,x);
while( j++ < nmax && gamma > tolerance )
{
   for(int ii=0;ii<n;ii++)
   {
    d[ii]=0.0;
    for(int kk=0;kk<n;kk++)
            {d[ii]=d[ii]-Q[ii][kk]*g[kk];}
   }
   System.out.print("j = "+j+"d = "+Matrix.toString(d));
   alpha=linmin(f,x,d);
   System.out.println("alfa="+alpha);
   if(alpha==0) break;
   for(i=0;i<n;i++){dx[i]=alpha*d[i];x[i]+=dx[i];ge[i]=g[i];}
   System.out.print(" j = "+j+" dx = "+Matrix.toString(dx));
   System.out.print("j = "+j+" x= "+Matrix.toString(x));
   g=turev(f,x);
   System.out.print("j = "+ j+" g = "+Matrix.toString(g));
   for(i=0;i<n;i++){dg[i]=g[i]-ge[i];}
   gamma=norm(g);
   for(int ii=0;ii<n;ii++)
   {r[ii]=0.0;
            for(int kk=0;kk<n;kk++)
            {r[ii]+=Q[ii][kk]*dg[kk];}
   }
   rdg=VT_X(r,dg);
   dxdg=VT_X(dx,dg);
   Q1=V_VT(r);
   Q2=V_VT(dx);
   for(int ii=0;ii<n;ii++)
     for(int jj=0;jj<n;jj++)
       Q[ii][jj]=Q[ii][jj]-Q1[ii][jj]/rdg+Q2[ii][jj]/dxdg;
   gamma=norm(dx);
   System.out.print(" j = "+j+" Q =  "+Matrix.toString(Q));
j++;k++;
if(k>=nn) {k=0;Q=I(n);}
```

```
}
if(j >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
public static double[] davidon_fletcher_powell_V1( f_xj f,double x[])
{
int i,j,k;
int nmax=500;
double gamma=1.0;
double tolerance=1.0e-12 ;
int n=x.length;
double alpha;
double dx[]=new double[n];
double g[] =new double[n];
double ge[]=new double[n];
double dg[]=new double[n];
double d[] =new double[n];
double r[] =new double[n];
double rdg,dxdg;
double beta;
double alpha_d;
double Q[][]=I(n);
double Q1[][]=new double[n][n];
double Q2[][]=new double[n][n];
int nn=15;
i=0;
j=0;k=0;
g=turev(f,x);
while( j++ < nmax && gamma > tolerance )
{

   for(int ii=0;ii<n;ii++)
   {
    d[ii]=0.0;
    for(int kk=0;kk<n;kk++)
            {d[ii]=d[ii]-Q[ii][kk]*g[kk];}
   }
   alpha=linmin(f,x,d);
   if(alpha==0) break;
   for(i=0;i<n;i++){dx[i]=alpha*d[i];x[i]+=dx[i];ge[i]=g[i];}
   g=turev(f,x);
   for(i=0;i<n;i++){dg[i]=g[i]-ge[i];}
   gamma=norm(g);
   for(int ii=0;ii<n;ii++)
   {for(int kk=0;kk<n;kk++)
            {r[ii]=Q[ii][kk]*dg[kk];}
   }
   rdg=VT_X(r,dg);
   dxdg=VT_X(dx,dg);
   beta=dxdg/rdg;
```

```
    Q1=V_VT(r);
    Q2=V_VT(dx);
    for(int ii=0;ii<n;ii++)
      for(int jj=0;jj<n;jj++)
        Q[ii][jj]=beta*(Q[ii][jj]-Q1[ii][jj]/rdg)+Q2[ii][jj]/dxdg;
    gamma=norm(dx);
j++;k++;
if(k>=nn) {k=0;Q=I(n);}
}
if(j >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
            public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }
            public static void main (String args[])
    {
            String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=verigir(s);
    f1 f_x=new f1();
    double [] r1= davidon_fletcher_powell(f_x,x0);
    double [] r2= davidon_fletcher_powell_V1(f_x,x0);
    s=" çözüm seti DFP : \n"+Matrix.toStringT(r1);
    s+=" çözüm seti düzeltilmiş DFP : \n"+Matrix.toStringT(r2);
    String s2="Davidon - Fletcher - Powell çok değişkenli fonksiyon optimizasyonu:";
    System.out.println(s2+s);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO12F
j = 1d =        3.000000000000817000000000        -3.000000000000940000000000
alfa=0.11111111111110793
 j = 1 dx =        0.333333333333328700000000        -0.333333333333342600000000
j = 1 x=        1.333333333333328000000000        1.666666666666656000000000
j = 1 g =        0.333333333334531000000000        0.333333333332924600000000
 j = 1 Q =        0.445799457994572        0.432249322493227
       0.432249322493227        0.665311653116533
```

```
j = 3d =        -0.29268292682930197000000000        -0.36585365853661120000000000
alfa=1.1388888888887798
 j = 3 dx =        -0.33333333333333975000000000        -0.41666666666665620000000000
j = 3 x=        0.99999999999999310000000000        1.25000000000000930000000000
j = 3 g =        -0.00000000000000825535400664        -0.00000000000000245827211616
 j = 3 Q =        0.499999999999876        0.499999999999864
       0.499999999999864        0.749999999999838
j = 5d =        0.00000000000005356813061400        0.00000000000005971381090440
alfa=0.6188820480799224
 j = 5 dx =        0.00000000000003315235438620        0.00000000000003695580559120
j = 5 x=        1.00000000000002620000000000        1.25000000000004620000000000
j = 5 g =        -0.00000000000006740314594830        0.00000000000011023759649060
 j = 5 Q =        0.344286249180296        0.207211216441036
       0.207211216441036        0.250826244777427
Davidon - Fletcher - Powell çok değişkenli fonksiyon optimizasyonu: çözüm seti DFP :
       1.00000000000005280000000000
       1.25000000000000270000000000
 çözüm seti düzeltilmiş DFP :
       1.00000000000005280000000000
       1.25000000000000270000000000

> Terminated with exit code 0.
```

5.14 Çok bilinmiyenli-lineer olmayan-geometrik direk hessian değiştirme metodu : Broyden-Fletcher-Goldberg-Shanno (BFGS)

Önceki iki konuda dik iki vektör (conjuge gradyan) kullanarak İkinci türev matrisinin (Hessian matrisi) ters matrisinin döngüsel olarak oluşturulmasını vere Fletcher-Reeves, Polak-Ribiere ve Davidon-Fletcher-Powell metodlarını incelemiştik. Metodumuzu Hessian matrisinin direk olarak yine benzer dik iki vektör döngüsel yöntemleriyle hesaplanıp ters matrisinin çözümle hesaplanması şeklinde de oluşturabiliriz. Bu tür metod örneği olarak Broyden-Fletcher-Goldberg ve Shanno metodunu verebiliriz. Bu metodu Bir algoritma olarak adım adım tanımlıyalım:

1. Bir ilk tahmin vektörü, $x^{(0)}$ verelim. Hessian matrisinin ilk tahmin değeri olarak birim vektörü alalım. $H^{(0)} = I$. Dönüştürme parametresi ε ve maksimum döngü (iterasyon) sayısı nmax'ı tanımlıyalım. K=0 olarak döngüye başlayalım
   $c^{(0)} = \nabla f(x^{(0)})$ birinci türev (gradyan) vektörünü hesaplayalım.
2. Birinci türev (gradyan) vektörünün normunu $\left\|c^{(k)}\right\|$ hesaplayalım. Eğer $\left\|c^{(k)}\right\| < \varepsilon$ ise döngüyü durduralım sonuca ulaşmış bulunuyoruz.
3. Yeni arama yönü vektörü d'yi çözmek için $H^{(k)}d^{(k)} = -c^{(k)}$ lineer denklem sistemini çözelim.
4. d yönündeki optimum değer olan $\alpha^{(k)}$ değerini $f(x^{(k)} + \alpha^{(k)}d^{(k)})$ tek boyutlu fonksiyonunun minimum değerini bularak saptayalım. Burada k=0 için olan ilk çözümün en dik yamaç problemine eşdeğer olduğunu not edelim
5. x vektörünün yeni değerini hesaplayalım $x^{(k+1)} = x^{(k)} + \alpha^{(k)}d^{(k)}$
6. Hessian matrisinin yeni değerini hesaplayalım:
   $H^{(k+1)} = H^{(k)} + D^{(k)} + E^{(k)}$
   $D^{(k)} = \dfrac{y^{(k)}y^{(k)T}}{(y^{(k)}.s^{(k)})}$ $\quad$ $E^{(k)} = \dfrac{c^{(k)}c^{(k)T}}{(c^{(k)}.d^{(k)})}$
   $s^{(k)} = \alpha^{(k)}d^{(k)}$ $\quad$ $y^{(k)} = c^{(k+1)} - c^{(k)}$
   $c^{(k+1)} = \nabla f(x^{(k+1)})$

7. k'yi bir arttır k=k+1 ve step 2 ye geri dön. Eğer k değeri maksimum döngü (iterasyon) sayısı nmax a ulaşmışsa uyarı mesajı ver ve en son döngü değerlerini çıktı olarak ver.

Bu algoritmada birinci türev (gradyan) vektörü dışarıdan girilen bir sınıfta tanımlanacağı gibi, sayısal türv programları tarafından da hesaplanabilir. Örnek programımızda iki durum için de metod verilmiştir. Örnek olarak yine bir önceki bölümde incelediğimiz $f(x_0,x_1)=3x_0^2-4x_0x_1+2x_1^2-x_0-x_1$ fonksiyonunun $\{x\}=[-2,2]$ noktasından başlayarak minimum değerlerini bulma problemini irdeleyelim. Birinci ve ikinci döngü değerlerini elde hesaplama ödev olarak size bırakılmıştır.

Program 5.14-1 Broyden - Fletcher-Goldfarb-Shanno (BFGS) metodu

```
 // Broyden - Fletcher-Goldfarb-Shanno (BFGS) metodu
// Metod Referansı : Intruduction to Optimum Design, Second Edition
// Jasbir S. Arora, Elsevier, ISBN: 0-12-064155-0
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=3.0*x[0]*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
//ff=5.0*x[0]*x[0]+2.0*x[0]*x[1]+x[1]*x[1]+7.0;
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

class df1 extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyonun türev fonksiyonları
double ff[]=new double[2];
//ff[0]=10.0*x[0]+2.0*x[1];
//ff[1]=2.0*x[0]+2.0*x[1];
ff[0]=6*x[0]-4.0*x[1]-1;
ff[1]=-4.0*x[0]+4.0*x[1]-1;
return ff;
}
}



public class OPO12I1
{
public static double VT_V(double [] left)
{
//multiplys a vector transpose with a vector
```

```
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*left[i];
  }
return tot;
}

public static double[][] V_VT(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
for(int i=0;i<n;i++)
  {
  for(int j=0;j<n;j++)
    {
            aa[i][j]=left[i]*left[j];
            }
  }
return aa;
}

public static double[][] V_VT_y_s(double [] left,double [] y,double s[])
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
double ss=y_s(y,s);
double x1=y_s(y,s);
for(int i=0;i<n;i++)
  {
  for(int j=0;j<n;j++)
    {aa[i][j]=left[i]*left[j]/ss;}
  }
return aa;

}

public static double y_s(double [] y,double [] s)
{
//scalar multiply two vectors
int n=y.length;
double aa=0;
for(int i=0;i<n;i++)
  {aa+=y[i]*s[i];}
return aa;
}

public static double  VT_X(double [] left,double [] right)
{
```

```
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*right[i];
  }
return tot;
}

public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}

public static double turev(f_xj f,double x[],int denklem_ref,int x_ref)
{
//
// verilen fonksiyonun ikinci türevlerinin matrisi
// fonksiyon f in denklem_ref sayılı fonksiyonunun(sayılar 0 dan başlar)
// x[x_ref] değişkenine göre türevi (sayılar 0 dan başlar)
// df_denklem_ref(x)/d_x_ref
// bu metod newtond metodu içinde kullanılmak içindir.
double h0=0.256808;
int i,m;
int n=7;
double f1[];
f1=new double[x.length];
double f2[];
f2=new double[x.length];
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
x2[i]=x[i];
}
//turev of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
```

```
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i]*1.0094847;
f1=df(f,x1);
f2=df(f,x2);
T[i][0]=( f1[denklem_ref] - f2[denklem_ref])/(2.0*h[i]);
x1[x_ref]=x[x_ref];
x2[x_ref]=x[x_ref];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}

public static double[] df(f_xj f,double xxi[])
{
//Gredyen vektörü
// f(x0,x1,x2,...xn) founksiyonunun türev vektörü
// [df/dx0, df/dx1,df/dx2,....,df/dxn]
// df/dxj  j=0...x.length
//
// bu fonksiyon bilgisayar tarafından hesaplanmaktadır
// kullanıcı tarafından tanımlanan f(x) fonksiyonunun
// sayısal olarak alınan türevidir.

//
double a[]=new double[xxi.length];
for(int i=0;i<xxi.length;i++)
   {
   a[i]=turev(f,xxi,i);
   }
return a;
}
public static double turev(f_xj f,double xxi[],int x_ref)
```

```
{
// df/dxj
double h0=0.256808;
int i,m;
int n=7;
double x1[];
x1=new double[xxi.length];
double x2[];
x2=new double[xxi.length];
for(i=0;i<xxi.length;i++)
{
x1[i]=xxi[i];
x2[i]=xxi[i];
}
//turev of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
//h[i]=h0*Math.pow(r,i);
h0/=2.0;
h[i]=h0;
}
//first turev (difference formula)
for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i];
T[i][0]=(f.func(x1)-f.func(x2))/(2.0*h[i]);
x1[x_ref]=xxi[x_ref];
x2[x_ref]=xxi[x_ref];
}

for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] -
  h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i] - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
```

```
return xx;
}

public static double[] turev(f_xj f_deriv,double x[])
{
// df/dxj  j=0...x.length
// This method calculates turev of a function with more than one independent variable.
// Accuracy of method can be adjusted by changing variables h0 and n
// function input should be in the form given in abstract class
// f_xj,j=0...x.length = df/dx(x_ref)
double a[]=new double[x.length];
for(int x_ref=0;x_ref<x.length;x_ref++)
  {
  a[x_ref]=turev(f_deriv,x,x_ref);
  }
return a;
}

public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}


public static double linmin(f_xj f,double p[],double xi[])
{
        double tol=2.0e-10;
   int n=p.length;
   double pcom[],xicom[];
        int j;
        double xx,xmin,bx,ax;
        f1dim f1=new f1dim(f,p,xi);
        ax=0.0;
        xx=1.0;
        bx=2.0;
        double aa[]=mnbrak(f1,ax,xx,bx);
        ax=aa[0];
        xx=aa[1];
        bx=aa[2];
        xmin=brent(f1,ax,xx,bx,tol);
        return xmin;
}

public static double[][] linminiter(f_xj f,double p[],double xi[])
{  // yeni iterasyon noktasını verir
   int n=p.length;
   double xmin=linmin(f,p,xi);
   double aa[][]=new double[2][n];
        for (int j=0;j<n;j++)
        {  xi[j] *= xmin;
                p[j] += xi[j];
                aa[0][j]=p[j];
                aa[1][j]=xi[j];
        }
```

```
   return aa;
}
public  static double[] brentf(f_x f,double ax,double bx,double cx,double tol)
{
// ax,bx,cx üç adet tahmin noktası (bxax ile cx arasında yer almalıdır)
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double CGOLD=(3.0-Math.sqrt(5))/2.0; //altın oran
double ZEPS=1.0e-12;
double xmin;
double aa[]=new double[2];
// SHFT(double a,double b,double c,d) (a)=(b);(b)=(c);(c)=(d);
        int iter;
        double a,b,d,etemp,fu,fv,fw,fx,p,q,r,tol1,tol2,u,v,w,x,xm;
        double e=0.0;
        d=0.0;
        a=((ax < cx) ? ax : cx);
        b=((ax > cx) ? ax : cx);
        x=w=v=bx;
        fw=fv=fx=f.func(x);
        for (iter=1;iter<=ITMAX;iter++) {
                xm=0.5*(a+b);
                tol2=2.0*(tol1=tol*Math.abs(x)+ZEPS);
                if (Math.abs(x-xm) <= (tol2-0.5*(b-a))) {
                        xmin=x;
            aa[0]=xmin;
            aa[1]=fx;
            return aa;
                }
                if (Math.abs(e) > tol1) {
                        r=(x-w)*(fx-fv);
                        q=(x-v)*(fx-fw);
                        p=(x-v)*q-(x-w)*r;
                        q=2.0*(q-r);
                        if (q > 0.0) p = -p;
                        q=Math.abs(q);
                        etemp=e;
                        e=d;
                        if (Math.abs(p) >= Math.abs(0.5*q*etemp) || p <= q*(a-x) || p >= q*(b-x))
                                d=CGOLD*(e=(x >= xm ? a-x : b-x));
                        else {
                                d=p/q;
                                u=x+d;
                                if (u-a < tol2 || b-u < tol2)
                                        d=SIGN(tol1,xm-x);
                        }
                } else {
                        d=CGOLD*(e=(x >= xm ? a-x : b-x));
                }
                u=(Math.abs(d) >= tol1 ? x+d : x+SIGN(tol1,d));
                fu=f.func(u);
```

```
                    if (fu <= fx) {
                        if (u >= x) a=x; else b=x;
                        {v=w;w=x;x=u;}
                        {fv=fw;fw=fx;fx=fu;}
                    } else {
                        if (u < x) a=u; else b=u;
                        if (fu <= fw || w == x) {
                            v=w;
                            w=u;
                            fv=fw;
                            fw=fu;
                        } else if (fu <= fv || v == x || v == w) {
                            v=u;
                            fv=fu;
                        }
                    }
            }
            System.out.println("BRENT metodunda maksimum iterasyon sayısı aşıldı");
            xmin=x;
            //minumum değer a[0] ile fonksiyon değeri a[1] ile geridöndürülmektedir
            aa[0]=xmin;
            aa[1]=fx;
            return aa;
}
public  static double[] mnbrak(f_x f,double ax,double dx)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçilen uzaklık
return mnbrak(f,ax,(ax-dx),(ax+dx));
}

public  static double[] mnbrak(f_x f,double ax)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçiln uzaklık
return mnbrak(f,ax,(ax-0.1),(ax+0.1));
}

public  static double[] mnbrak(f_x f,double ax,double bx,double cx)
{
// ax,bx,cx üç adet tahmin noktası verildiğinde içinde minimum bulunan ax,bx,cx üç adetnoktayı bize arama
// sonucu bulur
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double xmin;
double GOLD=(Math.sqrt(5.0)+1.0)/2.0;;
double GLIMIT=100.0;
double TINY=1.0e-20;
double fa,fb,fc;
fa=0;
fb=0;
```

```
fc=0;
        double ulim,u,r,q,fu,dum;
        double aa[]=new double[3];
        fa=f.func(ax);
        fb=f.func(bx);
        if (fb > fa) {
                //SHFT(dum,*ax,*bx,dum)
                {dum=ax;ax=bx;bx=dum;}
                //SHFT(dum,*fb,*fa,dum)
                {dum=fb;fb=fa;fa=dum;}
        }
        cx=(bx)+GOLD*(bx-ax);
        fc=f.func(cx);
                aa[0]=ax;
                aa[1]=bx;
                aa[2]=cx;
        while (fb > fc) {
                r=(bx-ax)*(fb-fc);
                q=(bx-cx)*(fb-fa);
                u=(bx)-((bx-cx)*q-(bx-ax)*r)/
                        (2.0*SIGN(MAX(Math.abs(q-r),TINY),q-r));
                ulim=(bx)+GLIMIT*(cx-bx);
                if ((bx-u)*(u-cx) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                ax=bx;
                                bx=u;
                                fa=fb;
                                fb=fu;
                                return aa;
                        } else if (fu > fb) {
                                cx=u;
                                fc=fu;
                                return aa;
                        }
                        u=(cx)+GOLD*(cx-bx);
                        fu=f.func(u);
                } else if ((cx-u)*(u-ulim) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                //SHFT(bx,cx,u,cx+GOLD*(cx-bx))
                                {bx=cx;cx=u;u=cx+GOLD*(cx-bx);}
                                //SHFT(fb,fc,fu,f.func(u))
                                {fb=fc;fc=fu;fu=f.func(u);}
                        }
                } else if ((u-ulim)*(ulim-cx) >= 0.0) {
                        u=ulim;
                        fu=f.func(u);
                } else {
                        u=cx+GOLD*(cx-bx);
                        fu=f.func(u);
                }
```

```
                        //SHFT(ax,bx,cx,u)
                        {ax=bx;bx=cx;cx=u;}
                        //SHFT(fa,fb,fc,fu)
                        {fa=fb;fb=fc;fc=fu;}
                        aa[0]=ax;
                        aa[1]=bx;
                        aa[2]=cx;
            }
                        return aa;
}
public  static double brent(f_x f,double ax,double bx,double cx,double tol)
{
 double aa[]=brentf(f,ax,bx,cx,tol);
 return aa[0];
}

public static double[][] I(int n)
{
//unit matrix
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
   b[i][i]=1.0;
return b;
}

// Broyden - Fletcher-Goldfarb-Shanno (BFGS) metodu
//
// sadece  fonksiyon verilmiş türev sayısal hesaplanıyor
//_____________________________________________________________________

public static double[] BFGS( f_xj f,double x[])
{
int i,j,k;
int nmax=100;
double gamma=1.0;
double tolerance=1.0e-6 ;
int n=x.length;
double alpha;
double s[]=new double[n];
double c[] =new double[n];
double c1[] =new double[n];
double d[] =new double[n];
double y[] =new double[n];
double alpha_d;
double H[][]=I(n);
double H1[][]=I(n);
//double H[][]={{1,0},{0,0}};
double D[][]=new double[n][n];
double E[][]=new double[n][n];
int nn=15;
j=0;k=0;
```

```
c=turev(f,x);
for(i=0;i<c.length;i++)
{d[i]=-c[i];} //first iteration : steepest descent
gamma=norm(c);
while( k++ < nmax && gamma > tolerance )
{
   alpha=linmin(f,x,d);
   for(i=0;i<n;i++)
   {s[i]=alpha*d[i];
    x[i]+=s[i];
   }
   c1=turev(f,x);
   for(i=0;i<n;i++)
   {y[i]=c1[i]-c[i];}
   D=V_VT_y_s(y,y,s);
   E=V_VT_y_s(c,c,d);
   for(i=0;i<n;i++)
   {for(j=0;j<n;j++)
   {H[i][j]=H[i][j]+D[i][j]+E[i][j];
    H1[i][j]=H[i][j];
   }
   }
   for(i=0;i<n;i++)
   {c[i]=c1[i];
    c1[i]=-c1[i];
   }
   d=pivotlugauss(H1,c1);
   gamma=norm(c);
}
if(k >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}

// fonksiyon ve türev fonksiyonu verilmiş
//_____________________________________________________________

public static double[] BFGS( f_xj f,fi_xi df,double x[])
{
int i,j,k;
int nmax=100;
double gamma=1.0;
double tolerance=1.0e-6 ;
int n=x.length;
double alpha;
double s[]=new double[n];
double c[] =new double[n];
double c1[] =new double[n];
double d[] =new double[n];
double y[] =new double[n];
double alpha_d;
```

```
double H[][]=I(n);
double H1[][]=I(n);
//double H[][]={{1,0},{0,0}};
double D[][]=new double[n][n];
double E[][]=new double[n][n];
int nn=15;
j=0;k=0;
c=df.func(x);
for(i=0;i<c.length;i++)
{d[i]=-c[i];} //first iteration : steepest descent
gamma=norm(c);
while( k++ < nmax && gamma > tolerance )
{
   alpha=linmin(f,x,d);
   for(i=0;i<n;i++)
   {s[i]=alpha*d[i];
    x[i]+=s[i];
   }
   c1=df.func(x);
   for(i=0;i<n;i++)
   {y[i]=c1[i]-c[i];}
   D=V_VT_y_s(y,y,s);
   E=V_VT_y_s(c,c,d);
   for(i=0;i<n;i++)
   {for(j=0;j<n;j++)
   {H[i][j]=H[i][j]+D[i][j]+E[i][j];
    H1[i][j]=H[i][j];
   }
   }
   for(i=0;i<n;i++)
   {c[i]=c1[i];
    c1[i]=-c1[i];
   }
   d=pivotlugauss(H1,c1);
   gamma=norm(c);
}
if(k >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}


public static double[] multiply(double[] left,double right)
{
//multiplying a vector with a constant
int i;
int n=left.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {
```

```
  b[i]=right*left[i];
  }
return b;
}


public static double[] divide(double[] left,double[][] right)
{
return pivotlugauss(right,left);
}

  //en kucuk kareler metodu
  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
              if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
                a[k][jj]=dummy;
              }
              dummy=b[p];
              b[p]=b[k];
              b[k]=dummy;
            }
            //gauss elemeyi çözme
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
          b[i]   =b[i]   -carpan*b[k];
            }
  }
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
```

```
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
}

public static double[][] add(double[][] left,double[][] right)
{
//addition of two matrices
int n1=left.length;
int m1=left[0].length;
int n2=right.length;
int m2=right[0].length;
int nMax,mMax;
int i,j;
if(m1>=m2) mMax=m1;
else       mMax=m2;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[][];
b=new double[nMax][mMax];
for(i=0;i<n1;i++)
  {
  for(j=0;j<m1;j++)
    {
    b[i][j]=left[i][j]+right[i][j];
```

```
    }
  }
return b;
//end of matrix addition method
}
public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }

            public static void main (String args[])
    {
            String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=verigir(s);
    f1 f_x=new f1();
    //df1 df_x=new df1();
    //double [] r1= BFGS(f_x,df_x,x0);
    double [] r1= BFGS(f_x,x0);
    s=" çözüm seti BFGS: \n"+Matrix.toStringT(r1);
    String s2="Broyden - Fletcher-Goldfarb-Shanno (BFGS) çok değişkenli fonksiyon optimizasyonu:";
     System.out.println(s2+s);
    }
}
```

Program çıktımız :

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO12I1
Broyden - Fletcher-Goldfarb-Shanno (BFGS) çok değişkenli fonksiyon optimizasyonu: çözüm seti BFGS:
        1.000000004775036000000000
        1.250000005968813000000000

> Terminated with exit code 0.
```

5.15 Çok bilinmiyenli-lineer olmayan-geometrik süreklilik metodu

Süreklilik metodunu kullanarak lineer olmıyan bir denklemin köklerini bulma işlemi detayları ile bölüm 4.13 de verilmişti. Newton metodundan hatırlayacağımız gibi optimizasyon problemi :
$f(x_1,x_2,x_3,\ldots,x_n)$ fonksiyon seti verildiğinde, birinci, ikinci türevler ve x değerleri iterasyon farkları

$$\nabla f = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{Bmatrix} \quad \text{ve} \quad \delta^{m+1} = \begin{Bmatrix} (x_1^{m+1} - x_1^m) \\ (x_2^{m+1} - x_2^m) \\ (x_3^{m+1} - x_3^m) \\ \cdots \\ (x_n^{m+1} - x_n^m) \end{Bmatrix} \text{ve} \quad [H] = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_{21} \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_3} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \frac{\partial^2 f}{\partial x_2 \partial x_n} & \frac{\partial^2 f}{\partial x_3 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

şeklinde verilebilir. (m+1) inci iterasyon için

$$\left[H^m\right]\{\delta^{m+1}\} = \{\nabla f^m\}$$

şeklinde yazılabilir. Veya

$\{\delta^{m+1}\} = \left[H^m\right]^{-1}\{\nabla f^m\}$ olarak ta yazılabilir. Burada $\left[H^m\right]^{-1}$ Hesian matrisinin ters (inverse) matrisidir. Newton-Raphson yönteminde bu matrisi iteratif olarak çözüyorduk. Burada ise bölüm 4.13 de gördüğümüz süreklilik(continuity) yöntemini kullanacağız. Bu yöntemi, Newton-Raphson yöntemi gibi bir yöntem için ilk değer tahmini sağlamak amacı ile de kullanabiliriz.

Program 5.15-1 Süreklilik(Continuity-homotopy) yöntemi kullanarak optimizasyon

```
 / FSCO5D Süreklilik(Continuation-homotopy) optimizasyon
// Metod Referansı : Numerical Analysis, 8th edition Richard L. Burden, J. Douglas Faires 3
// sayfa 637
// bu versiyon fonksiyonların türevini sayısal olarak almaktadır.
// kullanıcının türev fonksiyonu tanımlaması gerekmez


import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============== Tanımı gereken fonksiyon ================
abstract class fi_xi
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x) returns the value of f[0] and f[1]
  // as a two dimensional vector

  abstract double[] func(double x[]);
}

abstract class fij_xi
{
  // multifunction multi independent variable
  // for n independent variable n*n matrix of
  // dependent variables are returned
  // example f[0][0]=x[0]+Math.sin(x[1])   f[0][1]=x[0]-x[1]
```

```
  //        f[1][0]=x[0]*x[0]-x[1]       f[1][1]=Math.exp(x[0]+x[1]*x[1]
  // func(x) returns the value of f[0][0], f[0][1]
  //                              f[1][0], f[1][1]
  // as a two dimensional matrix
  abstract double[][] func(double x[]);
}

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
//ff=2.0*x[0]*x[1]+2.0*x[0]-x[0]*x[0]-2.0*x[1]*x[1];
ff=3.0*x[0]*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
return -ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

public class FSCO5D
{
// Süreklilik(Continuity-homotopy) metodu kullanarak denklem sisteminin köklerini bulmak
// Bu metod denklem sistemi çözümü için Matrix.java programını
// kullanmaktadır.


public static double[] multiply(double left,double[] right)
{
//multiplying a vector with a constant
int i;
int n=right.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {b[i]=left*right[i];}
return b;
}

public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right.length;
double[] b;
b=new double[m2];
 if(n1 != m2)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
   {
```

```
    b[ii]=0;
   }
   return b;
 }
 for(i=0;i<m1;i++)
   {
    b[i]=0;
    for(k=0;k<n1;k++)
          b[i]+=left[i][k]*right[k];
   }
return b;
//end of multiply of a matrix and a vector
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
   b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
   b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
}

public static double[][] inv(double[][] a)
{
// INVERSION OF A MATRIX
// inversion by using gaussian elimination
// with full pivoting
int n=a.length;
int m=a[0].length;
double b[][];
b=new double[n][n];
int indxc[];
int indxr[];
double ipiv[];
indxc=new int[n];
indxr=new int[n];
ipiv=new double[n];
```

```
int i,j,k,l,ll,ii,jj;
int icol=0;
int irow=0;
double big,dum,pivinv,temp;
if(n!=m)
{
   System.out.println("Matrix must be square ");
              for(ii=0;ii<n;ii++)
                for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
              return b;
}
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
      b[i][j]=a[i][j];
for(i=0;i<n;i++)
{
  big=0.0;
  for(j=0;j<n;j++)
  {
    if(ipiv[j] != 1)
      for(k=0;k<n;k++)
      {
            if(ipiv[k] == 0)
            {
              if(Math.abs(b[j][k]) >= big)
              {
                big=Math.abs(b[j][k]);
                irow=j;
                icol=k;
              }
            }
            else if(ipiv[k] > 1 )
              {
              System.out.println("error : inverse of the matrix : singular matrix-1");
              for(ii=0;ii<n;ii++)
                for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
              return b;
              }
      }
  }
  ++ ipiv[icol];
  if(irow != icol)
    for(l=0;l<n;l++)
    {
    temp=b[irow][l];
    b[irow][l]=b[icol][l];
    b[icol][l]=temp;
    }
  indxr[i]=irow;
  indxc[i]=icol;
```

```
  if(b[icol][icol] == 0.0)
  {
    System.out.println("error : inverse of the matrix : singular matrix-2");
              for(ii=0;ii<n;ii++)
                for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
    return b;
  }
  pivinv=1.0/b[icol][icol];
  b[icol][icol]=1.0;
  for(l=0;l<n;l++) b[icol][l] *=pivinv;
  for(ll=0;ll<n;ll++)
  if(ll != icol)
  {
    dum=b[ll][icol];
    b[ll][icol]=0.0;
    for(l=0;l<n;l++) b[ll][l]-= b[icol][l]*dum;
  }
}
for(l=n-1;l>=0;l--)
{
  if(indxr[l] != indxc[l])
     for(k=0;k<n;k++)
     {
       temp=b[k][indxc[l]];
       b[k][indxc[l]]=b[k][indxr[l]];
       b[k][indxr[l]]=temp;
     }
}
return b;
}

public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}

public static double[] df(f_xj f,double xxi[])
{
//Gredyen vektörü
// f(x0,x1,x2,...xn) founksiyonunun türev vektörü
// [df/dx0, df/dx1,df/dx2,....,df/dxn]
// df/dxj  j=0...x.length
//
// bu fonksiyon bilgisayar tarafından hesaplanmaktadır
// kullanıcı tarafından tanımlanan f(x) fonksiyonunun
// sayısal olarak alınan türevidir.

//
double a[]=new double[xxi.length];
for(int i=0;i<xxi.length;i++)
   {
   a[i]=dfdx(f,xxi,i);
   }
```

```
return a;
}


public static double dfdx(f_xj f,double xxi[],int xref)
{ int i;
            double h=1.0e-3;
  double hh=1.0/h;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
  for(i=0;i<xxi.length;i++)
  {
  if(i!=xref)
    {xp2h[i]=xxi[i];xph[i]=xxi[i];xm2h[i]=xxi[i];xmh[i]=xxi[i];}
  else {xp2h[i]=xxi[i]+2.0*h;xph[i]=xxi[i]+h;xm2h[i]=xxi[i]-2.0*h;xmh[i]=xxi[i]-h;}
  }
  fp2h=f.func(xp2h);
  fph=f.func(xph);
  fm2h=f.func(xm2h);
  fmh=f.func(xmh);
  return (-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
}
public static double d2fdx2(f_xj f,double xxi[],int xiref,int xjref)
{ //ikincitürev matrisi
  int i;
  double h=1.0e-4;
  double hh=1.0/h;
  int n=xxi.length;
  double ff;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
     for(int j=0;j<n;j++)
     {if(j!=xjref) {xp2h[j]=xxi[j];xph[j]=xxi[j];xm2h[j]=xxi[j];xmh[j]=xxi[j];}
      else {xp2h[j]=xxi[j]+2.0*h;xph[j]=xxi[j]+h;xm2h[j]=xxi[j]-2.0*h;xmh[j]=xxi[j]-h;}
     }
     fp2h=dfdx(f,xp2h,xiref);
     fph=dfdx(f,xph,xiref);
     fm2h=dfdx(f,xm2h,xiref);
     fmh=dfdx(f,xmh,xiref);
     ff=(-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
  return ff;
}

public static double[][]H(f_xj f,double x[])
{
  int n=x.length;
```

```
  double dy[][]=new double[n][n];;
  for(int ii=0;ii<n;ii++)
  {
    for(int jj=0;jj<n;jj++)
    {   //Hessian matrisi
       dy[ii][jj]=d2fdx2(f,x,ii,jj);
    }
  }
  return dy;
}
public static double[] continuationRK4(f_xj f,double x[],int N)
{
//=====================================================
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi J de tanımlanacaktır
// Homotopy, continuation (devamlılık) metodu ile lineer olmıyan denklem sistemi çözümü
// 4.derece Runge-Kutta denklemi ile
// yi+1 = yi + (1/6)*( k1 + 2k2 +2k3+k4)h
// k1=f(xi,yi)
// k2=f(xi+0.5h,yi+0.5k1h)
// k3=f(xi+0.5h,yi+0.5k2h)
// k4=f(xi+h,yi +k3h)
//=====================================================
// x  : bağımsız değişken vektörü
// y  : bağımlı değişken vektörü
// dy : bağımlı değişkenin türev fonksiyonları vektörü
// N  : Runge-Kutta diferansiyel denklem noktasayısı

int i;
int nmax=400;
double tolerance=1.0e-20;
int n=x.length;
double h=1.0/(double)N;
double b[]=new double[n];
double x1[]=new double[n];
double k[][]=new double[4][n];
double A[][]=new double[n][n];
b=multiply(-h,df(f,x));
for(i=0;i<N;i++)
{
x1=x;
A=H(f,x1);
k[0]=multiply(inv(A),b);
x1=add(x,multiply(0.5,k[0]));
A=H(f,x1);
k[1]=multiply(inv(A),b);
x1=add(x,multiply(0.5,k[1]));
A=H(f,x1);
k[2]=multiply(inv(A),b);
x1=add(x,k[2]);
A=H(f,x1);
```

```
k[3]=multiply(inv(A),b);
for(int j=0;j<n;j++) {x[j]=x[j]+1.0/6.0*(k[0][j]+2.0*k[1][j]+2.0*k[2][j]+k[3][j]);}
}
return x;
}

public static double[] continuationRK6(f_xj f,double x[],int N)
{
//=====================================================
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi J de tanımlanacaktır
// Homotopy, continuation (devamlılık) metodu ile lineer olmıyan denklem sistemi çözümü
// 6. derece Runge-Kutta denklemi ile
// yi+1 = yi + (1/90)*( 7k1 + 32k3 +12k4+32k5+7k6)h
// k1=f(xi,yi)
// k2=f(xi+0.25h , yi+0.25k1h)
// k3=f(xi+0.25h , yi+0.125k1h+0.125k2h)
// k4=f(xi+0.5h , yi - 0.5k2h+k3h)
// k5=f(xi+0.75h , yi + (3/16)k1h+(9/16)k4h)
// k6=f(xi+h , yi - (3/7)k1h+(2/7)k2h+(12/7)k3h - (12/7)k4h+(8/7)k5h)
//=====================================================
// x  : bağımsız değişken vektörü
// y  : bağımlı değişken vektörü
// dy : bağımlı değişkenin türev fonksiyonları vektörü
// N  : Runge-Kutta diferansiyel denklem noktasayısı
int i;
int nmax=400;
double tolerance=1.0e-20;
int n=x.length;
double h=1.0/(double)N;
double b[]=new double[n];
double x1[]=new double[n];
double k[][]=new double[6][n];
double A[][]=new double[n][n];
b=multiply(-h,df(f,x));
for(i=0;i<N;i++)
{
x1=x;
A=H(f,x1);
// k1=f(xi,yi)
k[0]=multiply(inv(A),b);
x1=add(x,multiply(0.25,k[0]));
A=H(f,x1);
// k2=f(xi+0.25h , yi+0.25k1h)
k[1]=multiply(inv(A),b);
x1=add(x,add(multiply(0.125,k[0]),multiply(0.125,k[1])));
A=H(f,x1);
// k3=f(xi+0.25h , yi+0.125k1h+0.125k2h)
k[2]=multiply(inv(A),b);
x1=add(x,add(multiply(-0.5,k[1]),k[2]));
A=H(f,x1);
```

```
// k4=f(xi+0.5h , yi - 0.5k2h+k3h)
k[3]=multiply(inv(A),b);
x1=add(x,add(multiply((3.0/16.0),k[0]),multiply((9.0/16.0),k[3])));
A=H(f,x1);
// k5=f(xi+0.75h , yi + (3/16)k1h+(9/16)k4h)
k[4]=multiply(inv(A),b);
x1=add(x,
      add(multiply((-3.0/7.0),k[0]),add(multiply((2.0/7.0),k[1]),
      add(multiply((12.0/7.0),k[2]),
         add(multiply((-12.0/7.0),k[3]),multiply((8.0/7.0),k[4])))));
A=H(f,x1);
// k6=f(xi+h , yi - (3/7)k1h+(2/7)k2h+(12/7)k3h - (12/7)k4h+(8/7)k5h)
k[5]=multiply(inv(A),b);
// yi+1 = yi + (1/90)*( 7k1 + 32k3 +12k4+32k5+7k6)h
for(int j=0;j<n;j++) {x[j]=x[j]+1.0/90.0*(7.0*k[0][j]+32.0*k[2][j]+12.0*k[3][j]+32.0*k[4][j]+7.0*k[5][j]);}
}
return x;
}

public static double[] newton_continuationRK4(f_xj f,double x[],int N)
{
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi dfj/dxi de tanımlanacaktır
// Newton-Raphson metodu ile lineer olmıyan denklem sistemi çözümü
//ti :önem katsayısı
//x bağımsız değişken vektörü
//y bağımlı değişken vektörü
//dy bağımlı değişkenin türev fonksiyonları vektörü
x=continuationRK4(f,x,4);
double ti=1.0;
int i;
int nmax=400;
double tolerance=1.0e-10;
int n=x.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
{
b[i]=1.0;
}
i=0;
while( i++ < nmax && Matrix.abs(b) > tolerance )
{ b=Matrix.multiply(Matrix.divide(df(f,x),H(f,x)),-ti);
  x=Matrix.add(x,b);
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
            " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI
UYARISI",JOptionPane.WARNING_MESSAGE);
System.out.println();
return x;
}
```

```
public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
   Double ax=new Double(token.nextToken());
   a[j++]=ax.doubleValue();
   }
   return a;
}

    public static void main (String args[])
    {
            f1 ff=new f1();
    double [] x0=verigir();
    double [] r= continuationRK4(ff,x0,4);
    double [] r1= continuationRK6(ff,x0,4);
    double [] r2=newton_continuationRK4(ff,x0,4);
    String s=" optimum süreklilik RK4 : \n"+Matrix.toStringT(r);
    s+=" optimum süreklilik RK6 : \n"+Matrix.toStringT(r1);
    s+=" optimum süreklilik newton-RK4 : \n"+Matrix.toStringT(r2);
    String s1="çok değişkenli süreklilik metodu ile optimizasyon : ";
     System.out.println(s1+s);
    System.exit(0);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO5D

çok değişkenli süreklilik metodu ile optimizasyon :  optimum süreklilik RK4 :
       0.99999999999998809000000000
       1.24999999999999098000000000
 optimum süreklilik RK6 :
       0.99999999999998809000000000
       1.24999999999999098000000000
 optimum süreklilik newton-RK4 :
       0.99999999999999178000000000
       1.24999999999998728000000000

> Terminated with exit code 0.
```

5.16 Çok bilinmiyenli-lineer olmayan-geometrik Sherman Morrison formüllü Broyden kiriş metodu ile optimizasyon

Bölüm 4.17 de Broyden formülünü türetmiştik. Bu formül temel olarak Kiriş (Secant) kök bulma yönteminin çok boyutlu bir uygulamasıdır. Bir optimizasyon problemi olarak f(x) fonksiyonunun minimumunu bulmak istiyorsak bu prosesi daha önce de Newton metodu ve diğer metodlarda birinci türev vektörünün köklerini bulma olarak tanımlamıştık. Problemimiz f(x)'in minimumunu bulma olsun, burada x={x_1, x_1,…, x_{n-1}, x_n} çok boyutlu değişken setidir. Bu durumda kiriş metodunu uygulayarak türev fonksiyonunu yaklaşık değerini oluşturabiliriz. Hatırlanacağı gibi, Newton-Raphson denkleminde f(x_1,x_2,x_3,…,x_n) fonksiyon seti verildiğinde, birinci, ikinci türevler ve x değerleri iterasyon farkları

$$g = \nabla f = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{Bmatrix} \text{ ve } \quad \delta^{m+1} = \begin{Bmatrix} (x_1^{m+1} - x_1^m) \\ (x_2^{m+1} - x_2^m) \\ (x_3^{m+1} - x_3^m) \\ \cdots \\ (x_n^{m+1} - x_n^m) \end{Bmatrix} \text{ ve}$$

$$[H] = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_{21} \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_3} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \frac{\partial^2 f}{\partial x_2 \partial x_n} & \frac{\partial^2 f}{\partial x_3 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} & \cdots & \frac{\partial g_3}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial g_n}{\partial x_1} & \frac{\partial g_n}{\partial x_2} & \frac{\partial g_n}{\partial x_3} & & \frac{\partial g_n}{\partial x_n} \end{bmatrix}$$

şeklinde verilebilir. (m+1) inci iterasyon için Newton-Raphson denklemi

$$\left[H^m\right]\{\delta^{m+1}\} = \{\nabla f^m\} = g^m$$

şeklinde verilebilir. Burada $[H]$ yerine yaklaşık türev formülünden yararlanmak istersek

$\frac{\partial g_j}{\partial x_k}(x^{(i)}) \approx \frac{g_j(x^{(i)} + e_k h) - g_j(x^{(i)})}{h}$ gibi bir formülle hesaplayabiliriz. Formüldeki h küçük bir sonlu değişim miktarı, e_k ise k inci bileşeni 1 diğer bileşenleri 0 olan bir vektördür. Buradaki (i) iterasyon sayısını belirtir. Kiriş yöntemi uygulandığında Newton-raphson formülünün kök civarındaki hızlı yakınsaması yerini daha yavaş bir yakınsama hızına bırakır. Bu yüzden burada standart kiriş yöntemi yerine yakınsama hızı birim hesaplama başına daha etkin olan Broyden yöntemini inceleyeceğiz. Broyden yönteminde $x^{(0)}$ ilk iterasyonu verilmiş ise $x^{(1)}$ iterasyonunu Newton metodunu kullanarak hesaplayabiliriz. Bundan sonraki stepler için şu yaklaşımı kullanırız :

$[H(x^{(0)})]\ \{x^{(1)} - x^{(0)}\} = -\{g(x^{(0)}\}$ denklemi

$[A_1]\ \{x^{(1)} - x^{(0)}\} = \{g(x^{(1)} - g(x^{(0)}\}$ şeklini alabilir. Bu denklemin geçerli olabilmesi için $[A_1]$ vektörünün her z vektörü için

$[A_1]\ \{z\} = [\nabla g(x^{(0)})]z \qquad \{x^{(1)} - x^{(0)}\}^t z = 0$ tanımlarını gerçekleştirmesi gerekir. Bu temel tanımdan $[A_1]$ formülünü oluşturabiliriz.

$$[A_1] = H(x^{(0)}) + \frac{[\{g(x^{(1)}) - g(x^{(0)})\} - H(x^{(0)})(x^{(1)} - x^{(0)})] * (x^{(1)} - x^{(0)})^t}{\left\| x^{(1)} - x^{(0)} \right\|_2^2}$$

buradaki $\left\| x^{(1)} - x^{(0)} \right\|_2^2$ ifadesi 2 normudur. Norm tanımları 3. bölümümüzde verilmişdi. $[A_1]$ değerini bulduktan sonra x vektörünün yeni değerini oluşturabiliriz.

$$x^{(2)} = x^{(1)} - A_1^{-1} g(x^{(1)})$$

$x^{(2)}$ bulunduktan sonra bu değeri aynı yolla $x^{(3)}$ ü hesaplamak için kullanırız. Genel olarak formülümüz

$$[A_i] = [A_{i-1}] + \frac{[\{g(x^{(1)}) - g(x^{(0)})\} - [A_{i-1}](x^{(1)} - x^{(0)})] * (x^{(1)} - x^{(0)})^t}{\left\| x^{(1)} - x^{(0)} \right\|_2^2}$$

$$x^{(i+1)} = x^{(i)} - A_i^{-1} g(x^{(i)})$$

formunu alır. Bu formülde her seferinde A matrisinin ters matrisini çözme zorunluluğumuz mevcuttur. Bu işlemi Gauss eleme yöntemi gibi bir yöntem kullanarak gerçekleştirebiliriz. Ancak Gauss yöntemide oldukça yüksek hesaplama zamanına ihtiyaç gösterir. Daha iyi bir alternatif iteratif Sherman-Morrison formülünü kullanmaktır. Sherman_Morrison formülü A_i^{-1} değerini direk olarak A_{i-1}^{-1} değerinden hesaplıyabilmemizi sağlar. Bu formülü şu stepleri kullanarak oluştururuz:

$$s_i = \{x^{(1)} - x^{(i-1)}\}$$

$$y_i = \{g(x^{(i)}) - g(x^{(i-1)})\}$$

$$(A + xy^t)^{-1} = A^{-1} - \frac{A^{-1} x y^t A^{-1}}{1 + y^t A^{-1} x}$$

$$A_i^{-1} = \left(A^{-1} + \frac{y_i - A_{i-1} s_i}{\left\| s_i \right\|_2^2} s_i^t \right)^{-1}$$

$$A_i^{-1} = A^{-1} + \frac{A_{i-1}^{-1} \left(\frac{y_i - A_{i-1} s_i}{\left\| s_i \right\|_2^2} s_i^t \right) A_{i-1}^{-1}}{1 + s_i^t A_{i-1}^{-1} \left(\frac{y_i - A_{i-1} s_i}{\left\| s_i \right\|_2^2} s_i^t \right)}$$

$$A_i^{-1} = A^{-1} + \frac{(s_i - A_{i-1}^{-1} y_i) s_i^t A_{i-1}^{-1}}{s_i^t A_{i-1}^{-1} y_i}$$

bu denklem sadece matris çarpımlarını kapsadığından toplam işlem olarak yapılması çok daha kolaydır.

Bu denklemleri kullanarak oluşturulmuş Broyden optimizasyon algoritması Program 5.15-1 de verilmiştir.

Program 5.16.1 Sherman-Morrison Formüllü Broyden kiriş optimizasyon algoritması

```
 // SCO5E Sherman-Morrison formülü destekli broyden sekant metoduyla
// optimizasyon
// Metod Referansı : Numerical Analysis, 8th edition Richard L. Burden, J. Douglas Faires 3
// sayfa 617

import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//============== Tanımı gereken fonksiyon ================

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
//ff=2.0*x[0]*x[1]+2.0*x[0]-x[0]*x[0]-2.0*x[1]*x[1];
ff=3.0*x[0]*x[0]-4.0*x[0]*x[1]+2.0*x[1]*x[1]-x[0]-x[1];
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

class fxb extends fi_xi
{
public double[] func(double x[])
{
//çözümü istenen fonksiyonun türev fonksiyonları
double ff[]=new double[2];
ff[0]=6.0*x[0]-4.0*x[1]-1.0;
ff[1]=-4.0*x[0]+4.0*x[1]-1.0;
return ff;
}
}

class fb extends fij_xi
{
fi_xi f;

public fb(fi_xi fi)
{f=fi;}

public double[][] func(double x[])
{
  int n=x.length;
  double a[][]=new double[n][n];
  for(int i=0;i<n;i++)
  {
```

```
    for(int j=0;j<n;j++)
    {
       a[i][j]=turev(x,i,j);
    }
  }
return a;
}

public double turev(double x[],int denklem_ref,int x_ref)
{
// fonksiyon f in denklem_ref sayılı fonksiyonunun(sayılar 0 dan başlar)
// x[x_ref} değişkenine göre türevi (sayılar 0 dan başlar)
// df_denklem_ref(x)/d_x_ref
// bu metod newtond metodu içinde kullanılmak içindir.
double h0=0.0256;
int i,m;
int n=7;
double f1[];
f1=new double[x.length];
double f2[];
f2=new double[x.length];
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
x2[i]=x[i];
}
//derivative of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
```

```
x2[x_ref]-=h[i];
f1=f.func(x1);
f2=f.func(x2);
T[i][0]=( f1[denklem_ref] - f2[denklem_ref])/(2.0*h[i]);
x1[x_ref]=x[x_ref];
x2[x_ref]=x[x_ref];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}

}

public class FSCO5E1
{
// SCO5E Sherman-Morrison formülü destekli broyden sekant metoduyla
//denklem sistemlerinin köklerini (çözümünü) bulmak

public static double[] multiply(double left,double[] right)
{
//multiplying a vector with a constant
int i;
int n=right.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {b[i]=left*right[i];}
return b;
}

public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right.length;
double[] b;
b=new double[m2];
 if(n1 != m2)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
   {
```

```
    b[ii]=0;
   }
   return b;
 }
 for(i=0;i<m1;i++)
   {
    b[i]=0;
    for(k=0;k<n1;k++)
          b[i]+=left[i][k]*right[k];
   }
return b;
//end of multiply of a matrix and a vector
}

public static double[] multiply(double[] left,double[][] right)
{
//multiplication of one vector with one matrix
int ii,jj,i,j,k;
int m2=right[0].length;
int n2=right.length;
int m1=left.length;
double[] b;
b=new double[m1];
 if(n2 != m1)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n2;ii++)
   {
    b[ii]=0;
   }
   return b;
 }
 for(i=0;i<m2;i++)
   {
    b[i]=0;
    for(k=0;k<m1;k++)
          b[i]+=right[i][k]*left[k];
   }
return b;
//end of multiply of a vector and a matrix
}

public static double  VT_X(double [] left,double [] right)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*right[i];
  }
return tot;
```

```
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
}

public static double[][] multiply(double left,double[][] right)
{
//multiplying a matrix with a constant
int i,j;
int n=right.length;
int m=right[0].length;
double b[][];
b=new double[n][m];
for(i=0;i<n;i++)
  {
  for(j=0;j<m;j++)
    b[i][j]=right[i][j]*left;
  }
return b;
//end of multiplying a matrix with a constant double
}

public static double[][] add(double[][] left,double[][] right)
{
//addition of two matrices
int n1=left.length;
int m1=left[0].length;
int n2=right.length;
int m2=right[0].length;
int nMax,mMax;
int i,j;
if(m1>=m2) mMax=m1;
```

```
else       mMax=m2;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[][];
b=new double[nMax][mMax];
for(i=0;i<n1;i++)
  {
  for(j=0;j<m1;j++)
    {
    b[i][j]=b[i][j]+left[i][j];
    }
  }
for(i=0;i<n2;i++)
  {
  for(j=0;j<m2;j++)
    {
    b[i][j]=b[i][j]+right[i][j];
    }
  }
return b;
//end of matrix addition method
}

public static double[] substract(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]-right[i];
  }
return b;
//end of vector addition method
}

public static double[][] inv(double[][] a)
{
// INVERSION OF A MATRIX
// inversion by using gaussian elimination
// with full pivoting
int n=a.length;
```

```
int m=a[0].length;
double b[][];
b=new double[n][n];
int indxc[];
int indxr[];
double ipiv[];
indxc=new int[n];
indxr=new int[n];
ipiv=new double[n];
int i,j,k,l,ll,ii,jj;
int icol=0;
int irow=0;
double big,dum,pivinv,temp;
if(n!=m)
{
   System.out.println("Matrix must be square ");
              for(ii=0;ii<n;ii++)
                for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
              return b;
}
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
       b[i][j]=a[i][j];
for(i=0;i<n;i++)
{
  big=0.0;
  for(j=0;j<n;j++)
  {
    if(ipiv[j] != 1)
      for(k=0;k<n;k++)
      {
            if(ipiv[k] == 0)
            {
              if(Math.abs(b[j][k]) >= big)
              {
                big=Math.abs(b[j][k]);
                irow=j;
                icol=k;
              }
            }
            else if(ipiv[k] > 1 )
              {
              System.out.println("error : inverse of the matrix : singular matrix-1");
              for(ii=0;ii<n;ii++)
                for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
              return b;
              }
      }
  }
  ++ ipiv[icol];
```

```
  if(irow != icol)
    for(l=0;l<n;l++)
    {
    temp=b[irow][l];
    b[irow][l]=b[icol][l];
    b[icol][l]=temp;
    }
  indxr[i]=irow;
  indxc[i]=icol;
  if(b[icol][icol] == 0.0)
  {
    System.out.println("error : inverse of the matrix : singular matrix-2");
              for(ii=0;ii<n;ii++)
                for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
    return b;
  }
  pivinv=1.0/b[icol][icol];
  b[icol][icol]=1.0;
  for(l=0;l<n;l++) b[icol][l] *=pivinv;
  for(ll=0;ll<n;ll++)
  if(ll != icol)
  {
    dum=b[ll][icol];
    b[ll][icol]=0.0;
    for(l=0;l<n;l++) b[ll][l]-= b[icol][l]*dum;
  }
}
for(l=n-1;l>=0;l--)
{
  if(indxr[l] != indxc[l])
     for(k=0;k<n;k++)
     {
       temp=b[k][indxc[l]];
       b[k][indxc[l]]=b[k][indxr[l]];
       b[k][indxr[l]]=temp;
     }
}
return b;
}
public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}

public static double[][] multiply(double[] left,double[] right)
{
//multiplication of one vector with one matrix
```

```
int ii,jj,i,j,k;
int m2=left.length;
int n2=right.length;
int m1=left.length;
double[][] b;
b=new double[m2][n2];
 for(i=0;i<m2;i++)
   {
     for(k=0;k<n2;k++)
               b[i][k]=left[i]*right[k];
   }
return b;
//end of multiply of a vector and a matrix
}

public static double[] broyden(fi_xi f,double x[])
{
//=====================================================
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi J de tanımlanacaktır
// Sherman-Morrison formülü uygulanmış broyden (kiriş tipi) metodu
// ile lineer olmıyan denklem sistemi çözümü
fb J=new fb(f);
int i;
int nmax=400;
double tolerance=1.0e-15;
int n=x.length;
double b[]=new double[n];
double v[]=new double[n];
double s[]=new double[n];
double w[]=new double[n];
double y[]=new double[n];
double z[]=new double[n];
double p;
double ut[]=new double[n];
double x1[]=new double[n];
double A[][]=new double[n][n];
x1=x;
A=J.func(x1);
v=f.func(x1);
A=inv(A);
s=multiply(-1.0,multiply(A,v));
x=add(x,s);
int k=1;
while(k<nmax)
{
w=v;
v=f.func(x);
y=substract(v,w);
z=multiply(-1.0,multiply(A,y));
p=-VT_X(s,z);
```

```
ut=multiply(s,A);  //*********
A=add(A,multiply((1.0/p),multiply(add(s,z),ut)));
s=multiply(-1.0,multiply(A,v));
x=add(x,s);
if(norm(s)<tolerance) break;
k++;
}
return x;
}

public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
   Double ax=new Double(token.nextToken());
   a[j++]=ax.doubleValue();
   }
   return a;
}

    public static void main (String args[])
    {
            fa f =new fa();
            fxb f1=new fxb();
    double [] x0=verigir();
    double [] r= broyden(f1,x0);
    String s=" kök değerleri süreklilik broyden : \n"+Matrix.toStringT(r);
    String s1="çok değişkenli broyden metodu ile kök bulma : ";
      System.out.println(s1+s);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO5E1
çok değişkenli broyden metodu ile kök bulma :  kök değerleri süreklilik broyden :
      1.000000000000000000000000000
      1.250000000000000000000000000

> Terminated with exit code 0.
```

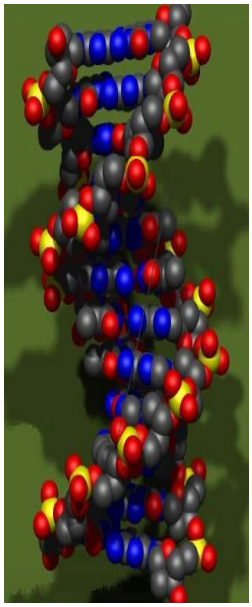
5.17 Global tesadüfi çözüm metodları : Genetik Algoritmalar

Genetik algoritmalar evrim programlaması denilen ve günümüzde suni zeka alanında oldukça yoğun kullanılan bir programlama tekniğidir. Adından da anlaşılacağı gibi temelini evrim kuramından almıştır. Temel olarak problemlerin

evrim prosesinin en iyilerin hayatta kalması prensibine dayanır. Genetik algoritmalar canlı yapıların Darwin teorisinde belirtildiği gibi doğaya uyum mekanizması içinde kendi kendilerini genlerdeki değişmeyle değiştirmelerinin fonksiyonlara optimizasyon problemi olarak uyarlanmış halidir. Darwin teorisi anne ve babaların özelliklerinin çocuklara geçtiğini, bazen de anne ve babada olmayan yeni bazı özelliklerin mutasyon denilen mekanizmayla oluştuğunu, bu özellikler türün yaşama şansını arttırıyorsa bu bireylerin çoğalarak bu özellikleri kendi çocuklarına geçirdiği, özellikler başarısızsa ölecekleri için başarısız özellikleri gelecek nesillere geçiremeyecekleri böylece ancak doğaya uyumda başarılı olan özelliklerin daha sonraki nesillerde birikmesi sonucu doğaya uyum sağlayan türlerin oluşabileceğini getirmiştir. Daha sonra Genetik biliminin gelişmesiyle bu prensibin altındaki biokimyasal yapılar (genler) bulunmuş ve bu sistemin matematiksel ve istatistiksel olarak nasıl çalıştığı daha detaylı anlaşılmıştır. Genler 4 amino asitin yapısı içinde sıralanması ile canlı yapıların özelliklerinin anahtarını oluştururlar. Bu özellikler, anne ve babadan gelen genlerle çocuklara aktarıldığı gibi bu aktarma prosesinde zaman zaman oluşabilen hatalarda darwinin tanımladığı mutasyon prosesini oluşturmaktadır.

Evrimsel hesaplama ilk defa 1960'lı yıllarda I Rechenberg'in "*Evolutionsstrategie* in original " evrim statejileri çalışmasında yer almıştır. Fikir daha sonra başka araştırıcılarca geliştirilmiştir. Genetik algoritmalar John Holland ve öğrencileri tarafından ilk defa bu günkü anlamıyla oluşturulmuştur. 1975 yılında basılan "*Adaption in Natural and Artificial Systems*" kitabında yazar bu tekniğin nasıl kullanılacağını açıklamıştır

5.17.1 Genel kavramlar



(bolum5_015.jpg, Resimaltı:DNA yapısı)

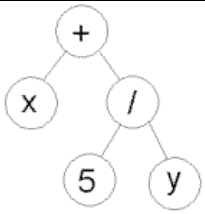
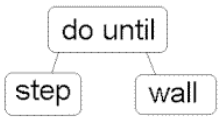
Tüm canlı yapılar hücreler içerir. Her hücre bir veya daha fazla kromozom seti barındırır. Kromozomlar DNA dizinleridir. Ve tüm organizmanın modelini içlerinde barındırırlar. Kromozom DNA'lardan oluşan gen bloklarını barındırır. Her genin belli bir özelliğin kodunu taşıdığı söylenebilir, örneğin göz rengi gibi. Bu özelliklerin alabildiği değerlere de gen değerleri adı verilir(örneğin kahverengi göz, yeşil göz). Kromozomların hepsi bir arada canlının karekterlerini oluşturur, buna cenom denir. Cenomdaki belli bir özelliği oluşturan guruba cenotip ismi verilir. Canlıların doğumdan sonra çevresel etkiler yoluyle oluşan özeliklerine de fenotip ismi verilir.
Üreme esnasında önce rekombinasyon oluşur, anne ve babaları genleri bölünerek birleşirler ve yeni bir gen yapısı oluştururlar. Yeni oluşan genlerin bir kısmı mutasyona da uğrayabilir, mutasyon gen DNA dizilimindeki anne ve babadan genlerin kopyelenmesi sırasında oluşan hatalardır, bunlar yeni bir dizilim oluşmasına sebep olurlar. Doğal Uyum organizmanın hayatta kalma başarısı ile ölçülen ve gen guruplarının direk olarak etkilediği bir olaydır.

Genetik algoritmalar yukarda değindiğimiz mekanizmanın yeni nesiller üretmekte ve yeni nesillerin doğaya uyum mekanizmasını açıklayan Darwin teorisinden yola çıkarak oluşturulmuştur.
Algoritma kromozomlardan oluşan bir gurup çözüm setinden oluşur. Bir gurubun oluşturduğu çözüm setlerinden problemin karekteristiğine göre seçilen çözüm setleri üreme yolu ile yeni nesillere taşınır. Başarılı bireylerin daha fazla üreme olasılığı olacağından genlerinin bir sonraki nesle geçme olasılığı daha yüksektir. Bireylerde aynı zamanda mutasyon yoluyla tamamen yeni özellikler oluşabilir, bu özellikleri bir kısmı bu bireylerin başarısını azaltırken bazen de daha başarılı yeni bireyler de oluşabilir.
Genetik algoritmaların temel yapıları şu şekilde özetlenebilir :

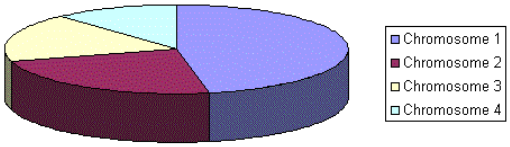
[BAŞLA] Programa göre n kromozondan oluşan bir set oluştur
[UYUM] x genlerinden oluşan her kromozonun *f(x)* uyum değerini *tüm nüfus için hesapla*
[YENİ NESİL] Alttaki stepler uyarınca yeni bir nesil oluştur
[SEÇİM] Toplam nüfustan uyumları göz önünde bulundurularak bir anne baba çifti seçin, bu seçimde uyum daha fazla ise seçim olasılıkları da buna göre fazlalaşsın.

[ÜREME] Üreme yoluyla yeni bir nesil üretip anne babalarıyla bu yeni nesli değiştirn. Eğer üreme olmadı ise eski nesli koruyun.
[MUTASYON] belli bir olasılık içinde mutasyonları oluşturun
[KABUL] Yeni bireyleri nüfusa ekleyin
[DEĞİŞTİRME] yeni nesli aynı şekilde tekrar yeni nesiller elde etmek için kullanın
[TEST] eğer istenilen şartlar oluşmuşsa simulasyonu durdurun
[DÖNGÜ] Oluşmamışsa geri dönerek işleme devam edin.

Gen veya kromozon genetik hesaplamada kullanılan temel elemanlardır. Matematiksel olarak düşünürsek n değişkenli bir sistemde her değişkeni bir gen olarak düşünürüz. Tüm değişken setini bütün olarak kromozon olarak adlandırabiliriz. Kromozonların yapıları

kromozom A **kromozon B** **kromozon C**	**5.3243 0.4556 2.3293 2.4545** **ABDJEIFJDHDIERJFDLDFLFEGT** **(geri), (geri), (sağ), (ileri), (sol)**
kromozom D	 (bolum5_016.jpg,Resimaltı:) (+ x (/ 5 y))
kromozom E	 (bolum5_017.jpg,Resimaltı:) (do_until step wall)

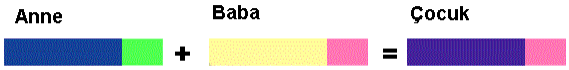
Gibi birbirinden çok farklı eleman tipleri kullanılarak oluşturulabilir. Doğadaki gerçek genlerde uyum dediğimizde doğada hayatta kalabilme yeteneğini anlıyoruz. Matematiksel olarak optimizasyon(maksimizasyon) için bu kavramı kullanacak olursak burada fonksiyonun değeridir. Daha sonra bu uyum değerleri toplanarak yüzde şekline çevrilebilir.



(bolum5_018.jpg,Resimaltı:)
Bu yüzdeye çevrilmiş uyum değer fonksiyonlarının arasından tesadüfi olarak seçim yaparsak,
tesadüfi seçimde sayıların toplanmış yüzde olasılığı daha iyi uyum gösteren (maksimum fonksiyon değerini alan) birey için daha yüksektir. Uyum gösteren bireyler arasından tesadüfi olarak seçilen kısmının bir sonraki nesli üretmesine(çoğalmasına) izin verilir. Çoğalma prosesi anne ve baba diyebileceğimiz iki bireyin genlerinin birleşerek yeni bir birey (çocuk) oluşturmasıdır. Matematiksel olarak buna çaprazlama prosesi denir. Çaprazlama anne ve baba genlerinin bir kısmının (tesadüfi olarak) seçilerek bir araya getirilmesi ile gerçekleşir.
Örneğin binari rakamlardan oluşan bir gen için : Eğer tek çaprazlama metod kullanırsak, bu metodda tek bir tesadüfi sayı gen gurubunu belli bir oranda böler. Aynı oranda bölünen genin bir kısmı anneden diğer kısmı da babadan alınır. Ve yeni bireyi oluşturur. Genlerin bölme oranı tesadüfi olarak belirlenir.

Örneğin 8 bitten oluşan bir gurup için:
8*random(0-1) = 5 ise ilk 5 bitti anneden, son 3 biti babadan alarak yeni bireyi oluştururuz.

11001011 + 11011111 = 11001111



(bolum5_019.jpg,Resimaltı:)

Kromozom 1 (Anne)	11011 | 00100110110
Kromozom 2 (Baba)	11011 | 11000011110
1. Çocuk	11011 | 11000011110
2. Çocuk	11011 | 00100110110

Eğer çift çaprazlama uygulamak istersek, bu metodda iki tesadüfi sayı gen gurubunu belli bir oranda üç parçaya böler. Aynı oranda bölünen genin bir kısmı anneden diğer kısmı da babadan sonra kısım tekrar anneden alınır (veya tersi yapılır). Ve yeni bireyi oluşturur. Genlerin bölme oranı tesadüfi olarak belirlenir.
Örneğin 8 bitten oluşan bir gurup için:
Random1=8*random(0-1) = 2
Random2=8*random(0-1) = 6
ise ilk 2 biti anneden, 3 den 6ıncı bite kadar babadan ve 7 ve 8 inci biti tekrar biti babadan alarak yeni bireyi oluştururuz.

11001011 + 11011111 = 11011111

(bolum5_020.jpg,Resimaltı:)
Eğer düzenli çaprazlama uyulamak istersek, bu metodda her gen tesadüfi olarak anneden veya babadan seçilerek yeni birey(çocuk) oluşturulur.
Örneğin 8 bitten oluşan bir gurup için:
Random1=random(0-1) 0.5 den küçük ise 1 inci gen olarak annenin geni, 0.5 den büyük ise babanın geni seçilir.
11001011 + 11011111 = 11001011

(bolum5_021.jpg,Resimaltı:)

5.17.2 MÜHENDİSLİK OPTİMİZASYONU

Biz burada sizlere bu metodun bir mühendislik aracı olarak optimizasyon (minimizasyon – maksimizasyon) problemlerine uygulanmasını ve yöntemin temel çalışma prensibini açıklamaya çalışacağız.

Optimizasyon (Minimizasyon-Maksimizasyon) problemi
Matematiksel olarak
f(x1,x2,x3,…..xi) fonksiyonunun maksimumunu bulma olarak tanımlanabilir.
Bu işlemi oluştururken önce maksimum ve minimum işleminin aynı işlem olduğunu unutmıyalım

max [f(x)] = min [- f(x)] = min [g(x)]

Fonksiyon oluşturmada göz önüne alınması gereken diğer bir konu da fonksiyon değerine sabit bir değer ilave ettiğimizde maximum değerin yerinin değişmiyeceğidir.

max [f(x)] = max [f(x) + C] = max [g(x)]

Bu özelliği kullanarak fonksiyonumuzun çözümünün her zaman pozitif bölgeye düşmesini sağlıyabiliriz.

Mühendislik optimizasyonunda çözdüğümüz sayıların çoğu gerçek sayılar olmak zorundadır. Ancak bu gerçek bir sayıyı bir binari sayılar dizini olarak ifade etmek her zaman mümkündür. Programi optimizasyon problemi olarak düşündüğümüzde bir gurup bağımsız değişkenimiz mevcuttur
(x0,x1,x2,..xi..xn)
Genetik algoritmaları uygulamak için her bağımsız değişkenin sınır değerlerinin (maksimum ve minumum) bilinmesi zorunludur. Bu yüzden değişken gurubumuz için
xo : xo_minimum xo_maximum
x1 : x1_minimum x1_maximum
….
xi : xi_minimum xi_maximum
…
xn : xn_minimum xn_maximum
Tanımları yapılır.
Her bağımsız değişken için ayrıca değişkenin binari eşdeğerinin kaç bir olacağının tanımlanması gerekir. Bağımsız değişkenimiz
xi_min <= xi <= xi_max N bit binary
Olarak tanımlanmış ise :
Binary eşdeğeri veya binari olarak verilmiş sayının gerçek sayı eşdeğeri iki adımlık bir prosesle hesaplanabilir.
N digitlik bir binary gurubu tamsayıya

$$(b_n, b_{N-1}, b_{N-2}, \ldots b_1, b_0)_2 = (\sum_{i=0}^{N-1} b_i 2^i)_{10} = x'$$

Denklemi ile dönüştürülebilir. x' 10 tabanından bir tamsayıdır. Buna karşı gelen gerçek sayı ise

$$x = x\_min + x'\left(\frac{x\_max - x\_min}{2^N - 1}\right)$$

İfadesiyle hesaplanır. Aynı işlem ters olarak yapılarak da gerçek sayıdan binari sayıya ulaşılabilir

Bağımsız değişkenler her biri bir gen olmak için NDEĞİŞKEN kapsayan bir gurupta genlerin tek bir uzun binari dizide toplanması ile veya geni ihtiva eden sınıfın dizini olarak ifade edilebilir. Dizinlerden elde edilen bağımsız değişkenler optimizasyonu yapılacak fonksiyonda girdi olarak kullanılarak fonksiyonun değeri hesaplanır. Programlama yapılırken değişik fonksiyonlar girilebilecek şekilde yapılırsa kodlama açısından daha genel bir kod elde edebilirsiniz.

```
class fa extends f_xj
{
 double func (double x[])
 { return (21.5+x[0]*Math.sin(4.0*Math.PI*x[0])+x[1]*Math.sin(20.0*Math.PI*x[1]));
 }
}
```

Bağımsız değişken ve fonksiyonun bu bağımsız değişken için aldığı değer bütün olarak bir cenotip oluşturur. Bir veya daha fazla cenotip kromozomlar olarak bir araya gelebilir. (bir veya birden fazla fonksiyon bir arada değerlendirilebilir). Kromozon sayılarını da birden fazla düşünebiliriz. Bu sistem hep birlikte toplumumuzun bireylerini oluşturur (Optimize edilecek denklem sistemi ve bağımsız değişkenler sınır değerleri)

Programlayıcı veya program kullanıcısı tarafından toplumuzun kaç bireyden oluşacağı önceden belirlenir. Bu sayıya eşdeğer birey bağımsız değişken değerleri tesadüfi sayılardan oluşan(sayını verilen limit değerleri içinde) bir set olarak oluşturulur. Fonksiyon değerleri de her cenotip için hesaplanır. (Mühendislik optimizasyon problemlerinde genelde çok

değişkenli bir fonksiyon mevcuttur). Bu değerler genin uyum değerleridir. Tüm toplum bireylerinin uyum değerleri toplanarak toplam uyum hesaplanır. Sonra uyum değerleri toplam uyuma bölünerek göreceli uyum değerleri elde edilir. Göreceli uyum değerleri de birinci bireyden itibaren her birey için toplanarak toplanmış göreceli (kümülatif) uyum değerleri saptanır. Listedeki ilk bireyin toplanmış göreceli uyum değeri göreceli uyum değerine eşittir. Son bireyin ise bir'e eşittir.

Toplanmış göreceli uyum değerleri olasılıkları, uyumun büyüklüğüne göre oluşmuştur. Ve 0 ile 1 arasında yer alır. Eğer listemizden tesadüfi rakamlar aracılığı ile yeni bir nesil için seçme yapacak olursak tesadüfi rakamlar göreceli olarak eşit dağılacağından, daha yüksek uyuma sahip bireylerin seçilme şansı daha yüksektir. Aynı bireyin birden fazla kere seçilmesine de müsaade edilir.

Seçilen yeni toplum bireyleri arasında kullanıcılar tarafından belirlenen bir yüzde içerisinde kalan kısmının mutasyon prosesine girmesine izin verilir. Kullanıcı toplumdaki mutasyon yüzdesini tanımlar. Yine tesadüfi sayılar oluşturarak bu sayı mutasyon yüzdesinin altında ise o bireying genlerinin tesadüfi seçilen birisinin mutasyona uğraması (binari sistemde seçilen bitin değerinin değiştirilmesi sağlanır(1 ise 0, 0 ise 1 değerini alır)

Toplum yeniden değerlendirilmeye girer ve uyum, göreceli uyum değerleri saptanır, en iyi uyum sağlıyan birey seçilir. Bu bireyin değeri ayrı bir yerde tutularak tüm proses boyunca takip edilebilir. Eğer istenirse her nesilde en başarılı olan birey bir sonraki nesillere aktarılabilir.

5.17.3 Bilgisayar bit yapısı kullanarak geni oluşturan : gene sınıfı , genetype1 sınıfır ve genetıc1 sınıfı üzerinden ilk genetik algoritma modeli

GENE SINIFI

Genetik algoritmalar doğadaki bu prosesi taklit ederler. Gen yapısını taklit etmek için bizim kurduğumuz modelde bilgisayar bitleri kullanılmıştır. Gene sınıfı gerekli prosesleri bit olarak yapar, eşdeğer olarak double değerini de hesaplayabiliriz, double-bit değişkenini kendi isteğimize göre yapabiliriz. Değişkenin bit uzunluğu da isteğe göre değişken olarak verilmiştir. Gene sınıfı değişkeni maksimum 64 bit (0-63) olmak üzere ayarlanabilir. Gene sınıfı içersinde çaprazlama ve mutasyon yapma metodlarını da barındırır. Mutasyon için seçilen 1 tesadüfi gen bitinin değeri değiştirilir(1 ise 0 0 ise 1 yapılır). Çaprazlama olarak yukarda belirtilen 3 çeşit çaprazlama mümkündür (düzenli çarprazlama, tek çaprazlama, çift çaprazlama). Sınıf double değerden veya "00110100111" gibi bir bit String değerinden de okuma yapabilir.

Program 5.17.3-1 Gene sınıfı

```
İmport java.util.*;
import java.io.*;

public class Gene extends Object
{
//Gene class generates digital(binary) array to represent a double number
//actual binary defination changes depends on number of bits, xmin and xmax
//binary data stored in b BitSet array
int N;     //number of binary digits(number of bits in a byte)
       //N should be less than 64
double xN;//Limit value for integer(2^N-1)
double xmin,xmax;//maximum and minimum limits of double number
BitSet b; //binary array that strore the integer

public Gene(double ixmin,double ixmax,int iN)
{//randomly generate the gene
 N=iN;
 if(N>63) N=63;
 b=new BitSet(N);
```

```
 xN=Math.pow(2.0,N)-1;
 xmin=ixmin;
 xmax=ixmax;
 //generate random numbers to fill binary array
 setbits();
}

public Gene(String s,double ixmin,double ixmax)
{
//generate gene equal to a given string
 N=s.length();
 b=new BitSet(N);
 xN=Math.pow(2.0,N)-1;
 xmin=ixmin;
 xmax=ixmax;
 for(int i=N-1;i>=0;i--)
    if(s.charAt(i)=='1') b.set(N-1-i);
}

public Gene(double number,double ixmin,double ixmax,int iN)
{
//Generate a gene from a given double number
 N=iN;//number of binary digits(number of bits in a byte)
 if(N>63) N=63;
 b=new BitSet(N);
 xN=StrictMath.pow(2.0,N)-1;
 xmin=ixmin;
 xmax=ixmax;
 //z : integer equivalent of the double number
long z=(long)((number-xmin)/(xmax-xmin)*xN);
//System.out.println("inside double"+z);
long y=(long)xN;
double r=z;
for(int i=0;i<N;i++)
   {
   if((z-(z/2)*2)!=0) {b.set(i);}
   z/=2;
   }
}

public Gene()
{
//empty Gene
N=1;
b=new BitSet(N);
xN=StrictMath.pow(2.0,N)-1;
xmin=0;
xmax=1.0;
b.set(0);
}

public Gene(Gene g)
```

```
{
// copy a Gene from another one
copyGene(g);
}

public void copyGene(Gene g)
{
 N=g.N;
 b=(BitSet)g.b.clone();
 xN=Math.pow(2.0,N)-1;
 xmin=g.xmin;
 xmax=g.xmax;

}

public Gene copyGene()
{
 Gene g=new Gene(this);
 return g;
}

public Gene(Gene Father,Gene Mother)
{
//crosovertype will be selected randomly
double ix=Math.random();
if(ix<0.1) setGene(Father,Mother,"regular");
else if(ix>0.9) setGene(Father,Mother,"single");
else setGene(Father,Mother,"double");
}


public void  setGene(Gene Father,Gene Mother,String crossovertype)
{
xmin=Father.xmin;
xmax=Father.xmax;
N=Father.N;
xN=Father.xN;
b=new BitSet(N);
int ir1,ir2,ir3;
if(crossovertype.equals("regular"))
{
for(int i=0;i<N;i++)
   {if(boolean_random()) { if(Father.b.get(i)) b.set(i);}
    else { if(Mother.b.get(i)) b.set(i);}
   }
}
else if(crossovertype.equals("single"))
{
    ir1=int_random();
    if(boolean_random())
    {   for(int i=0;i<ir1;i++)
          { if(Father.b.get(i)) b.set(i);}
```

```
      for(int i=ir1;i<N;i++)
        { if(Mother.b.get(i)) b.set(i);}
   }
   else
   {   for(int i=0;i<ir1;i++)
        { if(Mother.b.get(i)) b.set(i);}
      for(int i=ir1;i<N;i++)
        { if(Father.b.get(i)) b.set(i);}
   }
}
else if(crossovertype.equals("double"))
{
   ir1=int_random();
   ir2=int_random();
   int ix;
   if(ir1>ir2) {ix=ir2;ir2=ir1;ir1=ix;}
   if(boolean_random())
   {   for(int i=0;i<ir1;i++)
        { if(Father.b.get(i)) b.set(i);}
      for(int i=ir1;i<ir2;i++)
        { if(Mother.b.get(i)) b.set(i);}
      for(int i=ir2;i<N;i++)
        { if(Father.b.get(i)) b.set(i);}
   }
   else
   {   for(int i=0;i<ir1;i++)
        { if(Mother.b.get(i)) b.set(i);}
      for(int i=ir1;i<ir2;i++)
        { if(Father.b.get(i)) b.set(i);}
      for(int i=ir2;i<N;i++)
        { if(Mother.b.get(i)) b.set(i);}
   }
}
}


public void cross1(double crossratio,Gene Father,Gene Mother)
{// create gene from two parent genes
 // crossover process
//  Genes from Father and Mother should have the same limit values
xmin=Father.xmin;
xmax=Father.xmax;
N=Father.N;
xN=Father.xN;
b=new BitSet(N);
int ir1=(int)(crossratio*N+0.5);
for(int i=0;i<ir1;i++)
  { if(Father.b.get(i)) b.set(i);}
for(int i=ir1;i<N;i++)
  { if(Mother.b.get(i)) b.set(i);}
}
```

```
public void cross2(double crossratio,Gene Father,Gene Mother)
{// create gene from two parent genes
 // crossover process
//  Genes from Father and Mother should have the same limit values
xmin=Father.xmin;
xmax=Father.xmax;
N=Father.N;
xN=Father.xN;
b=new BitSet(N);
int ir1=(int)(crossratio*N+0.5);
for(int i=0;i<ir1;i++)
   { if(Mother.b.get(i)) b.set(i);}
for(int i=ir1;i<N;i++)
   { if(Father.b.get(i)) b.set(i);}
}

public long getXP()
{
//returns intermediate integer of double number equivalent
long y=1;
double r=xN;
long z=0;
for(int i=0;i<N;i++)
   {
   if(b.get(i)) {z+=y;}
   y*=2;
   }
 return z;
}

public double getX()
{//return double number equivalent of bit set number
 return (xmin+(xmax-xmin)/xN*getXP());
}

public void clear()
{//fill the entire population with zero
b.clear(0,N);}

public void fill()
{//fill the entire population with ones
b.clear(0,N);b.flip(0,N);}

public String toBitSetString()
{//return bit set array values as defined in class BitSet
return b.toString();
}

public String toString()
{
// return bit set array values as a binary number definition
// plus double value of array
```

```
 String s="(";
 for(int i=N-1;i>=0;i--)
    if(b.get(i)) s+="1";
    else         s+="0";
    s+=") "+getX();
  return s;
}

public boolean boolean_random()
{
if(Math.random()<=0.5) return false;
else return true;
}

public int int_random()
{
  int x1=(int)(Math.random()*N);
  return x1;
}

public double double_random()
{
  return (xmin+Math.random()*(xmax-xmin));
}

public void setbits()
{
b.clear();
for(int i=0;i<N;i++)
   {
   if(boolean_random()) b.set(i);
   }
}

public void flip(int i)
{
b.flip(i);
}

public void mutate()
{
  //fip one bit of data
  int i1=int_random();
  if(i1<0) i1=0;
  b.flip(i1);
}
}
```

Gene sınıfının biraz daha açılabilmesi için çeşitli örnekler verilmiştir. İlk örneğimizde 63 bitlik 5 adet Gene tipi değişken oluşturuyoruz. Değişken değerleri gelişigüzel yüklenmektedir.

Program 5.17.3-2 geneTest sınıfı ile çeşitli 63 bitlik Gen'ler oluşturma (sayı 0 ile 10 arası)

```
public class geneTest
{
 public static void main(String arg[])
 {
 Gene x1=new Gene(0,10,63);
 for(int i=0;i<5;i++)
 { x1.setbits();System.out.println(x1.toString());}
 }
}
```

```
(0101010001000010001110000000010000011011000110111001011111111011) 3.291354188942094
(0110110000111110001011000010000001101001110110011010111011111110) 4.228236750730961
(1110000100011111100101101100010100000101110111001011111000001011) 8.793882590306223
(1010000100011101011110100011101000010001101010100100000110111100) 6.293560401697187
(0000000111111111100000010000000011001010000111101011111111101111) 0.078049303939407
```

Program 5.17.3-3 geneTest sınıfı ile çeşitli 63 bitlik Gen oluşturarak 3.2 değerini yükleme

```
public class geneTest
{ public static void main(String arg[])
 { Gene x1=new Gene(3.2,0,10,63);System.out.println(x1.toString());}
}
```

```
 (0101000111101011100001010001111010111000010100011111011000000000) 3.2
```

Program 5.17.3-4 geneTest sınıfı ile çeşitli 63 bitlik Gen oluşturarak String'den değerini yükleme

```
public class geneTest
{ public static void main(String arg[])
 {Gene x1=new Gene("0101000111101011100001010001111010111000010100011111011000000000",0,10);
   System.out.println(x1.toString());}
}
```

```
 (0101000111101011100001010001111010111000010100011111011000000000) 3.2
```

Genotype1 sınıfı

Genotype1 sınıfı gen bit-double değişken türünün bir anlamda boyutlu değişkeni gibi düşünülebilir. Genetik anlamda bakıldığında kromozona eşdeğerdir. Matematik olarak düşünüldüğünde ise n boyutlu $f(x_1,x_2,x_3,\ldots,x_n)$ fonksiyonun tüm değişkenlerine karşı gelen bir toplam değişken setidir. Fonksiyon uyum değerlerini ve kümülatif uyum değerlerini hesaplar.

Genetıc1 sınıfı

Genetic algoritmanın ana sınıfı Genetic1 sınıfıdır. Bu sınıfta genetik optimizasyon gerçekleştirilmektedir. Burada bazı değişkenler kullanıcı tarafından verilmelidir. Bu değişkenler :

```
int    POPSIZE; //populasyonu oluşturan birey sayısı
int    MAXGENS; // nesil sayısı(maksimum iterasyon?);
int    NVARS;   //değişken sayısı
int    N;       //genlerin bit eşdeğer boyutu-değişken boyutu
double PXOVER; //üreme-çaprazlama olasılığı;
double PMUTATION; //mutasyon olasılığı
```

şeklindedir. Genetic sınıfı tüm Genetik optimizasyon proseslerini barındırır. Bu işlemlerden evaluate bireylerin uyumlarını (fonksiyon değerlerini) hesaplar. Genetik algoritmalarda fonksiyon değerlendirmesinde önemli olan bir konuda tüm fonksiyon değerlerinin pozitif olma gerekliliğidir. – olarak değerlendirilen fonksiyonlar sabit bir değer

eklenerek artı olarak değerlendirilecek fonksiyonlara dönüştürülebilir. Bu proses fonksiyonun optimum noktasını değiştirmez.

$f(x_1,x_2,x_3,\ldots,x_n)$ fonksiyonunun maksimumu $g(x_1,x_2,x_3,\ldots,x_n) = f(x_1,x_2,x_3,\ldots,x_n) + C$ fonksiyonunun maksimumuyla aynıdır.

Bu proses kümülatif uyumu hesaplarken bir hata olmasını önler.
Evaluate prosesi bireylerin uyum değerlerini ve kümülatif uyum değerlerini hesaplar
Select metodu kümülatif uyumdaki başarılı bireyler arasından yeni nesli üretme hakkına sahip olacakları tesadüfi proses kullanarak seçer.
Crossover prosesi select prosesi tarafından seçilen bireylerden yeni bireyler üretir.
Mutate prosesi verilen belli bir yüzdedeki populasyonu mutasyona uğramasını sağlar
Calculate prosesi tüm üstteki prosesleri sırayla ve toplam nesil sayısına göre itere ederek toplam nesil sayısı sonundaki nufusu(populasyonu) elde eder.

Program 5.17.3-5 genotype1 sınıfı ve Genetic1 sınıfı

```
İmport java.io.*;
import java.text.*;
import java.util.Locale;

abstract class f_xj
{
  // single function  multi independent variable
  // a single value is returned indiced to equation_ref
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x,1) returns the value of f[1]
  // func(x,0) returns the value of f[0]
  abstract double func(double x[]);
}
class genotype1
{
// each genotype1 is member of the population
// Genotype is like a chromosone in real life genetic systems
// it can be thought as an evaluation point of a function with n variable
// f(xi)=f(x1,x2,x3,..,x_nvars)
public Gene G[]      ;  // string of variables makes a genotype1
public double Fitness;  // genotype1's fitness f(xi)
public double Upper[];  // gene's upper bound; xi_lower < xi < xi_upper
public double Lower[];  // Gene's lower bound; xi_lower < xi < xi_upper
public double RFitness; // the relative fitness
public double CFitness; // cumulative fitness
public int NVARS;       // Number of variables(Genes)
public int N;           // Number of bits defining each gene
public boolean Breeder;

  public genotype1(int iNVARS,double low[],double high[],int iN,f_xj fi)
  {setgenotype1(iNVARS,low,high,iN,fi);}
  public void setgenotype1(int iNVARS,double low[],double high[],int iN,f_xj fi)
  {
  setN(iN);
  setNVARS(iNVARS);
```

```
G=new Gene[NVARS];
Upper=new double[NVARS];
Lower=new double[NVARS];
setLower(low);
setUpper(high);
setGene();
setFitness(fi);
}
public genotype1(genotype1 gi,f_xj fi)
{setgenotype1(gi,fi);}

public void setgenotype1(genotype1 gi,f_xj fi)
{
setN(gi.N);
setNVARS(gi.NVARS);
G=new Gene[NVARS];
Upper=new double[NVARS];
Lower=new double[NVARS];
setLower(gi.Upper);
setUpper(gi.Lower);
for(int i=0;i<NVARS;i++)
  G[i]=gi.G[i].copyGene();
setFitness(fi);
}
public genotype1 copygenotype(f_xj fi)
{
genotype1 g1=new genotype1(this,fi);
return g1;
}
public void setN(int iN)    {N=iN;}
public void setNVARS(int iNVARS)    {NVARS=iNVARS;}
public void setLower(double lbound[]) {for(int i=0;i<NVARS;i++) Lower[i]=lbound[i];}
public void setUpper(double ubound[]) {for(int i=0;i<NVARS;i++) Upper[i]=ubound[i];}
public void setFitness(f_xj fi)
{          Fitness=fi.func(getGene());}
public void setRFitness(double isum)  {RFitness=Fitness/isum;}
public void setCFitness(double ix)  {CFitness=ix;}
public double getLower(int j) {return Lower[j];}
public double getUpper(int j) {return Upper[j];}
public double getFitness() {return Fitness;}
public double getRFitness(){return RFitness;}
public double getCFitness(){return CFitness;}
public void breed() {Breeder=true;}
public void not_breed() {Breeder=false;}
public boolean isBreed() {return Breeder;}

public void setGene(int val)
{  G[val]=new Gene(Lower[val],Upper[val],N);}

public void setGene()
{  for(int j=0;j<NVARS;j++)
              setGene(j);
```

```
  }
  public double getGene(int val)
  { return G[val].getX();}
  public double[] getGene()
  {   double x[]=new double[NVARS];
              for(int j=0;j<NVARS;j++)
                 x[j]=getGene(j);
              return x;
  }
  public String toString()
  {
  //x values, Fitness,relative fitness and cumulative fitness
  String s="";
  for(int j=0;j<NVARS;j++)
              s+=G[j].toString()+" F:";
  s+=getFitness();
  s+="RF:"+getRFitness();
  s+="CF:"+getCFitness();
  return s;
  }
}  //end of class genotype1

public class Genetic1
{
            // Maximizing a function
            int    POPSIZE; //population size
            int    MAXGENS; //maximum number of generations;
            int    NVARS;   //number of independent variables of function
    int    N;       //number of genes in each variable
            double PXOVER; //probability of crossover;
    double PMUTATION; //proportion of mutated variables
    int Generation; //Current generation number
    int NBreeder; //Number of survivors in a generation
    int Best; //The best genotype1 in the population
    genotype1 Population[];
    //f_xj f;  // function to be avaluated by genetic algorithm

    public Genetic1(int iPOPSIZE,int iMAXGENS,int iNVARS,int iN,
    double iPXOVER,double iPMUTATION)
    {
    POPSIZE=iPOPSIZE;
    MAXGENS=iMAXGENS;
    NVARS=iNVARS;
    N=iN;
    PXOVER=iPXOVER;
    PMUTATION=iPMUTATION;
    NBreeder=0;
    Population=new genotype1[POPSIZE+1];
            }
            public void setPopulation(double low[],double up[],f_xj fi)
            {
            for(int i=0;i<POPSIZE+1;i++)
```

```
  { Population[i]=new genotype1(NVARS,low,up,N,fi);}
}

public genotype1[] copyPopulation(f_xj fi)
{
genotype1 Po[]=new genotype1[POPSIZE+1];
for(int i=0;i<POPSIZE+1;i++)
  { Po[i]=new genotype1(Population[i],fi);
    Po[i].Fitness=Population[i].Fitness;
            Po[i].RFitness=Population[i].RFitness;
    Po[i].CFitness=Population[i].CFitness;
  }
return Po;
}

public void evaluate(f_xj fi)
{
  int mem;
  int j;
  double x[]=new double[NVARS];
  for(mem=0;mem<POPSIZE;mem++)
  {
  //System.out.println(Population[mem].toString());
  Population[mem].setFitness(fi);
  if(Population[mem].getFitness() > Population[POPSIZE].getFitness())
     {
            Best=mem;
            Population[POPSIZE]=Population[mem];
     for(j=0;j<NVARS;j++)
       {
       Population[POPSIZE].G[j].copyGene(Population[mem].G[j]);
              }//end of for(j=0..
            }//end of if
  }//end of for(mem=0;..
}


public void setRFCF(genotype1 Pi[])
{
//calculates relative and cumulative fitness functions
int mem;
int PN=Pi.length;
double sum=0.0;
//total fitness of population
for(mem=0;mem<PN;mem++)
  {sum+=Pi[mem].Fitness;}
//System.out.println("sum="+sum);
//calculate relative fitness of each genotype1
for(mem=0;mem<PN;mem++)
  {Pi[mem].setRFitness(sum);}
//calculate cumulative fitness
Pi[0].setCFitness(Pi[0].getRFitness());
```

```
for(mem=1;mem<PN;mem++)
          {Pi[mem].setCFitness(Pi[mem-1].getCFitness()+Pi[mem].getRFitness());
          }
         }

         public void toString(genotype1 Pi[])
         {
         int mem;
         int PN=Pi.length;
         for(mem=0;mem<PN;mem++)
         {
         //list them
         System.out.println("Population["+mem+"]="+Pi[mem].toString());
         }
         }

         public void select(f_xj fi)
         {
         //select the new generation members of population
         double r;
         int mem;
         setRFCF(Population);
         //create a new population;
         genotype1 Po[]=copyPopulation(fi);
 setRFCF(Po);
 for(int i=0;i<POPSIZE;i++)
 {   mem=0;
    r=Math.random();
    double gf1=Population[i].getCFitness();
            double gf2;
    if (gf1 > r) Po[i]=Population[mem].copygenotype(fi);
            for(mem=1;mem<POPSIZE;mem++)
      {
              gf2=Population[mem].getCFitness();
              if( gf2>=r && gf1< r ) {Po[i]=Population[mem].copygenotype(fi);break;}
      }
 }
 setRFCF(Po);
         Population=Po;
         evaluate(fi);
 //toString(Population);
 }

public void crossover(f_xj fi)
{
int i,count;
int mem;
int POne,a;
int PTwo,b;
int point;
//select members for breeding
int iselect[]=new int[POPSIZE];
```

```
int counter=0;
double r;
 for(mem=0;mem<POPSIZE;mem++)
         {
          r=Math.random();
          if(r < PXOVER)
            {
                  iselect[counter++]=mem;
            Population[mem].breed();
                  NBreeder++;
            }
          else
            Population[mem].not_breed();
         }
         //System.out.println("iselect="+Matrix.toString(iselect));
         //let also best of the population to breed
         Population[Best].breed();

//loop through the population select breeding pairs
 for(mem=0;mem<POPSIZE;mem++)
    {
      //select two in popolation in random
      a=(int)(Math.random()*NBreeder)+1;
      b=(int)(Math.random()*NBreeder)+1;
      count=0;
      POne=0;
      PTwo=0;
      //select two individuals for breeding
      for(i=0;i<POPSIZE;i++)
        {
        if(Population[i].isBreed())
          {
          count++;
          if(count==a) POne=count;
          if(count==b) PTwo=count;
                }
              }
            //perform a crossover;
              genotype1 Po[]=new genotype1[2];
                    Po[0]=Population[POne].copygenotype(fi);
                    Po[1]=Population[PTwo].copygenotype(fi);
              for(i=0;i<NVARS;i++)
              {
                    Population[POne].G[i].cross1(Math.random(),Po[0].G[i],Po[1].G[i]);
                    Population[PTwo].G[i].cross1(Math.random(),Po[0].G[i],Po[1].G[i]);
                    }
                    Population[POne].setFitness(fi);
                    Population[PTwo].setFitness(fi);
              }
            }

public void mutate(f_xj fi)
```

```
  {
  int i;
  double lbound,hbound;
  int nmutations;
  int member;
  int var;
  nmutations=(int)((double)(POPSIZE*NVARS*PMUTATION*N));
  for(i=0;i<nmutations;i++)
    {
    member=(int)(Math.random()*POPSIZE);
    var=(int)(Math.random()*NVARS);
    //replace the old value with a new mutated one
    Population[member].G[var].mutate();
          //after mutation recalculate the function value
    Population[member].setFitness(fi);
    }
  }

  public String report()
  {
  String s;
  int i;
  double best_value;
  double avg;
  double stdev;
  double sum_square;
  double square_sum;
  double sum;
  double xx;
  sum=0;
  sum_square=0.0;
  for(i=0;i<POPSIZE;i++)
    {
          xx= Population[i].getFitness();
    sum+=xx;
    sum_square=xx*xx;
          }
  avg=sum/(double)POPSIZE;
  square_sum=sum*sum/(double)POPSIZE;
  stdev=Math.sqrt(square_sum);
  best_value=Population[POPSIZE].getFitness();
  double aa[]=new double[NVARS+1];
  for(i=0;i < NVARS;i++)
   {
   aa[i]=Population[POPSIZE].getGene(i);
          }
          aa[NVARS]=Population[POPSIZE].getFitness();

  //s="Generation = "+Generation+"best value = "+Matrix.toString(aa)+"Average = "+avg+"Standart Deviation
= "+stdev;
  s="Generation = "+Generation+"best value = "+Matrix.toString(aa);
  return s;
```

```
  }

  public double[] getBest()
  {
  double aa[]=new double[NVARS+1];
  for(int i=0;i < NVARS;i++)
   {
   aa[i]=Population[POPSIZE].getGene(i);
            }
            aa[NVARS]=Population[POPSIZE].getFitness();
  return aa;
  }

  public double[] calculate(f_xj fi,boolean report)
  {
  evaluate(fi);
  Generation=0;
  while(Generation<MAXGENS)
   {
   Generation++;
   NBreeder=(int)(0.8*POPSIZE);
   select(fi);
   crossover(fi);
   mutate(fi);
   evaluate(fi);
   //if(report)
     //System.out.println(report());
   }
  return getBest();
  }
}
```

İlk örnek fonksiyon olarak $f(x_0,x_1)=2x_0\ x_1+2x_0-x_0^2-2x_1^2+10$ fonksiyonuna bakalım. bu fonksiyon örneğin **Program 5.10-1** da kullanılan $f(x_0,x_1)=2x_0\ x_1+2x_0-x_0^2-2x_1^2$ fonksiyonun aynısıdır, ancak burada artı değeri her zaman sağlayabilmek için fonksiyonumuza 10 değeri ilave edilmiştir. Bu tür bir fonsiyonda geometrik metodlarında bize rahatlıkla çözüm verebildiğini anımsayalım.

Program 5.17.3-6 OPO17 örnek problem sınıfı (iki bilinmyenli fonksiyon)

```
İmport java.io.*;
class fa extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=2.0*x[0]*x[1]+2.0*x[0]-x[0]*x[0]-2.0*x[1]*x[1]+10.0;
return ff; //maksimum testi
}
}
class OPO17
{
   public static void main(String args[]) throws IOException
```

```
    {
    int N=18;
            int POPSIZE=200;
    int MAXGENS=200;
    int NVARS=2;
    double PXOVER=0.3;
    double PMUTATION=0.02;
    fa fx=new fa();
    Genetic1 xx=new Genetic1(POPSIZE,MAXGENS,NVARS,N,PXOVER,PMUTATION);
    double low[]=new double[NVARS];
    double high[]=new double[NVARS];
    low[0]=-1.0;high[0]=5.0;
    low[1]=-1.0;high[1]=5.0;
    xx.setPopulation(low,high,fx);
    System.out.println("best=\n"+Matrix.toStringT(xx.calculate(fx,false)));
    }
}
```

Çıktı 5.17.3-5 OPO17 örnek problem sınıfı

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" OPO17
best=
       2.000240326844509
       1.000251770979961
      11.999999936480407

> Terminated with exit code 0.
```

İkinci örnek fonksiyon olarak $f(x_0)=x_0 \sin(10\pi x_1)+1$ fonksiyonuna bakalım. Bu fonksiyonu -1 ile 2 arasındaki maksimum değerini bulmak istiyoruz. Fonksiyonu değerlendirecek olur isek değerleri -1 ile 3 arasında değiştiğini görürüz. Dönüşümü garanti altına alabilmek için fonksiyona 2 değeri eklersek $f(x_0)=x_0 \sin(10\pi x_1)+3$ şekline dönüşür. Bu fonksiyonun geometrik metodlarla çözülmesi oldukça zordur. **Çıktı 5.17.3-7** de fonksiyonun grafiğini görmekteyiz. Buradan da görebileceğiniz gibi bu fonksiyonda bir çok yerel maksimum bulunmaktadır. Bu yüzden de geometrk algoritmalarımız bu yerel maksimumlarda takılarak bize global çözüm sağlamadan duracaklardır.

Program 5.17.3-7 OPO17A örnek problem sınıfı (bir bilinmeyenli fonksiyon)

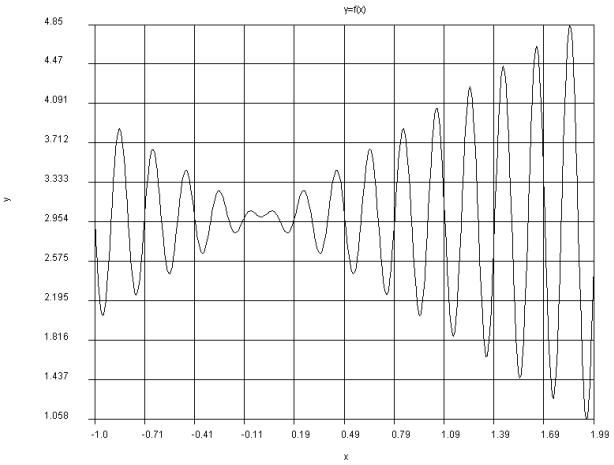
```
import java.io.*;

class fa extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=x[0]*Math.sin(10.0*Math.PI*x[0])+3.0;
return ff; //maksimum testi
}
}
class OPO17A
{
   public static void main(String args[]) throws IOException
```

```
    {
    int N=18;
            int POPSIZE=200;
    int MAXGENS=200;
    int NVARS=1;
    double PXOVER=0.3;
    double PMUTATION=0.02;
    fa fx=new fa();
    Genetic1 xx=new Genetic1(POPSIZE,MAXGENS,NVARS,N,PXOVER,PMUTATION);
    double low[]=new double[NVARS];
    double high[]=new double[NVARS];
    low[0]=-1.0;high[0]=2.0;
    xx.setPopulation(low,high,fx);
    System.out.println("best=\n"+Matrix.toStringT(xx.calculate(fx,false)));
    }
}
```

Çıktı 5.17.3-6 OPO17A örnek problem sınıfı (bir bilinmeyenli fonksiyon)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" OPO17A
best=
      1.850596811663863
      4.850271542089701
> Terminated with exit code 0.
```

Çıktı 5.17.3-7 OPO17A örnek problem sınıfı grafik gösterimi (bir bilinmeyenli fonksiyon)



(bolum5_022.jpg,Resimaltı: OPO17A örnek problem sınıfı grafik gösterimi bir bilinmeyenli fonksiyon)

Bu fonksiyonu geometrik bir arama yönteminde kullanamaya çalıştığımızda başarılı olamadığımızı görebiliriz. Son olarak iki boyutlu kompleks geometriye sahip bir fonksiyonu inceleyelim :

Program 5.17.3-8 OPO17B örnek problem sınıfı (iki bilinmeyenli fonksiyon)

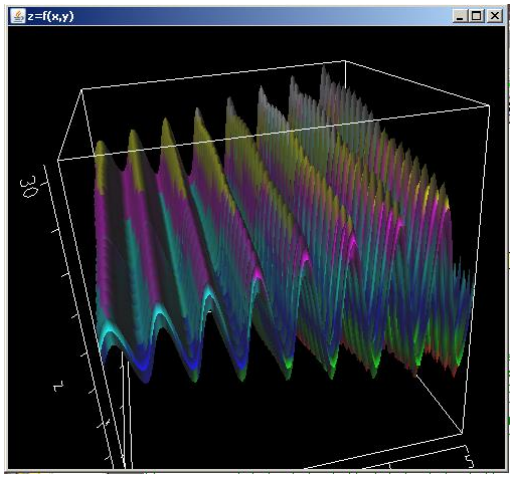
```
İmport java.io.*;


class fa extends f_xj
{
public double func(double x[])
{
```

```
//çözümü istenen fonksiyon
double ff;
ff=21.5+x[0]*Math.sin(4.0*Math.PI*x[0])+x[1]*Math.sin(20.0*Math.PI*x[1]);
return ff; //maksimum testi
}
}

class OPO17B
{
   public static void main(String args[]) throws IOException
   {
   int N=18;
            int POPSIZE=200;
   int MAXGENS=200;
   int NVARS=2;
   double PXOVER=0.3;
   double PMUTATION=0.02;
   fa fx=new fa();
   Genetic1 xx=new Genetic1(POPSIZE,MAXGENS,NVARS,N,PXOVER,PMUTATION);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=-3.0;high[0]=12.1;
   low[1]=4.1;high[1]=5.8;
   xx.setPopulation(low,high,fx);
   System.out.println("best=\n"+Matrix.toStringT(xx.calculate(fx,false)));
   }
}
```

Çıktı 5.15.3-8 OPO17B örnek problem sınıfı (iki bilinmeyenli fonksiyon)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" OPO17B
best=
      11.625012683916792
       5.800000000000000
      38.925010694354235
> Terminated with exit code 0.
```

Çıktı 5.17.3-9 OPO17B örnek problem sınıfı (iki bilinmeyenli fonksiyon) üç boyutlu grafik çıktısı



(bolum5_023.jpg,Resimaltı: OPO17B örnek problem sınıfı grafik gösterimi iki bilinmeyenli fonksiyon)

Bu fonksiyonun da geometrik optimizasyon metodları ile çözümü zor olabilir. Bitirirken son bir noktaya tekrar dikkatinizi çekmek istiyoruz. Genetik algoritmaların bulduğu değerler istatistik proses sonucu bulunduğundan her zaman aynı değeri vermeyecektir. Örneğin yukarıdaki problem için çıktıyı tekrar aldığımda :

Çıktı 5.17.3-10 OPO17B örnek problem sınıfı (iki bilinmeyenli fonksiyon)

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" OPO17B
best=
      11.624436662432336
       5.800000000000000
      38.920512521726340
> Terminated with exit code 0.
```

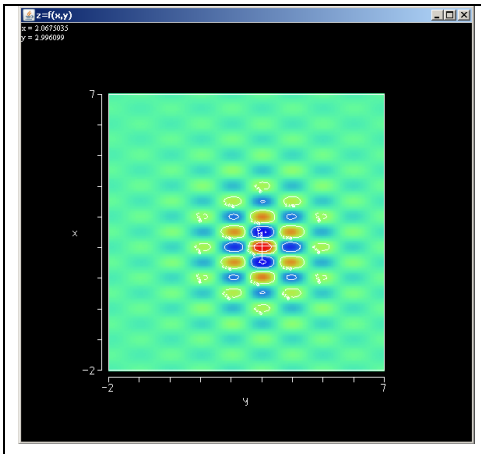
Görüldüğü gibi sonuçlar tam olarak aynı değildir. Genetik algoritmaların veya genelde istatistik yaklaşımlı metodların geometrik yöntemlerden en önemli farklarından birisi de budur. Geometrik yöntemler aynı şartlar verildiğinde hep aynı sonuca ulaşırlar. Geometrik formüllerle çözmesi zor olabilecek bir fonksiyon daha verelim:

Program 5.17.3-9 OPO19C örnek problem sınıfı (iki bilinmeyenli fonksiyon)

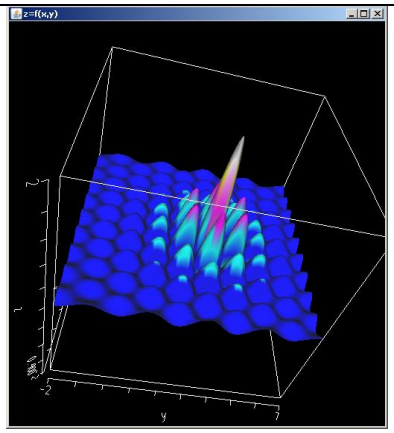
```
İmport java.io.*;
import javax.swing.*;



class fa extends f_xj
{
//çözümü istenen fonksiyon
 double func(double x[])
 {
   double ff= 1.0+Math.cos(Math.PI*(x[0]-3.0))*Math.cos(2.0*Math.PI*(x[1]-2.0))/
   (1+(x[0]-3.0)*(x[0]-3.0)+(x[1]-2.0)*(x[1]-2.0));
   return ff; //maksimum testi;
  }
}

class fb extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
fa fx=new fa();
return -fx.func(x); //maksimum testi
}
}

class OPO19C
{
   public static void main(String args[]) throws IOException
   {
   int N=63;
            int POPSIZE=200;
   int MAXGENS=200;
```

```
    int NVARS=2;
    double PXOVER=0.3;
    double PMUTATION=0.02;
    fa fax=new fa();
    fb fbx=new fb();
    Genetic1 xx=new Genetic1(POPSIZE,MAXGENS,NVARS,N,PXOVER,PMUTATION);
    double low[]=new double[NVARS];
    double high[]=new double[NVARS];
    low[0]=-10.0;high[0]=10.0;
    low[1]=-10.0;high[1]=10.0;
    xx.setPopulation(low,high,fax);
    double a[]=xx.calculate(fax,false);
    double x0[]=new double[NVARS];
    for(int i=0;i<NVARS;i++) x0[i]=a[i];
    String s="genetik algoritma sonucu =\n"+Matrix.toStringT(a);
     String s2="Genetik algoritma  çok değişkenli fonksiyon optimizasyonu:";
     JOptionPane.showMessageDialog(null,s,s2,JOptionPane.PLAIN_MESSAGE);
    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO19C
Genetik algoritma  çok değişkenli fonksiyon optimizasyonu:genetik algoritma sonucu =
        3.000488281266138000000000
        1.999511718740514300000000
        1.999993640432403700000000


> Terminated with exit code 0.
```



(bolum5_024.jpg,Resimaltı: OPO19C örnek problem sınıfı grafik gösterimi bir bilinmeyenli fonksiyon)



(bolum5_025.jpg,Resimaltı: OPO19C örnek problem sınıfı grafik gösterimi bir bilinmeyenli fonksiyon)

5.17.4 Genetik algoritmanın gauss tesadüfi sayılı gen yapısı ile oluşturulması

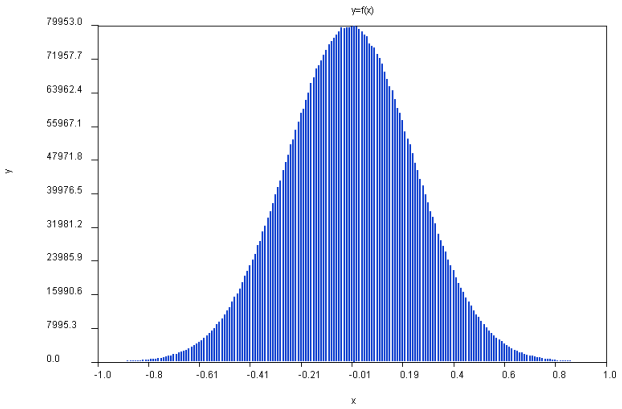
Üstte verdiğimiz genetik algoritma modelinde binary bit modelinden yararlanarak temel gen yapımızı oluşturmuştuk. Şimdi aynı temel gen kavramını farklı bir şekilde oluşturmaya çalışacağız. Bu versiyonumuzda temel olarak gauss dağılımlı tesadüfi rakam kavramından yararlanacağız. Şu ana kadar eşit dağılımlı tesadüfi rakam kavramından sıkça

yararlandık. Gauss tesadüfi rakamları verilen aralıkta her nokta için eşit ağırlıkta tesadüfi sayılar yerine verilen bir ortalama değer civarında daha fazla kümeleşen çan eğrisi şeklinde dağılım gösteren tesadüfi sayılar üretir. Gauss tesadüfi dağılımını oluşturan program ve deneme programı P. 5.16.4-1 de verilmiştir. Oluşturulan 500000 rakam için dağılım grafiğide çıktı 5.16.4-2 de verilmiştir. tesadügi sayının bellibirnoktada yoğunlaşması anne ve baba bireylerinden çocuk bireyi oluştururken anne ve baba bireylerin özelliklerinin çocuklara geçirilmesi yönünde önemlidir. Böylece oluşan yenitesadüfi sayı üretildiği bireyin özelliklerini belli bir oranda korur, ancak aynı olmak veya her zaman çok yakın olmak zorunda da değildir.

Program 5.17.4-1 Gauss dağılımlı tesadüfi rakam kontrolu 500000 rakam oluşturulduğunda dağılım grafiği

```
import java.io.*; //java girdi cikti sinifini cagir
class randeneme
{
public static double  gaussrandom(double xort,double r)
{ int iset=0;
 double gset=0;
 double fac,rsq,v1,v2;
 double aret=0;
 if(iset==0)
 {
  do{
            v1=2.0*Math.random()-1.0;
            v2=2.0*Math.random()-1.0;
            rsq=v1*v1+v2*v2;
    }while(rsq>=1.0 || rsq==0.0);
  fac=Math.sqrt(-2.0*Math.log(rsq)/rsq);
  gset=v1*fac;
  iset=1;
  aret=v2*fac;
 }
 else
 {
 iset=0;
 aret=gset;
 }
 return aret/8.0*r+xort;
 }
public static void main(String args[])
 {
  double xort,r;
  double gr=0;
  double x[]=new double[201];
  double y[]=new double[201];
  xort=0.0;
  r=2.0;
  for(int i=0;i<=200;i++) {x[i]=xort-r/2+r*(double)i/200.0;}
  for(int i=0;i<=5000000;i++)
    {gr=gaussrandom(xort,r);for(int j=0;j<200;j++){if(gr>=x[j] && gr< x[j+1]){y[j]+=1.0;}}}
  Plot pp=new Plot(x,y);
  pp.setColor(0,0,51,204);
  pp.setPlotType(0,41);
  pp.plot();
}
```

```
}
```



(bolum5_026.jpg,Resimaltı **:**gaussrandom metodunun 5000000 atışta oluşturduğu sonuç)

Gen sınıfı

Bu tesadüfi sayı sistemini kullanarak Gen sınıfı geliştirlmiştir. Gen sınıfı işlevsel olarak, yukarıda oluşturulmuş olan Gene sınıfı ile aynı işlevi görür. Temelfark Gene sınıfında mutasyon çaprazlama işlemleri bitler üzerinden gerçekleştirilirken, burada gauss tesedüfi sayıları üzerinden gerçekleştirilmektedir.

Program 5.17.4-2 Gen sınıfı Gauss tesadüfi sayı üzerinden temel gen işlemlerini(mutasyon, çaprazlama) hesaplar

```
// Dr. M. Turhan ÇOBAN
// EGE University School of Engineering, Mechanical Engineering Department
// package : Numerical analysis package
// subject : optimization
// group   : Genetic Algorithms gauss random metodu kullanarak.
// Gen Class
// Gen class generates a double number represent a gene

import java.util.*;
import java.io.*;

public class Gen extends Object
{
public double xmin,xmax;//maximum and minimum limits of double number
public double x;

public static double  gaussrandom(double ixmin,double ixmax,double ix)
{
  double d1=ixmax-ix;
  double d2=ix-ixmin;
  double d=Math.abs(Math.min(d1,d2));
  ix=gaussrandom(ix,d);
  return ix;
}

public static void  gaussrandom()
{
  double d1=xmax-x;
  double d2=x-xmin;
  double d=Math.abs(Math.min(d1,d2));
  x=gaussrandom(x,d);
```

```
}

public static double  gaussrandom(double xort,double r)
{ //r yayılma bölgesi
   //xort ortalama değer
 int iset=0;
 double gset=0;
 double fac,rsq,v1,v2;
 double aret=0;
 if(iset==0)
 {
   do{
              v1=2.0*Math.random()-1.0;
              v2=2.0*Math.random()-1.0;
              rsq=v1*v1+v2*v2;
     }while(rsq>=1.0 || rsq==0.0);
   fac=Math.sqrt(-2.0*Math.log(rsq)/rsq);
   gset=v1*fac;
   iset=1;
   aret=v2*fac;
 }
 else
 {
 iset=0;
 aret=gset;
 }
 return aret/4.0*r+xort;
}

public Gen(double ixmin,double ixmax)
{//randomly generate the gene
 xmin=ixmin;
 xmax=ixmax;
 x=double_random();
}

public Gen(double s,double ixmin,double ixmax)
{
xmin=ixmin;
xmax=ixmax;
//generate gene equal to a given string
if((s>=xmin)&& (s<=xmax)) x=s;
else if(s<xmin) x=xmin;
else if(s>xmax) x=xmax;
}



public Gen()
{
xmin=0;
xmax=1.0;
x=xmin+Math.random();
```

```
}

public Gen(Gen g)
{
xmin=g.xmin;
xmax=g.xmax;
x=g.x;
}

public void copyGen(Gen g)
{
xmin=g.xmin;
xmax=g.xmax;
x=g.x;
}

public Gen copyGen()
{
 Gen g=new Gen(this);
 return g;
}

public Gen(Gen Father,Gen Mother)
{setGen(Father,Mother);}

public void  setGen(Gen Father,Gen Mother)
{
xmin=Father.xmin;
xmax=Father.xmax;
double r1=xmax-xmin;
double r=Math.random();

double d1=xmax-Father.x;
double d2=Father.x-xmin;
double d=Math.abs(Math.min(d1,d2));

double r2=gaussrandom(Father.x,d);
d1=xmax-Mother.x;
d2=Mother.x-xmin;
d=Math.abs(Math.min(d1,d2));
double r3=gaussrandom(Mother.x,d);
x=r2*r+r3*(1-r);
System.out.println("r2="+r2+"r3="+r3+"r="+r);
if(x<xmin) x=xmin;
else if(x>xmax) x=xmax;

}

public void cross1(double crossratio,Gen Father,Gen Mother)
{xmin=Father.xmin;
xmax=Father.xmax;
double r1=xmax-xmin;
```

```
double d1=xmax-Father.x;
double d2=Father.x-xmin;
double d=Math.abs(Math.min(d1,d2));
double gr1=gaussrandom(Father.x,d);
d1=xmax-Mother.x;
d2=Mother.x-xmin;
d=Math.abs(Math.min(d1,d2));
double gr2=gaussrandom(Mother.x,d);
x=gr1*crossratio+gr2*(1-crossratio);
if(x<xmin) x=xmin;
else if(x>xmax) x=xmax;
}

public void cross2(double crossratio,Gen Father,Gen Mother)
{xmin=Father.xmin;
xmax=Father.xmax;
double r1=xmax-xmin;
double d1=xmax-Father.x;
double d2=Father.x-xmin;
double d=Math.abs(Math.min(d1,d2));
double gr1=gaussrandom(Father.x,d);
d1=xmax-Mother.x;
d2=Mother.x-xmin;
d=Math.abs(Math.min(d1,d2));
double gr2=gaussrandom(Mother.x,d);
x=gr1*(1.0-crossratio)+gr2*crossratio;
if(x<xmin) x=xmin;
else if(x>xmax) x=xmax;
}
public double getX()
{//return double number equivalent of bit set number
 return x;
}
public boolean boolean_random()
{
if(Math.random()<=0.5) return false;
else return true;
}

public double double_random()
{ return (xmin+Math.random()*(xmax-xmin));}

public void mutate()
{
  double r=(xmax-xmin);
  double d1=xmax-x;
  double d2=x-xmin;
  double d=Math.abs(Math.min(d1,d2));
  x=gaussrandom(x,d);
}}
```

Gentype sınıfı

İşlev olarak yukarda anlattığımız Genotype1 sınıfının aynı işlevi görür. Gen Gauss tesadüfi sayı-double değişken türünün bir anlamda boyutlu değişkeni gibi düşünülebilir. Genetik anlamda bakıldığında kromozona eşdeğerdir. Matematik olarak düşünüldüğünde ise n boyutlu
$f(x_1,x_2,x_3,\ldots,x_n)$ fonksiyonun tüm değişkenlerine karşı gelen bir toplam değişken setidir. Fonksiyon uyum değerlerini ve kümülatif uyum değerlerini hesaplar.

Program 5.17.4-3 Gentype sınıfı Gauss tesadüfi sayı üzerinden temel gen işlemlerini(mutasyon, çaprazlama) hesaplayan gen sınıfının boyutlu şeklini içerir (canlılardaki kromozona eşdeğerdir)

```
import java.io.*;
import java.text.*;
import java.util.Locale;

abstract class f_xj
{
  // single function  multi independent variable
  // a single value is returned indiced to equation_ref
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x,1) returns the value of f[1]
  // func(x,0) returns the value of f[0]

  abstract double func(double x[]);
}

class gentype
{
// each genotype1 is member of the population
// Genotype is like a chromosone in real life genetic systems
// it can be thought as an evaluation point of a function with n variable
// f(xi)=f(x1,x2,x3,..,x_nvars)
public Gen G[]      ;  // string of variables makes a genotype1
public double Fitness;  // genotype1's fitness f(xi)
public double Upper[];  // gene's upper bound; xi_lower < xi < xi_upper
public double Lower[];  // Gene's lower bound; xi_lower < xi < xi_upper
public double RFitness; // the relative fitness
public double CFitness; // cumulative fitness
public int NVARS;       // Number of variables(Genes)
public int N;           // Number of bits defining each gene
public boolean Breeder;

  public gentype(int iNVARS,double low[],double high[],int iN,f_xj fi)
  {setgentype(iNVARS,low,high,iN,fi);}

  public void setgentype(int iNVARS,double low[],double high[],int iN,f_xj fi)
  {
  setN(iN);
  setNVARS(iNVARS);
  G=new Gen[NVARS];
  Upper=new double[NVARS];
  Lower=new double[NVARS];
  setLower(low);
  setUpper(high);
```

```
setGen();
setFitness(fi);
}

public gentype(gentype gi,f_xj fi)
{setgentype(gi,fi);}

public void setgentype(gentype gi,f_xj fi)
{
setN(gi.N);
setNVARS(gi.NVARS);
G=new Gen[NVARS];
Upper=new double[NVARS];
Lower=new double[NVARS];
setLower(gi.Upper);
setUpper(gi.Lower);
for(int i=0;i<NVARS;i++)
  G[i]=gi.G[i].copyGen();
setFitness(fi);
}

public gentype copygentype(f_xj fi)
{
gentype g1=new gentype(this,fi);
return g1;
}

public void setN(int iN)    {N=iN;}
public void setNVARS(int iNVARS)    {NVARS=iNVARS;}
public void setLower(double lbound[]) {for(int i=0;i<NVARS;i++) Lower[i]=lbound[i];}
public void setUpper(double ubound[]) {for(int i=0;i<NVARS;i++) Upper[i]=ubound[i];}
public void setFitness(f_xj fi)
{           Fitness=fi.func(getGen());}
public void setRFitness(double isum)  {RFitness=Fitness/isum;}
public void setCFitness(double ix)  {CFitness=ix;}
public double getLower(int j) {return Lower[j];}
public double getUpper(int j) {return Upper[j];}
public double getFitness() {return Fitness;}
public double getRFitness(){return RFitness;}
public double getCFitness(){return CFitness;}
public void breed() {Breeder=true;}
public void not_breed() {Breeder=false;}
public boolean isBreed() {return Breeder;}


public void setGen(int val)
{  G[val]=new Gen(Lower[val],Upper[val]);}

public void setGen()
{  for(int j=0;j<NVARS;j++)
                setGen(j);
}
```

```
 public double getGen(int val)
 { return G[val].getX();}

 public double[] getGen()
 {   double x[]=new double[NVARS];
            for(int j=0;j<NVARS;j++)
              x[j]=getGen(j);
            return x;
 }

 public String toString()
 {
 //x values, Fitness,relative fitness and cumulative fitness
 String s="";
 for(int j=0;j<NVARS;j++)
           s+=G[j].toString()+" F:";
 s+=getFitness();
 s+="RF:"+getRFitness();
 s+="CF:"+getCFitness();
 return s;
 }
}  //end of class gentype
```

Genetic2 sınıfı

Genetic algoritmanın Gen ve Gentype gauss tesadüfi sayı sistemi kullanan versiyonunun ana sınıfı Genetic2 sınıfıdır. Yapı olarak Genetik1 sınıfının aynı yapısı korunmuştur(sadece alt değişken sınıfları değiştirilmiştir) Bu sınıfta genetik optimizasyon gerçekleştirilmektedir. Burada bazı değişkenler kullanıcı tarafından verilmelidir. Bu değişkenler :

```
int    POPSIZE; //populasyonu oluşturan birey sayısı
int    MAXGENS; // nesil sayısı(maksimum iterasyon?);
int    NVARS;   //değişken sayısı
int    N;       //genlerin bit eşdeğer boyutu-değişken boyutu
double PXOVER; //üreme-çaprazlama olasılığı;
double PMUTATION; //mutasyon olasılığı
```

şeklindedir. Genetic2 sınıfı tüm Genetik optimizasyon proseslerini barındırır. Bu işlemlerden evaluate bireylerin uyumlarını (fonksiyon değerlerini) hesaplar. Genetik algoritmalarda fonksiyon değerlendirmesinde önemli olan bir konuda tüm fonksiyon değerlerinin pozitif olma gerekliliğidir. – olarak değerlendirilen fonksiyonlar sabit bir değer eklenerek artı olarak değerlendirilecek fonksiyonlara dönüştürülebilir. Bu proses fonksiyonun optimum noktasını değiştirmez.

$f(x_1,x_2,x_3,\ldots,x_n)$ fonksiyonunun maksimumu $g(x_1,x_2,x_3,\ldots,x_n) = f(x_1,x_2,x_3,\ldots,x_n) + C$ fonksiyonunun maksimumuyla aynıdır.

Bu proses kümülatif uyumu hesaplarken bir hata olmasını önler.
Evaluate prosesi bireylerin uyum değerlerini ve kümülatif uyum değerlerini hesaplar
Select metodu kümülatif uyumdaki başarılı bireyler arasından yeni nesli üretme hakkına sahip olacakları tesadüfi proses kullanarak seçer.
Crossover prosesi select prosesi tarafından seçilen bireylerden yeni bireyler üretir.
Mutate prosesi verilen belli bir yüzdedeki populasyonu mutasyona uğramasını sağlar
Calculate prosesi tüm üstteki prosesleri sırayla ve toplam nesil sayısına göre itere ederek toplam nesil sayısı sonundaki nufusu(populasyonu) elde eder.

Program 5.17.4-4 Genetic2 sınıfı temel genetik algoritma hesaplarını içerir.

```
public class Genetic2
{
        // Maximizing a function
        int    POPSIZE; //population size
        int    MAXGENS; //maximum number of generations;
        int    NVARS;   //number of independent variables of function
  int    N;       //number of genes in each variable
        double PXOVER; //probability of crossover;
  double PMUTATION; //proportion of mutated variables
  int Generation; //Current generation number
  int NBreeder; //Number of survivors in a generation
  int Best; //The best gentype in the population
  gentype Population[];
  //f_xj f;  // function to be avaluated by genetic algorithm

  public Genetic2(int iPOPSIZE,int iMAXGENS,int iNVARS,int iN,
  double iPXOVER,double iPMUTATION)
  {
  POPSIZE=iPOPSIZE;
  MAXGENS=iMAXGENS;
  NVARS=iNVARS;
  N=iN;
  PXOVER=iPXOVER;
  PMUTATION=iPMUTATION;
  NBreeder=0;
  Population=new gentype[POPSIZE+1];
        }
        public void setPopulation(double low[],double up[],f_xj fi)
        {
        for(int i=0;i<POPSIZE+1;i++)
          { Population[i]=new gentype(NVARS,low,up,N,fi);}
        }

        public gentype[] copyPopulation(f_xj fi)
        {
        gentype Po[]=new gentype[POPSIZE+1];
        for(int i=0;i<POPSIZE+1;i++)
          { Po[i]=new gentype(Population[i],fi);
            Po[i].Fitness=Population[i].Fitness;
                  Po[i].RFitness=Population[i].RFitness;
            Po[i].CFitness=Population[i].CFitness;
          }
        return Po;
        }

        public void evaluate(f_xj fi)
        {
          int mem;
          int j;
          double x[]=new double[NVARS];
```

```
          for(mem=0;mem<POPSIZE;mem++)
          {
          //System.out.println(Population[mem].toString());
          Population[mem].setFitness(fi);
          if(Population[mem].getFitness() > Population[POPSIZE].getFitness())
            {
                  Best=mem;
                  Population[POPSIZE]=Population[mem];
            for(j=0;j<NVARS;j++)
              {
              Population[POPSIZE].G[j].copyGen(Population[mem].G[j]);
                  }//end of for(j=0..
                  }//end of if
          }//end of for(mem=0;..
        }


        public void setRFCF(gentype Pi[])
        {
        //calculates relative and cumulative fitness functions
        int mem;
        int PN=Pi.length;
        double sum=0.0;
        //total fitness of population
        for(mem=0;mem<PN;mem++)
          {sum+=Pi[mem].Fitness;}
        //System.out.println("sum="+sum);
        //calculate relative fitness of each gentype
        for(mem=0;mem<PN;mem++)
          {Pi[mem].setRFitness(sum);}
        //calculate cumulative fitness
        Pi[0].setCFitness(Pi[0].getRFitness());
for(mem=1;mem<PN;mem++)
          {Pi[mem].setCFitness(Pi[mem-1].getCFitness()+Pi[mem].getRFitness());
          }
        }

        public void toString(gentype Pi[])
        {
        int mem;
        int PN=Pi.length;
        for(mem=0;mem<PN;mem++)
        {
        //list them
        System.out.println("Population["+mem+"]="+Pi[mem].toString());
        }
        }

        public void select(f_xj fi)
        {
        //select the new generation members of population
        double r;
```

```
        int mem;
        setRFCF(Population);
        //create a new population;
        gentype Po[]=copyPopulation(fi);
 setRFCF(Po);
 for(int i=0;i<POPSIZE;i++)
 {   mem=0;
    r=Math.random();
    double gf1=Population[i].getCFitness();
          double gf2;
    if (gf1 > r) Po[i]=Population[mem].copygentype(fi);
          for(mem=1;mem<POPSIZE;mem++)
      {
            gf2=Population[mem].getCFitness();
            if( gf2>=r && gf1< r ) {Po[i]=Population[mem].copygentype(fi);break;}
      }
 }
 setRFCF(Po);
        Population=Po;
        evaluate(fi);
 //toString(Population);
 }

public void crossover(f_xj fi)
{
int i,count;
int mem;
int POne,a;
int PTwo,b;
int point;
//select members for breeding
int iselect[]=new int[POPSIZE];
int counter=0;
double r;
 for(mem=0;mem<POPSIZE;mem++)
        {
          r=Math.random();
          if(r < PXOVER)
            {
                  iselect[counter++]=mem;
            Population[mem].breed();
                  NBreeder++;
            }
          else
            Population[mem].not_breed();
        }
        //System.out.println("iselect="+Matrix.toString(iselect));
        //let also best of the population to breed
        Population[Best].breed();

//loop through the population select breeding pairs
 for(mem=0;mem<POPSIZE;mem++)
```

```
    {
      //select two in popolation in random
      a=(int)(Math.random()*NBreeder)+1;
      b=(int)(Math.random()*NBreeder)+1;
      count=0;
      POne=0;
      PTwo=0;
      //select two individuals for breeding
      for(i=0;i<POPSIZE;i++)
        {
       if(Population[i].isBreed())
         {
         count++;
         if(count==a) POne=count;
         if(count==b) PTwo=count;
                 }
              }
            //perform a crossover;
              gentype Po[]=new gentype[2];
                      Po[0]=Population[POne].copygentype(fi);
                      Po[1]=Population[PTwo].copygentype(fi);
              for(i=0;i<NVARS;i++)
              {
                      Population[POne].G[i].cross1(Math.random(),Po[0].G[i],Po[1].G[i]);
                      Population[PTwo].G[i].cross1(Math.random(),Po[0].G[i],Po[1].G[i]);
                      }
                      Population[POne].setFitness(fi);
                      Population[PTwo].setFitness(fi);
            }
          }

public void mutate(f_xj fi)
{
int i;
double lbound,hbound;
int nmutations;
int member;
int var;
nmutations=(int)((double)(POPSIZE*NVARS*PMUTATION*N));
for(i=0;i<nmutations;i++)
  {
  member=(int)(Math.random()*POPSIZE);
  var=(int)(Math.random()*NVARS);
  //replace the old value with a new mutated one
  Population[member].G[var].mutate();
        //after mutation recalculate the function value
  Population[member].setFitness(fi);
  }
}

public String report()
{
```

```
  String s;
  int i;
  double best_value;
  double avg;
  double stdev;
  double sum_square;
  double square_sum;
  double sum;
  double xx;
  sum=0;
  sum_square=0.0;
  for(i=0;i<POPSIZE;i++)
    {
          xx= Population[i].getFitness();
    sum+=xx;
    sum_square=xx*xx;
          }
  avg=sum/(double)POPSIZE;
  square_sum=sum*sum/(double)POPSIZE;
  stdev=Math.sqrt(square_sum);
  best_value=Population[POPSIZE].getFitness();
  double aa[]=new double[NVARS+1];
  for(i=0;i < NVARS;i++)
   {
   aa[i]=Population[POPSIZE].getGen(i);
          }
          aa[NVARS]=Population[POPSIZE].getFitness();

  //s="Generation = "+Generation+"best value = "+Matrix.toString(aa)+"Average = "+avg+"Standart Deviation
= "+stdev;
  s="Generation = "+Generation+"best value = "+Matrix.toString(aa);
  return s;
  }

  public double[] getBest()
  {
  double aa[]=new double[NVARS+1];
  for(int i=0;i < NVARS;i++)
   {
   aa[i]=Population[POPSIZE].getGen(i);
          }
          aa[NVARS]=Population[POPSIZE].getFitness();
  return aa;
  }

  public double[] calculate(f_xj fi,boolean report)
  {
  evaluate(fi);
  Generation=0;
  while(Generation<MAXGENS)
   {
   Generation++;
```

```
    NBreeder=(int)(0.8*POPSIZE);
    select(fi);
    crossover(fi);
    mutate(fi);
    evaluate(fi);
    //if(report)
      //System.out.println(report());
    }
  return getBest();
  }
}
```

Şimdi sonuçları görmek için bir örnek problem verelim. Örnek problem olarak yukarda bit temelli genetik algoritmada kullandığımız aynı test fonksiyonunu aldık.

Program 5.17.4-5 Genetic2 sınıfı test programı.

```
import java.io.*;

class fa extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=2.0*x[0]*x[1]+2.0*x[0]-x[0]*x[0]-2.0*x[1]*x[1]+10.0;
return ff; //maksimum testi
}
}


class OPO17C
{
   public static void main(String args[]) throws IOException
   {
   int N=25;
           int POPSIZE=400;
   int MAXGENS=400;
   int NVARS=2;
   double PXOVER=0.3;
   double PMUTATION=0.02;
   fa fx=new fa();
   Genetic2 xx=new Genetic2(POPSIZE,MAXGENS,NVARS,N,PXOVER,PMUTATION);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=-1.0;high[0]=5.0;
   low[1]=-1.0;high[1]=5.0;
   xx.setPopulation(low,high,fx);
   System.out.println("best=\n"+Matrix.toStringT(xx.calculate(fx,false)));
   }
}
```

Çıktı 5.17.4-2 Genetic2 sınıfı test programı.

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" OPO17C
best=
       2.001822704663157
       0.991019679146159
      11.999802648477040

> Terminated with exit code 0.
```

Yukarda da kullandığımız bu test fonksiyonunun çözüm seti 2 ve 1 değerleridir.

5.17.5 Gene3 long temel değişken tipi tabanlı gen kullanan, genotype3, ve genetıc3 sınıflarını kullanarak genetik algoritmayı gerçekleştirme

Bundan önceki 2 bölümde bit yapısı kullanan ve tesadüfi gauss dağılımını kullanan gen yapıları ile genetik algoritma genini tanımlamıştık. Bu bölümde ise temel gen yapısını long tipi değişken kullanarak oluşturacağız. Bildiğiniz gibi burada asıl amacımız double tipi sayılar oluşturmak ancak bu sayıları her zaman için verilen sınırlar içinde tutarak oluşturmaktır. Aynı zamanda eger sayı bir anne baba setinden oluşturuluyorsa o setlerin genetik karekterlerini (özeliklerini) belli bir oranda taşımalıdır. Matematiksel olarak baktığımızda bu rakamlardan çok farklı olmamalıdır diyebiliriz. İki değişkenli set (burada long ve double) bize birinci değişken olası tüm değerleri alırken ikinci değişkenin verilen değişken aralığında kalmasını sağlar. Burada da Genotype3 ve Gene3 sınıflarında daha önceki aynı yapı korunmuş, sadece temel değişken olarak Gene3 sınıfını kullanması sağlanmıştır. Systemimizde Maksimum long tipi değişken olarak 18 digitli x'_max=999999999999999999L kullanılmıştır. Bu durumda x_min ve x_max double sayıları aralığında sınırlı x double sayımız (genin double değeri)

$$x = x\_min + x\left(\frac{x\_max - x\_min}{999999999999999999L}\right)$$

Bu denklemdeki x' long tipi long tipi gen değerimizdir. Binary (ikili) genden farklı olarak bu gen yapımızda her digitin 0 ile 9 arasında değişebildiğini not edelim.

$$x' = 999999999999999999L\left(\frac{x - x\_min}{x\_max - x\_min}\right)$$

Program 5.17.5-1 Gene3 sınıfı

```
// Dr. M. Turhan ÇOBAN
// EGE University School of Engineering, Mechanical Engineering Department
// package : Numerical analysis package
// subject : optimization
// group   : Genetic Algorithms
// Gene1 Class
// Gene1 class generates integer array to represent a double number
// actual binary defination changes depends on number of bits, xmin and xmax
// binary data stored in b BitSet array

import java.util.*;
import java.io.*;

public class Gene3 extends Object
{
//long imin=0;
int N=9;
long imax=999999999999999999L;
```

```
public long ix;
public double x;//Limit value for integer(2^N-1)
public double xmin,xmax;//maximum and minimum limits of double number

public Gene3(double ixmin,double ixmax)
{//randomly generate the gene
 xmin=ixmin;
 xmax=ixmax;
 ix=(long)(imax*Math.random());
 x=xmin+(xmax-xmin)*(double)ix/(double)imax;
}

public Gene3(Gene3 g)
{
// copy a Gene from another one
copyGene(g);
}

public void copyGene(Gene3 g)
{
 ix=g.ix;
 x=g.x;
 xmin=g.xmin;
 xmax=g.xmax;
}

public Gene3 copyGene()
{Gene3 g=new Gene3(x,xmin,xmax);
 return g;
}

public Gene3(Gene3 Father,Gene3 Mother)
{cross(Math.random(),Father,Mother);}

public Gene3(double xi,double maxi,double mini)
{setGene(xi,maxi,mini);}


public Gene3(double crossratio,Gene3 Father,Gene3 Mother)
{   if(boolean_random()) cross(crossratio,Father,Mother);
   else                cross(crossratio,Mother,Father);
}

public void  setGene(long iix,double ixmin,double ixmax)
{
 xmin=ixmin;
 xmax=ixmax;
 ix=iix;
 x=xmin+(xmax-xmin)*(double)ix/(double)imax;
}

public void  setGene(double idx,double ixmin,double ixmax)
```

```
{
 xmin=ixmin;
 xmax=ixmax;
 if(idx>=xmin && idx<=xmax)      x=idx;
 else if(idx<xmin) x=xmin;
 else x=xmax;
 ix=(long)((x-xmin)/(xmax-xmin)*imax);
}

public void  setGene(double idx)
{

 if(idx>=xmin && idx<=xmax)      x=idx;
 else if(idx<xmin) x=xmin;
 else x=xmax;
 ix=(long)((x-xmin)/(xmax-xmin)*imax);
}

public void cross(Gene3 Father,Gene3 Mother)
{double crossratio=Math.random();
cross(crossratio,Father,Mother);
}

public double getX()
{return x;}

public void cross(double crossratio,Gene3 Father,Gene3 Mother)
{
Long IntFx=new Long(Father.ix);
String SFx=IntFx.toString();
int nSFx=SFx.length();
Long IntMx=new Long(Mother.ix);
String SMx=IntMx.toString();
int nSMx=SMx.length();
if(nSFx<N)
{String ss="";for(int i=0;i<(N-nSFx);i++){ss+="0";};SFx=ss+SFx;}
if(nSMx<N)
{String ss="";for(int i=0;i<(N-nSMx);i++){ss+="0";};SMx=ss+SMx;}
int N2 = (int)(crossratio*N+0.02)/2;
int N1=2*N2;
int N3=N1+(N-N1)/2;
StringBuffer dest = new StringBuffer(N);
int i;
for (i = 0; i <N2; i++)
{ dest.append(SFx.charAt(i));
  dest.append(SMx.charAt(i));
}
for (i = N1; i <N3; i++)
{ dest.append(SFx.charAt(i));}
for (i = N3; i <N; i++)
{ dest.append(SMx.charAt(i));}
String s=dest.toString();
```

```
long iix=Long.parseLong(s);
ix=iix;
xmin=Father.xmin;
xmax=Father.xmax;
x=xmin+(xmax-xmin)*(double)ix/(double)imax;
}


public boolean boolean_random()
{
if(Math.random()<=0.5) return false;
else return true;
}

public void mutate()
{
int N1 = (int)(N*Math.random());
Long IntFx=new Long(ix);
String SFx=IntFx.toString();
int nSFx=SFx.length();
if(nSFx<N)
{String ss="";for(int i=0;i<(N-nSFx);i++){ss+="0";};SFx=ss+SFx;}
StringBuffer dest = new StringBuffer(N);
int i;
for (i = 0; i <N; i++)
{  if(i==N1)
   {  int i1=(int)(9.0*Math.random());
      String s2=Integer.toString(i1);
      dest.append(s2.charAt(0));
   }
  else {dest.append(SFx.charAt(i));}
}
String s=dest.toString();
long iix=Long.parseLong(s);
ix=iix;
x=xmin+(xmax-xmin)*(double)ix/(double)imax;
//==================
}

public String toString()
{String s;
 s="x="+x+"ix="+ix+"xmin="+xmin+"xmax="+xmax;
 return s;
}
}
```

Program 5.17.5-2 genotype3 ve Genetic3 sınıfları

```
import java.io.*;
import java.text.*;
import java.util.Locale;

abstract class f_xj extends Mathd
```

```
{
  // single function  multi independent variable
  // a single value is returned indiced to equation_ref
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x,1) returns the value of f[1]
  // func(x,0) returns the value of f[0]

  abstract double func(double x[]);
}

class genotype3
{
// each genotype3 is member of the population
// Genotype is like a chromosone in real life genetic systems
// it can be thought as an evaluation point of a function with n variable
// f(xi)=f(x1,x2,x3,..,x_nvars)
public Gene3 G[]      ;  // string of variables makes a genotype3
public double Fitness;  // genotype3's fitness f(xi)
public double Upper[];  // gene's upper bound; xi_lower < xi < xi_upper
public double Lower[];  // Gene's lower bound; xi_lower < xi < xi_upper
public double RFitness; // the relative fitness
public double CFitness; // cumulative fitness
public int NVARS;       // Number of variables(Genes)
public int N;           // Number of bits defining each gene
public boolean Breeder;

  public genotype3(int iNVARS,double low[],double high[],int iN,f_xj fi)
  {setgenotype(iNVARS,low,high,iN,fi);}

  public void setgenotype(int iNVARS,double low[],double high[],int iN,f_xj fi)
  {
  setN(iN);
  setNVARS(iNVARS);
  G=new Gene3[NVARS];
  Upper=new double[NVARS];
  Lower=new double[NVARS];
  setLower(low);
  setUpper(high);
  setGene();
  setFitness(fi);
  }

  public genotype3(genotype3 gi,f_xj fi)
  {setgenotype(gi,fi);}

  public void setgenotype(genotype3 gi,f_xj fi)
  {
  setN(gi.N);
  setNVARS(gi.NVARS);
  G=new Gene3[NVARS];
  Upper=new double[NVARS];
```

```
Lower=new double[NVARS];
setLower(gi.Upper);
setUpper(gi.Lower);
for(int i=0;i<NVARS;i++)
  G[i]=gi.G[i].copyGene();
setFitness(fi);
}

public genotype3 copygenotype(f_xj fi)
{
genotype3 g1=new genotype3(this,fi);
return g1;
}

public void setN(int iN)    {N=iN;}
public void setNVARS(int iNVARS)    {NVARS=iNVARS;}
public void setLower(double lbound[]) {for(int i=0;i<NVARS;i++) Lower[i]=lbound[i];}
public void setUpper(double ubound[]) {for(int i=0;i<NVARS;i++) Upper[i]=ubound[i];}
public void setFitness(f_xj fi)
{          Fitness=fi.func(getGene());}
public void setRFitness(double isum)  {RFitness=Fitness/isum;}
public void setCFitness(double ix)  {CFitness=ix;}
public double getLower(int j) {return Lower[j];}
public double getUpper(int j) {return Upper[j];}
public double getFitness() {return Fitness;}
public double getRFitness(){return RFitness;}
public double getCFitness(){return CFitness;}
public void breed() {Breeder=true;}
public void not_breed() {Breeder=false;}
public boolean isBreed() {return Breeder;}


public void setGene(int val)
{  G[val]=new Gene3(Lower[val],Upper[val]);}

public void setGene()
{  for(int j=0;j<NVARS;j++)
              setGene(j);
}

public double getGene(int val)
{ return G[val].getX();}

public double[] getGene()
{   double x[]=new double[NVARS];
            for(int j=0;j<NVARS;j++)
              x[j]=getGene(j);
            return x;
}

public String toString()
{
```

```
  //x values, Fitness,relative fitness and cumulative fitness
  String s="";
  for(int j=0;j<NVARS;j++)
            s+=G[j].toString()+" F:";
  s+=getFitness();
  s+="RF:"+getRFitness();
  s+="CF:"+getCFitness();
  return s;
  }
}  //end of class genotype3

public class Genetic3
{
            // Maximizing a function
            int    POPSIZE; //population size
            int    MAXGENS; //maximum number of generations;
            int    NVARS;   //number of independent variables of function
    int    N;       //number of genes in each variable
            double PXOVER; //probability of crossover;
    double PMUTATION; //proportion of mutated variables
    int Generation; //Current generation number
    int NBreeder; //Number of survivors in a generation
    int Best; //The best genotype3 in the population
    genotype3 Population[];
    //f_xj f;  // function to be avaluated by genetic algorithm

    public Genetic3(int iPOPSIZE,int iMAXGENS,int iNVARS,int iN,
    double iPXOVER,double iPMUTATION)
    {
    POPSIZE=iPOPSIZE;
    MAXGENS=iMAXGENS;
    NVARS=iNVARS;
    N=iN;
    PXOVER=iPXOVER;
    PMUTATION=iPMUTATION;
    NBreeder=0;
    Population=new genotype3[POPSIZE+1];
            }
            public void setPopulation(double low[],double up[],f_xj fi)
            {
            for(int i=0;i<POPSIZE+1;i++)
              { Population[i]=new genotype3(NVARS,low,up,N,fi);}
            }

            public genotype3[] copyPopulation(f_xj fi)
            {
            genotype3 Po[]=new genotype3[POPSIZE+1];
            for(int i=0;i<POPSIZE+1;i++)
              { Po[i]=new genotype3(Population[i],fi);
                Po[i].Fitness=Population[i].Fitness;
                        Po[i].RFitness=Population[i].RFitness;
                Po[i].CFitness=Population[i].CFitness;
```

```
  }
return Po;
}

public void evaluate(f_xj fi)
{
  int mem;
  int j;
  double x[]=new double[NVARS];
  for(mem=0;mem<POPSIZE;mem++)
  {
  //System.out.println(Population[mem].toString());
  Population[mem].setFitness(fi);
  if(Population[mem].getFitness() > Population[POPSIZE].getFitness())
    {
        Best=mem;
        Population[POPSIZE]=Population[mem];
    for(j=0;j<NVARS;j++)
      {
      Population[POPSIZE].G[j].copyGene(Population[mem].G[j]);
          }//end of for(j=0..
        }//end of if
  }//end of for(mem=0;..
}


public void setRFCF(genotype3 Pi[])
{
//calculates relative and cumulative fitness functions
int mem;
int PN=Pi.length;
double sum=0.0;
//total fitness of population
for(mem=0;mem<PN;mem++)
  {sum+=Pi[mem].Fitness;}
//System.out.println("sum="+sum);
//calculate relative fitness of each genotype3
for(mem=0;mem<PN;mem++)
  {Pi[mem].setRFitness(sum);}
//calculate cumulative fitness
Pi[0].setCFitness(Pi[0].getRFitness());
for(mem=1;mem<PN;mem++)
  {Pi[mem].setCFitness(Pi[mem-1].getCFitness()+Pi[mem].getRFitness());
  }
}

public void toString(genotype3 Pi[])
{
int mem;
int PN=Pi.length;
for(mem=0;mem<PN;mem++)
{
```

```
        //list them
        System.out.println("Population["+mem+"]="+Pi[mem].toString());
        }
        }

        public void select(f_xj fi)
        {
        //select the new generation members of population
        double r;
        int mem;
        setRFCF(Population);
        //create a new population;
        genotype3 Po[]=copyPopulation(fi);
 setRFCF(Po);
 for(int i=0;i<POPSIZE;i++)
 {   mem=0;
    r=Math.random();
    double gf1=Population[i].getCFitness();
            double gf2;
    if (gf1 > r) Po[i]=Population[mem].copygenotype(fi);
            for(mem=1;mem<POPSIZE;mem++)
      {
              gf2=Population[mem].getCFitness();
              if( gf2>=r && gf1< r ) {Po[i]=Population[mem].copygenotype(fi);break;}
      }
 }
 setRFCF(Po);
         Population=Po;
         evaluate(fi);
 //toString(Population);
 }

public void crossover(f_xj fi)
{
int i,count;
int mem;
int POne,a;
int PTwo,b;
int point;
//select members for breeding
int iselect[]=new int[POPSIZE];
int counter=0;
double r;
 for(mem=0;mem<POPSIZE;mem++)
         {
          r=Math.random();
          if(r < PXOVER)
            {
                    iselect[counter++]=mem;
            Population[mem].breed();
                    NBreeder++;
            }
```

```
            else
               Population[mem].not_breed();
          }
          //System.out.println("iselect="+Matrix.toString(iselect));
          //let also best of the population to breed
          Population[Best].breed();

//loop through the population select breeding pairs
 for(mem=0;mem<POPSIZE;mem++)
    {
      //select two in popolation in random
      a=(int)(Math.random()*NBreeder)+1;
      b=(int)(Math.random()*NBreeder)+1;
      count=0;
      POne=0;
      PTwo=0;
      //select two individuals for breeding
      for(i=0;i<POPSIZE;i++)
        {
        if(Population[i].isBreed())
          {
          count++;
          if(count==a) POne=count;
          if(count==b) PTwo=count;
                  }
                }
              //perform a crossover;
                genotype3 Po[]=new genotype3[2];
                        Po[0]=Population[POne].copygenotype(fi);
                        Po[1]=Population[PTwo].copygenotype(fi);
                for(i=0;i<NVARS;i++)
                {
                        Population[POne].G[i].cross(Math.random(),Po[0].G[i],Po[1].G[i]);
                        Population[PTwo].G[i].cross(Math.random(),Po[0].G[i],Po[1].G[i]);
                        }
                        Population[POne].setFitness(fi);
                        Population[PTwo].setFitness(fi);
              }
            }

public void mutate(f_xj fi)
{
int i;
double lbound,hbound;
int nmutations;
int member;
int var;
nmutations=(int)((double)(POPSIZE*NVARS*PMUTATION*N));
for(i=0;i<nmutations;i++)
  {
  member=(int)(Math.random()*POPSIZE);
  var=(int)(Math.random()*NVARS);
```

```
    //replace the old value with a new mutated one
    Population[member].G[var].mutate();
            //after mutation recalculate the function value
    Population[member].setFitness(fi);
    }
  }

  public String report()
  {
  String s;
  int i;
  double best_value;
  double avg;
  double stdev;
  double sum_square;
  double square_sum;
  double sum;
  double xx;
  sum=0;
  sum_square=0.0;
  for(i=0;i<POPSIZE;i++)
    {
            xx= Population[i].getFitness();
    sum+=xx;
    sum_square=xx*xx;
            }
  avg=sum/(double)POPSIZE;
  square_sum=sum*sum/(double)POPSIZE;
  stdev=Math.sqrt(square_sum);
  best_value=Population[POPSIZE].getFitness();
  double aa[]=new double[NVARS+1];
  for(i=0;i < NVARS;i++)
   {
   aa[i]=Population[POPSIZE].getGene(i);
            }
            aa[NVARS]=Population[POPSIZE].getFitness();

  //s="Generation = "+Generation+"best value = "+Matrix.toString(aa)+"Average = "+avg+"Standart Deviation
= "+stdev;
  s="Generation = "+Generation+"best value = "+Matrix.toString(aa);
  return s;
  }

  public double[] getBest()
  {
  double aa[]=new double[NVARS+1];
  for(int i=0;i < NVARS;i++)
   {
   aa[i]=Population[POPSIZE].getGene(i);
            }
            aa[NVARS]=Population[POPSIZE].getFitness();
  return aa;
```

```
   }

   public double[] calculate(f_xj fi,boolean report)
   {
   evaluate(fi);
   Generation=0;
   while(Generation<MAXGENS)
     {
     Generation++;
     NBreeder=(int)(0.8*POPSIZE);
     select(fi);
     crossover(fi);
     mutate(fi);
     evaluate(fi);
     //if(report)
        //System.out.println(report());
     }
   return getBest();
   }
}
```

Program 5.17.5-3 Gene3 sınıfı testi Gene3test

```
public class Gene3test
{ public static void main(String arg[])
  { Gene3 x1=new Gene3(0.0,10.0);
    System.out.println("x1 = "+x1.toString());
    Gene3 x2=new Gene3(0.0,10.0);
    System.out.println("x2 = "+x2.toString());
    double xx=Math.random();
    Gene3 x3=new Gene3(xx,x1,x2);
    System.out.println("xx = "+xx+"x3 = "+x3.toString());
    x3.mutate();
    System.out.println("mutasyon sonucu x3 = "+x3.toString());
  }
}
```

Çıktı 5.17.5-1 Gene3 sınıfı testi

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" Gene3test
x1 = x = 8.792631687926317ix = 87926316xmin = 0.0xmax = 10.0
x2 = x = 8.852918088529181ix = 88529180xmin = 0.0xmax = 10.0
xx = 0.8077212904815896x3 = x = 8.887598688875986ix = 88875986xmin = 0.0xmax = 10.0
mutasyon sonucu x3 = x = 8.887498688874986ix = 88874986xmin = 0.0xmax = 10.0

> Terminated with exit code 0.
```

Program 5.17.5-4 Genetik optimizasyon örneği 1 , bir boyutlu fonksiyon

```
import java.io.*;

class fa extends f_xj
{
public double func(double x[])
```

```
{
//çözümü istenen fonksiyon
double ff;
ff=x[0]*Math.sin(10.0*Math.PI*x[0])+3.0;
return ff; //maksimum testi
}
}

class OPO17A3
{
   public static void main(String args[]) throws IOException
   {
   int N=18;
          int POPSIZE=200;
   int MAXGENS=200;
   int NVARS=1;
   double PXOVER=0.3;
   double PMUTATION=1.0;
   fa fx=new fa();
   Genetic3 xx=new Genetic3(POPSIZE,MAXGENS,NVARS,N,PXOVER,PMUTATION);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=-1.0;high[0]=2.0;
   xx.setPopulation(low,high,fx);
   System.out.println("best=\n"+Matrix.toStringT(xx.calculate(fx,false)));
   }
}
```

Çıktı 5.17.5-1 optimizasyon örneği 1 , bir boyutlu fonksiyon

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" OPO17A3
best=
      1.852909818529098
      4.845173160789873
> Terminated with exit code 0.
```

Program 5.17.5-5 Genetik optimizasyon örneği 2 , çok boyutlu fonksiyon

```
import java.io.*;
import javax.swing.*;

class fa extends f_xj
{
//çözümü istenen fonksiyon
 double func(double x[])
 {
   double ff= 1.0+Math.cos(Math.PI*(x[0]-3.0))*Math.cos(2.0*Math.PI*(x[1]-2.0))/
   (1+(x[0]-3.0)*(x[0]-3.0)+(x[1]-2.0)*(x[1]-2.0));
   return ff; //maksimum testi;
  }
}
class fb extends f_xj
{
```

```
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
fa fx=new fa();
return -fx.func(x); //minimum testi
}}

class OPO19C3
{
   public static void main(String args[]) throws IOException
   {
   int N=18;
          int POPSIZE=200;
   int MAXGENS=200;
   int NVARS=2;
   double PXOVER=0.3;
   double PMUTATION=0.02;
   fa fax=new fa();
   fb fbx=new fb();
   Genetic3 xx=new Genetic3(POPSIZE,MAXGENS,NVARS,N,PXOVER,PMUTATION);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=-10.0;high[0]=10.0;
   low[1]=-10.0;high[1]=10.0;
   xx.setPopulation(low,high,fax);
   double a[]=xx.calculate(fax,false);
   double x0[]=new double[NVARS];
   for(int i=0;i<NVARS;i++) x0[i]=a[i];
   String s="genetik algoritma sonucu =\n"+Matrix.toStringT(a);
    String s2="Genetik algoritma  çok değişkenli fonksiyon optimizasyonu:";
   System.out.println(s2+s);
   }
}
```

Çıktı 5.17.5-2 optimizasyon örneği 2 , iki boyutlu fonksiyon

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" NA109
Genetic algorithm multi-variable optimization:results of genetic algorithms =
      3.000000022020719500000000
      2.000000001661911500000000
      1.999999999999971000000000
> Terminated with exit code 0.
```

5.18 Global tesadüfi çözüm metodları :Monte-Carlo optimizasyonu

Monte carlo optimizasyonu yöntem olarak oldukça basittir, sınırları tanımlanmış olan sınırları içinde kalan noktalar tesadüfi olarak seçilir ve değerlendirilir. Nokta sayısı çok arttığında istatistik olarak tüm alan taranmuş olur. Bu yöntem temel olarak grafik yöntemine benzetilebilir, ancak toplam fonksiyon değerlendirme sayısı grafik yönteminden bile çok fazla olmalıdır. Temel olarak çok pratik bir yöntem olmamakla birlikte yazılması ve kullanılmasındaki kolaylık nedeniyle kullanılabilir.

Program 5.18.1 Monte-Carlo metodu

```
// Dr. M. Turhan ÇOBAN
//
// EGE University School of Engineering, Mechanical Engineering Department
// package : Numerical analysis package
// subject : optimization
// group   : Monte-Carlo optimisation(maximum) Algorithms
// monte_carlo_opt Class
// class selects random points at the given area and selects the best
// results found
//

import java.util.*;
import java.io.*;

abstract class f_xj extends Mathd
{
  // single function  multi independent variable
  // a single value is returned indiced to equation_ref
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x,1) returns the value of f[1]
  // func(x,0) returns the value of f[0]

  abstract double func(double x[]);
}

public class monte_carlo_opt extends Object
{
int N;
public double max[];
public double min[];
public double x;

public monte_carlo_opt(int Ni,double imin[],double imax[])
{
 max=new double[imin.length];
 min=new double[imin.length];
 N=Ni;
 int n=imin.length;
 for(int i=0;i<n;i++)
 {min[i]=imin[i];max[i]=imax[i];}
}

public double[] monte_carlo_max(f_xj fi)
{
 int n=min.length;
 double x[]=new double[n];
 double xmax[]=new double[n];
 double max_number=-1e99;
 double f;
```

```
 for(int k=0;k<N;k++)
 {
 for(int i=0;i<n;i++)
 {x[i]=min[i]+(max[i]-min[i])*Math.random();}
 f=fi.func(x);
 if(f>max_number)
   {  max_number=f;
      for(int i=0;i<n;i++)
         {xmax[i]=x[i];}
   }
 }
 return xmax;
}
}
```

Program 5.18-2 Monte-Carlo metodu bir boyutlu test programı

```
import java.io.*;

class fa extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=x[0]*Math.sin(10.0*Math.PI*x[0])+3.0;
return ff; //maksimum testi
}
}

class OPO17MC
{
   public static void main(String args[]) throws IOException
   {
            fa fx=new fa();
   double low[]=new double[1];
   double high[]=new double[1];
   low[0]=-1.0;high[0]=2.0;
   monte_carlo_opt mco=new monte_carlo_opt(100000,low,high);
   double xmax[]=mco.monte_carlo_max(fx);
   for(int i=0;i<xmax.length;i++)
   {System.out.println(xmax[i]);}
   }
}
```

Çıktı 5.18-1 Monte-Carlo metodu bir boyutlu program

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" OPO17MC
1.850550795059335

> Terminated with exit code 0.
```

Program 5.18-3 Monte-Carlo metodu iki boyutlu test programı

```
import java.io.*;

class fa extends f_xj
{
//çözümü istenen fonksiyon
 double func(double x[])
 {
   double ff= 1.0+Math.cos(Math.PI*(x[0]-3.0))*Math.cos(2.0*Math.PI*(x[1]-2.0))/
   (1+(x[0]-3.0)*(x[0]-3.0)+(x[1]-2.0)*(x[1]-2.0));
   return ff; //maksimum testi;
  }
}
class fb extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
fa fx=new fa();
return -fx.func(x); //minimum testi
}
}

class OPO19MC
{
   public static void main(String args[]) throws IOException
   {
          fb fx=new fb();
   double low[]=new double[2];
   double high[]=new double[2];
   low[0]=-10.0;high[0]=10.0;
   low[1]=-10.0;high[1]=10.0;
   //10 million iteration
   monte_carlo_opt mco=new monte_carlo_opt(10000000,low,high);
   double xmax[]=mco.monte_carlo_max(fx);
   for(int i=0;i<xmax.length;i++)
   {System.out.println(xmax[i]);}
   }
}
```

Çıktı 5.18-2 Monte-Carlo metodu iki boyutlu program

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" OPO19MC
2.9959754476624685
1.5192404684258225

> Terminated with exit code 0.
```

5.19 Global tesadüfi çözüm metodları: iterasyonlu tepe tırmanma

Daha önce en dik yamaç metod ile geometrik arama yöntemi ile optimizasyon problemini incelemiştik. Global optimumu bulmak için standart tepe tırmanma yöntemi yeterli değildir, çünki yerel çözüme ulaşınca durur. Eğer global bir metod oluşturmak ister isek yerel en iyi çözümü bulduktan sonra arama bölgesini değiştirerek işleme devam etmek gereklidir. İterasyonlu tepe tırmanma metodunu bir algoritma olarak yazacak olursak:

```
İteratif tepe tırmanma algoritması
başla
    t=0
    while
        local=false
        vc değişkenini tesadüfi olarak seç
        vc için fonksiyonu hesapla
        while
            vc nin yakınlarında 30 civarında komşu nokta seç ve bu noktaları değerlendir.
            Bu noktalar arasında maksimumu veren noktayı vn çözüm vektörü olarak seç
            if f(vc) < f(vn)  then  vc = vn
            else  local=true
        until local
        t=t+1
     until t=MAXIMUM
bitir
```

Buradaki önemli problemlerden birisi komşu noktalar nasıl tespit edeceğimizdir. Örneğin genetik programlamada kullandığımız Gene sınıfından yararlanabiliriz. Vc Gene sınıfında oluşturulmuşsa bir bit değerini değiştirmek komşu bir nokta elde etmemizi sağlayacaktır.

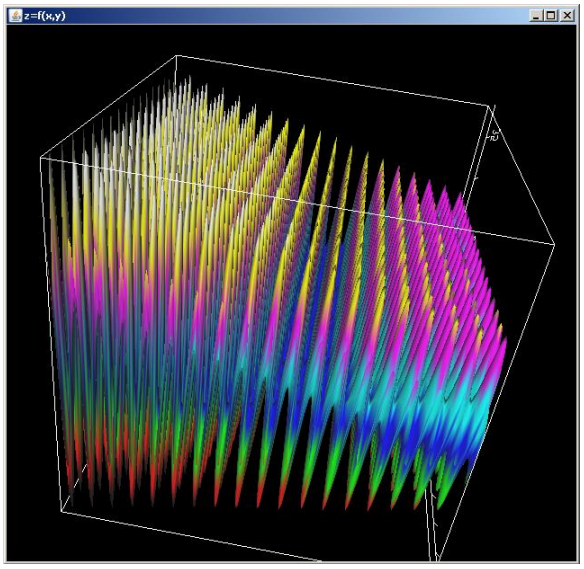
Program 5.19-1 İterasyonlu tepe tırmanma – gene-genotyoe sınıfını kullanan versiyon

```
import java.io.*;
import java.text.*;
import java.util.Locale;

public class iteratedhillclimber
{
    // Maximizing a function
   //iterated hill climber
   // a version steepest ascent with statistical approach to variable (endikyamaç)
   genotype1 Vc;
   genotype1 Vn[];
   genotype1 best;
   int MAX;
   double t;
   boolean local;
   int POPSIZE;
   int NVARS;
   int N;//number of bit representation for each variable

   //f_xj f;  // function to be avaluated by genetic algorithm

   public iteratedhillclimber(int iPOPSIZE,int iNVARS,int iN)
   {
   POPSIZE=iPOPSIZE;//Vn size number of neigbour variable f.e. 30
   NVARS=iNVARS; // number of variables equation to solve
```

```
MAX=20;
N=iN;          // number of bits represent each variable for example 32
Vn=new genotype1[POPSIZE+1];
t=0;
local=false;
        }

        public void setVc(double low[],double up[],f_xj fi)
        {
        //select originalgene andomly
        Vc=new genotype1(NVARS,low,up,N,fi);
        }
        public void setBest(genotype1 gi,f_xj fi)
        {
        //select originalgene andomly
        best=new genotype1(gi,fi);
        }
public void setVc(genotype1 gi,f_xj fi)
        {
        //select original gene from other gene
        Vc=new genotype1(gi,fi);
        }
        public void setVn(f_xj fi)
        {//select neighbouring genes similar to the original one
        for(int i=0;i<POPSIZE+1;i++)
          { Vn[i]=new genotype1(Vc,fi);}
           mutate(fi);
        }


        public void evaluate(f_xj fi)
        {
          int mem;
          int j;
          double x[]=new double[NVARS];
          for(mem=0;mem<POPSIZE;mem++)
          {
          //System.out.println(Vn[mem].toString());
          Vn[mem].setFitness(fi);
          if(Vn[mem].getFitness() > Vn[POPSIZE].getFitness())
             {
                    Vn[POPSIZE]=Vn[mem];
             for(j=0;j<NVARS;j++)
                {
                Vn[POPSIZE].G[j].copyGene(Vn[mem].G[j]);
                        }//end of for(j=0..
                    }//end of if
          }//end of for(mem=0;..
        }

        public void toString(genotype Pi[])
        {
```

```
        int mem;
        int PN=Pi.length;
        for(mem=0;mem<PN;mem++)
        {
        //list them
        System.out.println("Vn["+mem+"]="+Pi[mem].toString());
        }
        }


public void mutate(f_xj fi)
{
int i;
double lbound,hbound;
int nmutations;
int member;
int var;
nmutations=(int)((double)(POPSIZE*NVARS*N));
for(i=0;i<nmutations;i++)
  {
  member=(int)(Math.random()*POPSIZE);
  var=(int)(Math.random()*NVARS);
  //replace the old value with a new mutated one
  Vn[member].G[var].mutate();
  //after mutation recalculate the function value
  Vn[member].setFitness(fi);
  }
}


public double[] getBest(genotype1 b)
{
double aa[]=new double[NVARS+1];
for(int i=0;i < NVARS;i++)
 {
 aa[i]=b.getGene(i);
        }
        aa[NVARS]=b.getFitness();
return aa;
}

public double[] calculate(double low[],double up[],f_xj fi,boolean report)
{
 setVc(low,up,fi);// new set of random selection in each time
 setBest(Vc,fi);
        do
 {
        local=true;
 do
 {
 setVn(fi);
 evaluate(fi);
```

```
    if(Vn[POPSIZE].getFitness() > Vc.getFitness())
    {
                setVc(Vn[POPSIZE], fi);
                if(Vn[POPSIZE].getFitness() > best.getFitness()) setBest(Vn[POPSIZE],fi);
    }
    else local=false;
    } while(local);
    t++;
    setVc(low,up,fi); //new random value
    if(report)
      System.out.println(Matrix.toString(getBest(best)));
    } while(t<MAX);
  return getBest(best);
  }
}
```



(bolum5_027.jpg,Resimaltı **:**)

Program 5.19-2 İterasyonlu tepe tırmanma – gene sınıfını kullanan versiyon çıktı programı

```
//iterated hillclimber
//iterasyonlu en dik yamaç tırmanıcı
import java.io.*;
import javax.swing.JOptionPane;


class fa extends f_xj
{
//çözümü istenen fonksiyon
 double func(double x[])
 {
   double ff= 21.5+x[0]*Math.sin(4.0*Math.PI*x[0])+x[1]*Math.sin(20.0*Math.PI*x[1]);
   return ff; //maksimum testi
}
}

class fb extends f_xj
```

```
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
fa fx=new fa();
return -fx.func(x); //maksimum testi
}
}

class OPO20
{
   public static void main(String args[]) throws IOException
   {
   int N=18;
           int POPSIZE=40;
   int MAXGENS=400;
   int NVARS=2;
   fa fax=new fa();
   fb fbx=new fb();
   iteratedhillclimber xx=new iteratedhillclimber(POPSIZE,NVARS,N);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=-3.0;high[0]=12.1;
   low[1]=4.1;high[1]=5.8;
   double a[]=xx.calculate(low,high,fax,false);
   double x0[]=new double[NVARS];
   for(int i=0;i<NVARS;i++) x0[i]=a[i];
   String s="iterativ tepe tırmanma  sonucu =\n"+Matrix.toStringT(a);
    String s2="Genetik algoritma ve newton çok değişkenli fonksiyon optimizasyonun birlikte kullanımı:";
    JOptionPane.showMessageDialog(null,s,s2,JOptionPane.PLAIN_MESSAGE);
   }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO20
istatistiksel iteratif tepe tırmanma çok değişkenli fonksiyon optimizasyonu:iterativ tepe tırmanma  sonucu =
      11.61569520366908600000000000
       5.63499466780106900000000000
      37.59629635465318000000000000


> Terminated with exit code 0.
```

Program 5.19-3 İterasyonlu tepe tırmanma – double sınıfını ve Gauss tesadüfi fonksiyonlarını kullanan versiyon

```
import java.io.*;
import java.text.*;
import java.util.Locale;


public class IHC
{
```

```
        // Maximizing a function
    // iterated Hill climber
    // a statistical approach
    double T;
    double Vc[];
    double Vn[][];
    double best[];
    int MAX;//number
    int GLIMIT;//arka arkaya başarılı steplerin sayısı
    boolean local;
    int NVARS;//numberof variables
    int N;//number of bit representation for each variable
    double T0;
    double k,k0;//time index
    int POPSIZE;
    double low[];
    double high[];
    int t;
    //f_xj f;  // function to be avaluated by genetic algorithm

    public IHC(int iPOPSIZE,int iNVARS,int iN,double Ti)
    {
        T=Ti;
        T0=T;
    POPSIZE=iPOPSIZE;
    NVARS=iNVARS; // number of variables equation to solve
    MAX=100;
    N=iN;          // number of bits represent each variable for example 32
    local=false;

    t=0;
    Vc=new double[NVARS];
    Vn=new double[POPSIZE+1][NVARS];
    best=new double[NVARS];
    low=new double[NVARS];
    high=new double[NVARS];
    local=true;
        }
public static double  gaussrandom(double xort,double r)
{ int iset=0;
 double gset=0;
 double fac,rsq,v1,v2;
 double aret=0;
 if(iset==0)
 {
   do{
        v1=2.0*Math.random()-1.0;
        v2=2.0*Math.random()-1.0;
        rsq=v1*v1+v2*v2;
    }while(rsq>=1.0 || rsq==0.0);
   fac=Math.sqrt(-2.0*Math.log(rsq)/rsq);
   gset=v1*fac;
```

```
 iset=1;
 aret=v2*fac;
}
else
{
iset=0;
aret=gset;
}
return aret/4.0*r+xort;
}

        public void setVc(double ilow[],double ihigh[],f_xj fi)
        {

        //select original gene randomly
        for(int i=0;i<NVARS;i++)
          {low[i]=ilow[i];high[i]=ihigh[i];Vc[i]=(high[i]-low[i])*Math.random();}
  }

        public void setBest(double gi[],f_xj fi)
        {
        //select originalgene andomly
        for(int i=0;i<NVARS;i++)
          {best[i]=gi[i];}
        }

  public void setVc(double gi[],f_xj fi)
        {
        //select original gene from other gene
                for(int i=0;i<NVARS;i++)
          {Vc[i]=gi[i];}
        }

        public void setVn(f_xj fi)
        {//select neighbouring genes similar to the original one
        for(int i=0;i<POPSIZE;i++)
          { for(int j=0;j<NVARS;j++)
          {Vn[i][j]=Vc[j];}}
          mutate(fi);
        }

 public void mutate(f_xj fi)
 {
 int i;
 double lbound,hbound;
 int nmutations;
 int member;
 int var;
 nmutations=(int)((double)(POPSIZE*NVARS*N));
 for(i=0;i<nmutations;i++)
   {
  member=(int)(Math.random()*POPSIZE);
```

```
   var=(int)(Math.random()*NVARS);
   //replace the old value with a new mutated one
   //neigboring factor
   double d1=high[var]-Vn[member][var];
   double d2=Vn[member][var]-low[var];
   double d=Math.abs(Math.min(d1,d2));

   double R=gaussrandom(Vn[member][var],d);
   Vn[member][var] = R;
   //System.out.println("R="+R+"d1="+d1+"d2="+d2+"d="+d+"var = "+var+" "+Vn[member][var]);
   if(Vn[member][var]>high[var]) Vn[member][var]=high[var]-d*Math.random()*0.122561462;
   else if (Vn[member][var]<low[var]) Vn[member][var]=low[var]+d*Math.random()*0.122561462;;
          //after mutation recalculate the function value
   }
}



        public void evaluate(f_xj fi)
        {
          int mem;
          int j;
          double x[]=new double[NVARS];
          for(mem=0;mem<POPSIZE;mem++)
          {
          //System.out.println(Vn[mem].toString());
          if(getFitness(Vn[mem],fi) > getFitness(Vn[POPSIZE],fi))
            {
                  //Vn[POPSIZE]=Vn[mem];
            for(j=0;j<NVARS;j++)
              {
              Vn[POPSIZE][j]=Vn[mem][j];
                    }//end of for(j=0..
                  }//end of if
          }//end of for(mem=0;..
        }



        public void toString(genotype Pi[])
        {
        int mem;
        int PN=Pi.length;
        for(mem=0;mem<PN;mem++)
        {
        //list them
        System.out.println("Vn["+mem+"]="+Pi[mem].toString());
        }
        }

  public double[] getBest(double[] b,f_xj fi)
  {
  double aa[]=new double[NVARS+1];
  for(int i=0;i < NVARS;i++)
```

```
    {
    aa[i]=b[i];
            }
            aa[NVARS]=getFitness(b,fi);
  return aa;
  }

  public double getFitness(double []xi,f_xj fi)
  {
  return fi.func(xi);
  }

  public double[] calculate(double low[],double up[],f_xj fi,boolean report)
  {
    local=true;
    setVc(low,up,fi); //new random value
             t=0;
    do
    {
    do
    {   setVn(fi);
       evaluate(fi);
       if(getFitness(Vn[POPSIZE],fi) > getFitness(Vc,fi))
       {  setVc(Vn[POPSIZE], fi);
                if(getFitness(Vn[POPSIZE],fi) > getFitness(best,fi)) setBest(Vn[POPSIZE],fi);
       }
       else local=false;
    } while(local);
    if(report)
      System.out.println(Matrix.toString(getBest(best,fi)));
    t++;
    } while(t<MAX);
  return getBest(best,fi);
  }
}
```

Program 5.19-4 İterasyonlu tepe tırmanma – double sınıfını kullanan versiyon

```
 // iterated hill climbing
//
import java.io.*;
import javax.swing.JOptionPane;

class fb extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
// newton metodu minimumaçalışıyor bu yüzden
//- işaretitanımladık.
double ff;
double eps=1.0e-5;
double y=Math.sqrt(x[0]*x[0]+x[1]*x[1]+eps);
```

```
return Math.sin(y)/y; //maksimum testi
}
}

class OPO23
{
   public static void main(String args[]) throws IOException
   {
   int N=18;
         int POPSIZE=40;
   int MAXGENS=400;
   int NVARS=2;
   fb fbx=new fb();
   IHC xx=new IHC(POPSIZE,NVARS,N,20);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=-20.0;high[0]=20;
   low[1]=-20.0;high[1]=20;
   double a[]=xx.calculate(low,high,fbx,false);
   double x0[]=new double[NVARS];
   for(int i=0;i<NVARS;i++) x0[i]=a[i];
   String s="iterasyonlu tepe tırmanma metodu sonucu =\n"+Matrix.toStringT(a);
   String s2="iterasyonlu tepe tırmanma çok değişkenli fonksiyon optimizasyonu:";
   System.out.println(s2+s);
   }
}
```

Çıktı 5.19-2 iterasyonlu tepe tırmanma metodu çıktısı

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO23
iterasyonlu tepe tırmanma çok değişkenli fonksiyon optimizasyonu:iterasyonlu tepe tırmanma metodu sonucu =
      0.000000000000000000000000
      0.000000000000000000000000
      0.999998333334166700000000



> Terminated with exit code 0.
```

5.20 Global tesadüfi çözüm metodları : tavlama simulasyonu (simulated annealing)

Termodinamik analojiden elde edilen alternatif bir yaklaşım istatistiksel optimizasyona uygulanabilir. Bir metal ergime sıcaklığının üzerine ısıtıldığında atomları yüksek enerjili gelişigüzel harekete başlar. Sıcaklık düşürüldüğünde atomların hareketleri daha az enerjili hale dönüşür ve düşük eneji seviyesine kayar. En sonunda atomlar minimum enerji seviyesindeki hal durumlarını oluşturacak kristal yapılar halinde kendilerini organize ederler. Bu proses tavlama olarak bilinir. Tavlanmış metalin enerji durumunu Bolzman dağılımı gösterir.

$$p(e) = \alpha \exp(\frac{-e}{kT})$$

burada e>=0 enerji seviyesi, T sıcaklık, α=1/(kT), k boltzman sabiti p(e) olasılık dağılım fonksiyonudur. Tavlama doğal bir optimizasyon olayı olarak kabul edilebilir ve bu olay bilgisayar programlamasında taklit edilebilir. **$f(x_1,x_2,x_3,\ldots,x_n)$** fonksiyonunu minimize etme problemini göz önüne alalım. Bu tür bir model oluşturabilmek için T değerinin suni olarak oluşturulan bir sıcaklık faktörü olduğunu düşünelim. T'nin büyük bir değerinden optimizasyona

başlıyabiliriz(tavlamanın başında metalin sıcaklığı yüksektir). Başlangıç için $x^0=(x_1,x_2,x_3,\ldots,x_n)^0$ vektörünü alalım. Eğer x vektöründe Δx tesadüfi hareketini yapar isek, bu Δx, x noktası etrafında gauss değişimi ile dağılan bir tasadüfi dağılımla dağıldığını kabul edebiliriz . x^0 vektörüne komşu bir x^1 vektörünü $x^1=x^0+\Delta x$ şeklinde tanımlayarak yeni vektörü elde etmiş< oluyoruz. $\Delta f=f(x^1)-f(x^0)$ olarak bulunur. Gauss dağılımı

$$p(\Delta x_j)=\frac{1}{T\sqrt{2\pi}}\exp\left(\frac{-\Delta x_j}{2T^2}\right),\quad 1\le j\le n$$

eğer Δf<0 olursa bunun anlamı minimuma yaklaştık demektir ve yeni vektör x^1 yeni değer olarak kabul edilir. Eğer Δf>=0 ise bu otomatik olarak red edilmesi anlamına gelmez. Yine de kabul edilebilmesi mümkündür. Bunun için Boltzman dağılımına bakarız.

$$p(\Delta f)=\frac{1}{kT}\exp\left(\frac{-\Delta f}{kT}\right)$$

kabul ve red konusunda karar vermek için [0,1/(kT)] aralığında sabit dağılımlı bir tesadüfi sayı z oluştururuz. Eğer p(Δf) > z ise x^1 değeri yeni değer olarak kabul edilir. Değil ise reddedilir, ve yeni bir doğrultu oluşturmak için yeni bir Δx hesaplanır. Zaman içinde T sıcaklık fonksiyonu azaltılır. İterasyonun başlangıcında T değeri büyük olduğundan Δf>=0 ın kabul olasılığı büyüktür. T küçüldükçe arama Δf>=0 durumlarının kabulünde daha tutucu hale gelir. Bu prosesi bir algoritma halinde yazarsak :

```
Tavlama simulasyonu (simulated annealing) algoritması
başla
   t=0
   T sıcaklığının  değerini oluştur.
    v_c değişkenini tesadüfi olarak seç
    v_c için fonksiyonu hesapla
   while
       while
            v_c  noktası komşu bölgesinde v_n yeni vektörünü seç
            if f(v_c) < f(v_n)  then  v_c = v_n (yeni noktayı kabul et)
            else  if tesadüfi sayı(0-1) < exp[(f(v_n)-f(v_c))/T] then v_c = v_n (yeni noktayı f(v_c) >= f(v_n) olamasına
                                                                                 rağmen kabul et)
       until (çıkış şartı)
       t=t+1
    until (bitirme şartı)
bitir
```

Zamana bağlı olan T fonksiyonunun değerlendirilmesi çeşitli şekillerde olabilir. Birkaç örnek verecek olursak

$$T=\frac{T_0}{1+\log(1+t)}$$

$$T=T-\frac{\log(t_0)}{t\log(t)}$$

$$T=\frac{T_0}{1+t}$$

program olarak iki versiyon geliştirilmiştir. Birinci versiyonda yeni seçilen noktanın bir önceki noktanın komşu bölgesinde olmasını sağlamak için genetik algoritmada kullandığımız genotype1 sınıfından yararlandık (bit yapısında 1 biti tesadüfi olarak değiştirdik)

Program 5.20-1 Tavlama simulasyonu (simulated annealing) metodu (genotype1 mettodu ile)

```
İmport java.io.*;
import java.text.*;
import java.util.Locale;


public class simulatedannealing
{
   //  Maximizing a function
   // simulated annealing
   //  a version of simulated annealing statistical approach
   double T; //Temperature
   genotype1 Vc;
   genotype1 Vn[];
   genotype1 best;
   genotype1 oldbest;
   int MAX;//number
   int GLIMIT;//arka arkaya başarılı steplerin sayısı
   boolean local;
   int NVARS;//numberof variables
   int N;//number of bit representation for each variable
   double T0;
   double k,k0;//time index
   int POPSIZE;
   int BESTLIMIT;
   int bestcounter;

   //f_xj f;  // function to be avaluated by genetic algorithm

   public simulatedannealing(int iPOPSIZE,int iNVARS,int iN,double Ti)
   {
          T=Ti;
          T0=T;
   POPSIZE=iPOPSIZE;
   NVARS=iNVARS; // number of variables equation to solve
   MAX=20;
   N=iN;          // number of bits represent each variable for example 32
   Vn=new genotype1[POPSIZE+1];
   local=false;
   GLIMIT=10;
   BESTLIMIT=3;
   k=100.0;
   k0=k;
   bestcounter=0;
          }

          public void setVc(double low[],double up[],f_xj fi)
          {
```

```
        //select originalgene andomly
        Vc=new genotype1(NVARS,low,up,N,fi);
        }
        public void setBest(genotype1 gi,f_xj fi)
        {
        //select originalgene andomly
        best=new genotype1(gi,fi);
        }
        public void setoldBest(f_xj fi)
        {
        //select originalgene andomly
        oldbest=new genotype1(best,fi);
        }

  public void setVc(genotype1 gi,f_xj fi)
        {
        //select original gene from other gene
        Vc=new genotype1(gi,fi);
        }
        public void setVn(f_xj fi)
        {//select neighbouring genes similar to the original one
        for(int i=0;i<POPSIZE+1;i++)
          { Vn[i]=new genotype1(Vc,fi);}
           mutate(fi);
        }

        public void evaluate(f_xj fi)
        {
          int mem;
          int j;
          double x[]=new double[NVARS];
          for(mem=0;mem<POPSIZE;mem++)
          {
          //System.out.println(Vn[mem].toString());
          Vn[mem].setFitness(fi);
          if(Vn[mem].getFitness() > Vn[POPSIZE].getFitness())
            {
                  Vn[POPSIZE]=Vn[mem];
            for(j=0;j<NVARS;j++)
              {
              Vn[POPSIZE].G[j].copyGene(Vn[mem].G[j]);
                    }//end of for(j=0..
                  }//end of if
          }//end of for(mem=0;..
        }

        public void setRFCF(genotype1 Pi[])
        {
        //calculates relative and cumulative fitness functions
        int mem;
        int PN=Pi.length;
        double sum=0.0;
```

```
        //total fitness of Vn
        for(mem=0;mem<PN;mem++)
          {sum+=Pi[mem].Fitness;}
        //System.out.println("sum="+sum);
        //calculate relative fitness of each genotype1
        for(mem=0;mem<PN;mem++)
          {Pi[mem].setRFitness(sum);}
        //calculate cumulative fitness
        Pi[0].setCFitness(Pi[0].getRFitness());
 for(mem=1;mem<PN;mem++)
          {Pi[mem].setCFitness(Pi[mem-1].getCFitness()+Pi[mem].getRFitness());
          }
        }

        public void toString(genotype1 Pi[])
        {
        int mem;
        int PN=Pi.length;
        for(mem=0;mem<PN;mem++)
        {
        //list them
        System.out.println("Vn["+mem+"]="+Pi[mem].toString());
        }
        }


public void mutate(f_xj fi)
{
int i;
double lbound,hbound;
int nmutations;
int member;
int var;
nmutations=(int)((double)(POPSIZE*NVARS*N));
for(i=0;i<nmutations;i++)
  {
  member=(int)(Math.random()*POPSIZE);
  var=(int)(Math.random()*NVARS);
  //replace the old value with a new mutated one
  Vn[member].G[var].mutate();
        //after mutation recalculate the function value
  Vn[member].setFitness(fi);
  }
}

public double[] getBest(genotype1 b)
{
double aa[]=new double[NVARS+1];
for(int i=0;i < NVARS;i++)
 {
 aa[i]=b.getGene(i);
        }
```

```
            aa[NVARS]=b.getFitness();
  return aa;
  }
  public double[] calculate(double low[],double up[],f_xj fi,boolean report)
  {
   int good_swap=0;
            setVc(low,up,fi);// new set of random selection in each time
   setBest(Vc,fi);
            do
   {
            setoldBest(fi);
   do
   {   setVn(fi);
      evaluate(fi);
      if(Vn[POPSIZE].getFitness() > Vc.getFitness())
      {  setVc(Vn[POPSIZE], fi);
                  if(Vn[POPSIZE].getFitness() > best.getFitness())
                     {setBest(Vn[POPSIZE],fi); }
                      if(oldbest.getFitness() == best.getFitness()) bestcounter++;
                  if(bestcounter>BESTLIMIT) break;
                  good_swap++;
      }
      else if(Math.random()<Math.exp((Vn[0].getFitness()-Vc.getFitness())/T))
      {   setVc(Vn[POPSIZE],fi);}
    if(report)
      System.out.println(Matrix.toString(getBest(best)));
   } while(good_swap<GLIMIT);
   k+=1.0;
   double lk=Math.log(k);
   T=T-Math.log(k0)/(k*lk*lk);
   setVc(low,up,fi); //new random value

   } while(k<MAX);
  return getBest(best);
  }
}
```

Program 5.20-2 Tavlama simulasyonu çıktı programı metodu (genotype1 mettodu ile)

```
// simulated annealing
//
import java.io.*;
import javax.swing.JOptionPane;


class fa extends f_xj
{
//çözümü istenen fonksiyon
 double func(double x[])
 {
   double ff= 21.5+x[0]*Math.sin(4.0*Math.PI*x[0])+x[1]*Math.sin(20.0*Math.PI*x[1]);
   return ff; //maksimum testi
}
```

```
}

class fb extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
// newton metodu minimumaçalışıyor bu yüzden
//- işaretitanımladık.
double ff;
fa fx=new fa();
return -fx.func(x); //maksimum testi
}
}

class OPO21
{
   public static void main(String args[]) throws IOException
   {
   int N=18;
           int POPSIZE=1;
   int MAXGENS=400;
   int NVARS=2;
   fa fax=new fa();
   fb fbx=new fb();
   simulatedannealing xx=new simulatedannealing(POPSIZE,NVARS,N,20);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=-3.0;high[0]=12.1;
   low[1]=4.1;high[1]=5.8;
   double a[]=xx.calculate(low,high,fax,false);
   double x0[]=new double[NVARS];
   for(int i=0;i<NVARS;i++) x0[i]=a[i];
   String s="simulated annealing (tavlama simulasyonu)metodu sonucu =\n"+Matrix.toStringT(a);
    String s2="simulated annealing çok değişkenli fonksiyon optimizasyonu:";
    System.out.println(s2+s);
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO21
simulated annealing çok değişkenli fonksiyon optimizasyonu:simulated annealing (tavlama
simulasyonu)metodu sonucu =
      11.610554544656923000000000
       5.725915244732836000000000
      38.636233636633500000000000



> Terminated with exit code 0.
```

İkinci program versiyonumuzda ise double sayıları gauss tesadüfi sayılarını kullanarak komşu bölgeleri oluşturduk.

Program 5.20-3 Tavlama simulasyonu(Simulated annealing) programı metodu (double vektör ile)

```
import java.io.*;
import java.text.*;
import java.util.Locale;


public class SA
{           // Maximizing a function
   //simulated annealing
   // a version of simulated annealing statistical approach
   double T; //Temperature
   double Vc[];
   double Vn[][];
   double best[];
   int MAX;//number
   int GLIMIT;//arka arkaya başarılı steplerin sayısı
   boolean local;
   int NVARS;//numberof variables
   int N;//number of bit representation for each variable
   double T0;
   double k,k0;//time index
   int POPSIZE;
   double low[];
   double high[];

   //f_xj f;  // function to be avaluated by genetic algorithm

   public SA(int iPOPSIZE,int iNVARS,int iN,double Ti)
   {
           T=Ti;
           T0=T;
   POPSIZE=iPOPSIZE;
   NVARS=iNVARS; // number of variables equation to solve
   MAX=20;
   N=iN;           // number of bits represent each variable for example 32
   local=false;
   GLIMIT=30;
   k=100.0;
   k0=k;
   Vc=new double[NVARS];
   Vn=new double[POPSIZE+1][NVARS];
   best=new double[NVARS];
   low=new double[NVARS];
   high=new double[NVARS];
           }

public static double  gaussrandom(double xort,double r)
{ int iset=0;
 double gset=0;
 double fac,rsq,v1,v2;
 double aret=0;
 if(iset==0)
```

```
{
  do{
            v1=2.0*Math.random()-1.0;
            v2=2.0*Math.random()-1.0;
            rsq=v1*v1+v2*v2;
    }while(rsq>=1.0 || rsq==0.0);
  fac=Math.sqrt(-2.0*Math.log(rsq)/rsq);
  gset=v1*fac;
  iset=1;
  aret=v2*fac;
}
else
{
iset=0;
aret=gset;
}
return aret/4.0*r+xort;
}


            public void setVc(double ilow[],double ihigh[],f_xj fi)
            {

            //select original gene randomly
            for(int i=0;i<NVARS;i++)
              {low[i]=ilow[i];high[i]=ihigh[i];Vc[i]=(high[i]-low[i])*Math.random();}
   }

            public void setBest(double gi[],f_xj fi)
            {
            for(int i=0;i<NVARS;i++)
              {best[i]=gi[i];}
            }

   public void setVc(double gi[],f_xj fi)
            {
            //select original gene from other gene
                        for(int i=0;i<NVARS;i++)
              {Vc[i]=gi[i];}
            }

            public void setVn(f_xj fi)
            {//select neighbouring genes similar to the original one
            for(int i=0;i<POPSIZE+1;i++)
              { for(int j=0;j<NVARS;j++)
              {Vn[i][j]=Vc[i];}}
              mutate(fi);
            }

  public void mutate(f_xj fi)
  {
  int i;
```

```
  double lbound,hbound;
  int nmutations;
  int member;
  int var;
  nmutations=(int)((double)(POPSIZE*NVARS*N));
  for(i=0;i<nmutations;i++)
    {
    member=(int)(Math.random()*POPSIZE);
    var=(int)(Math.random()*NVARS);
    //replace the old value with a new mutated one
    //neigboring factor
    double d1=high[var]-Vn[member][var];
    double d2=Vn[member][var]-low[var];
    double d=Math.min(d1,d2);
    double R=gaussrandom(Vn[member][var],d*Math.random());
    Vn[member][var]= R;
    if(Vn[member][var]>high[var]) Vn[member][var]=high[var]-Math.random()*0.22561462;
    else if (Vn[member][var]<low[var]) Vn[member][var]=low[var]+Math.random()*0.22561462;;
          //after mutation recalculate the function value
    }
  }


        public void evaluate(f_xj fi)
        {
          int mem;
          int j;
          double x[]=new double[NVARS];
          for(mem=0;mem<POPSIZE;mem++)
          {
          //System.out.println(Vn[mem].toString());
          if(getFitness(Vn[mem],fi) > getFitness(Vn[POPSIZE],fi))
            {
                  //Vn[POPSIZE]=Vn[mem];
            for(j=0;j<NVARS;j++)
              {
              Vn[POPSIZE][j]=Vn[mem][j];
                    }//end of for(j=0..
                  }//end of if
          }//end of for(mem=0;..
        }


        public void toString(genotype Pi[])
        {
        int mem;
        int PN=Pi.length;
        for(mem=0;mem<PN;mem++)
        {
        //list them
        System.out.println("Vn["+mem+"]="+Pi[mem].toString());
        }
```

```
            }

  public double[] getBest(double[] b,f_xj fi)
  {
  double aa[]=new double[NVARS+1];
  for(int i=0;i < NVARS;i++)
   {
   aa[i]=b[i];
            }
            aa[NVARS]=getFitness(b,fi);
  return aa;
  }

  public double getFitness(double []xi,f_xj fi)
  {
  return fi.func(xi);
  }

  public double[] calculate(double low[],double up[],f_xj fi,boolean report)
  {
   int good_swap=0;
            setVc(low,up,fi);// new set of random selection in each time
   setBest(Vc,fi);
            do
   {
   do
   {   setVn(fi);
       evaluate(fi);
       if(getFitness(Vn[POPSIZE],fi) > getFitness(Vc,fi))
       {  setVc(Vn[POPSIZE], fi);
                  if(getFitness(Vn[POPSIZE],fi) > getFitness(best,fi)) setBest(Vn[POPSIZE],fi);
                  good_swap++;
       }
       else if(Math.random()<Math.exp((getFitness(Vn[0],fi)-getFitness(Vc,fi))/(k*T)))
       {   setVc(Vn[POPSIZE],fi);}
   if(report)
      System.out.println(Matrix.toString(getBest(best,fi)));
   } while(good_swap<GLIMIT);
   k+=1.0;
   double lk=Math.log(k);
   T=T-Math.log(k0)/(k*lk*lk);
   setVc(low,up,fi); //new random value
   } while(k<MAX);
  return getBest(best,fi);
  }
}
```

Program 5.20-4 Tavlama simulasyonu(Simulated annealing) çıktı metodu (double vektör ile)

```
// simulated annealing
//
import java.io.*;
import javax.swing.JOptionPane;
```

```
class fa extends f_xj
{
//çözümü istenen fonksiyon
 double func(double x[])
 {
   double ff= 21.5+x[0]*Math.sin(4.0*Math.PI*x[0])+x[1]*Math.sin(20.0*Math.PI*x[1]);
   return ff; //maksimum testi
}
}

class fb extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
// newton metodu minimumaçalışıyor bu yüzden
//- işaretitanımladık.
double ff;
fa fx=new fa();
return -fx.func(x); //maksimum testi
}
}

class OPO22
{
   public static void main(String args[]) throws IOException
   {
   int N=18;
           int POPSIZE=1;
   int MAXGENS=400;
   int NVARS=2;
   fa fax=new fa();
   fb fbx=new fb();
   SA xx=new SA(POPSIZE,NVARS,N,20);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=-3.0;high[0]=12.1;
   low[1]=4.1;high[1]=5.8;
   double a[]=xx.calculate(low,high,fax,false);
   double x0[]=new double[NVARS];
   for(int i=0;i<NVARS;i++) x0[i]=a[i];
   String s="simulated annealing (tavlama simulasyonu)metodu sonucu =\n"+Matrix.toStringT(a);
   String s2="simulated annealing ve newton çok değişkenli fonksiyon optimizasyonun birlikte kullanımı:";
       System.out.println(s2+s);
   }
}
```

Çıktı 5.20-3 Tavlama simulasyonu(Simulated annealing) programı metodu (double vektör ile)

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO22
```

```
simulated annealing  optimizasyonu :simulated annealing (tavlama simulasyonu)metodu sonucu =
      0.46749058495418630000000000
      1.59380038365824970000000000
      2.61099919706452400000000000
     -0.59988603011306500000000000


> Terminated with exit code 0.
```

5.21 Global tesadüfi çözüm metodları : Ateşböceği algoritması

Yaz akşamlarında ateşböceklerinin parıldamasını görmüşüzdür. Bu etki biolojik ışık oluşturma prosesi dediğimiz bir biokimyasal reaksiyon sonucunda oluşturulur. Bu olgunun kesin sebebi henüz tartışma halindedir, ancak uzmanlar uzmanlar büyük olasılıkla eş bulma prosesinin bir parçası olduğunu düşünmektedirler. Işıltılrın ritmi, zamanlaması eşlerin bir araya gelmesini sağlar. Belli bir mesafedeki ışık şiddetinin aradaki mesafenin karesiyle değiştiğini biliyoruz. Yani mesafe arttıkça ışık şiddeti (parlaklık) azalmaktadır. Buna ek olarak hava molekülleride ışığı emdiği için mesafeyla ışık şiddetinin azalması daha da fazla olabilir. Bu temel prensibi kullanarak ateş böcekleri birbirlerini bulabilirler.

(bolum5_028.jpg,Resimaltı **:ışık saçan bir ateşböceği**)

Aynı temel prensibi bizde matematik optimizasyonu için kullanabiliriz. Xin-She Yang [52] ateş böceği hareketlerinden esinlenerek bir optimizasyon algoritması geliştirmiştir. Algoritma ateşböceklerinin tek cinsiyetli olduğunu varsaymaktadır. Bunun anlamı herhangi bir ateş böceğinin ışık yanıp sönmesi diğer tüm ateşböceklerini kendine çekecektir. Tabiki burada bilgisayar kodundaki ateşböceklerinden bahsediyoruz. Çekim gücü parlaklığın fonksiyonu olarak değişecektir, ve parlaklık aradaki mesafenin fonksiyonudur. Daha az parlak ateş böcekleri daha parlak olanlara doğru uçacaklardır. Eğer etrafta parlak ateşböceği yoksa tesadüfi olarak hareket edeceklerdir. Optimizasyon prosesinde ateşböceklerinin parlaklığını optimuma ulaşmasını istediğimiz fonksiyon belirler. Bu kurallara göre aşağıdaki algoritma geliştirilmiştir.

Tablo 5.21.1 Ateşböceği simülasyon algoritması

Optimizasyon fonksiyonu $f(X)$, $X = (x_0, ...,x_n)$
Ateşböceklerinin ilk dağılımını tesadüfi olarak oluştur X^i $(i = 1,2, ...,m)$
X^i noktasındaki ışık parlaklığı Ii $f(X^i)$*fonksiyonu tarafından belirlenir*
Havanın ışığı absorbe etme katsayısını tanımla γ
while $(t <$MaxGeneration$)$
for $i = 1 : n$ tüm ateşböcekleri için
  for $j = 1 : n$ tüm ateş böcekleri için (iç döngü)
    if $(Ii < Ij)$, ateşböceği *i 'yi* ateşböceği *j'ye doğru hareket ettir;* end i f
    çekicilik faktörü r yi mesafeyle değiştir.

$$\beta = \beta_0 e^{-\gamma d^2} \text{ or } \beta = \frac{\beta_0}{1+\gamma d^2}$$

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (x_{i,k} - x_{i,k})^2}$$

    Yeni çözümleri değerlendir ve ışık parlaklığını yenile

$$X^i = (1-\beta)X^i + \beta X^j + \alpha\,(gaussrandom() - \frac{1}{2})$$

end for j

end for i

ateş böceklerini başarılarına göre sırala ve en iyi çözümü bul *gt*

end while

sonuçları değerlendir

Bu prensibi kullanan bir java programramı geliştirilmiştir. Program kodu altta verilmiştir. Kod Nelder-Mead koduyla birlikte kullanılarak daha etkin hale getirilmiştir.

Program 5.21.1 Ateşböceği simülasyonu, Nelder-Mead metoduyla entegre edilmiştir.

```
import java.io.*;
import java.text.*;
import java.util.Locale;

abstract class f_xj
{
  abstract double func(double x[]);
}

public class atesbocegi
{
double gamma,alpha,beta;
double alpha0,alphan;
int m;
int n;
// m ateşböceği sayısı
// n her ateşböceği için değişken sayısı
// I0 kaynaktaki ışık parlaklığı
// gamma hava ışık absorbsiyon katsayısı
// alpha tesadügi adım büyüklüğü
// beta çekim faktörü
double x[][];
double I0[];
double ymin;
double xmin[];
int imin;
int ngeneration;
public atesbocegi(f_xj ff,int mi,int ni, double xxmin[],double xxmax[],double gammai,double alpha0i,double
alphani,int ngenerationi)
{
//ilk ateşböceği nüfusunu yarat
n=ni;
m=mi;
x=new double[m][n];
I0=new double[m];
xmin=new double[n];
gamma=gammai;
alpha0=alpha0i;
alphan=alphani;
ngeneration=ngenerationi;
ymin=1.0e50;

//ilk ateşböceği nüfusunu yaratıldı
int l=0;
double d=0;
double t=0;
while(l<ngeneration)
{
```

```
// nufusun minimum değeri veren üyesini sapta
for(int i=0;i<m;i++)
{for(int j=0;j<n;j++)
  {x[i][j]=xxmin[j]+Math.random()*(xxmax[j]-xxmin[j]);}
  I0[i]=ff.func(x[i]);
  if(I0[i]<ymin)
    {ymin=I0[i];imin=i;
    for(int k=0;k<n;k++)
    {xmin[k]=x[i][k];}
    alpha=alpha0;
    };
}
l++;
double fi=0,fj=0;
for(int i=0;i<m;i++)
{ //fi=ff.func(x[i]);
  for(int j=0;j<n;j++)
  {
    //fj=ff.func(x[j]);
    if(I0[i]<I0[j])
    {
    beta=attractiveness(i,j);
    for(int k=0;k<n;k++)
    {   //en iyiyi sakla, gerisini değiştir
                if(i!=imin)
       { x[i][k]=(1.0-beta)*x[i][k]+beta*x[j][k]+alpha*(Math.random()-0.5);} }
    }
    else
    {
                for(int k=0;k<n;k++)
       {  if(i!=imin)
                  { x[i][k]=x[i][k]+alpha*(Math.random()-0.5); }
                }
    }
  }

}

    //alpha zaman içinde değişir
    //(daha küçük bir değere doğru gider)
    alpha=alphan+(alpha0-alphan)*Math.exp(-t);
    t=0.1*l;
}
}

public double distance(int i, int j)
{
double d=0;
for(int k=0;k<n;k++)
{d+=(x[i][k]-x[j][k])*(x[i][k]-x[j][k]);}
d=Math.sqrt(d);
return d;
}

public double attractiveness(int i,int j)
{   double d=distance(i,j);
            return I0[i]/(1+gamma*d*d);
            //return I0[i]*Math.exp(-gamma*d*d);
}
```

```
public String toString()
{ String s="ymin = "+ymin+"\nxmin = ";
  for(int k=0;k<n;k++)
  {s+=xmin[k]+" ";}
  return s;
}
public String toString(f_xj ff,double xx[])
{ String s="ymin = "+ff.func(xx)+"\nxmin = ";
  for(int k=0;k<n;k++)
  {s+=xx[k]+" ";}
  return s;
}
public double[] xmin()
{
  return xmin;
}

public static double[] nelder(f_xj fnelder,double a[],double da[],int maxiteration,double tolerance,int printlist)
{
            int i,j;
double x[][]=new double[a.length+1][a.length];
double p[][]=new double[a.length+1][a.length+1];
for(i=0;i<x.length;i++)
   {for(j=0;j<x[0].length;j++)
     {if(i==j){x[i][j]=a[j]+da[j];p[i][j]=x[i][j];}
      else    {x[i][j]=a[j];p[i][j]=x[i][j]; }
     }
     p[i][j] = fnelder.func(p[i]);
   }
        int NDIMS = x.length-1;
        int NPTS = x.length;
        int FUNC = NDIMS;
        int ncalls = 0;
          double z[]=new double[NDIMS];
          double best = 1E99;
          int iter=0;
          for (iter=1; iter<maxiteration; iter++)
          {
             int  ilo=0, ihi=0, inhi = -1; // -1 means missing
             double flo = p[0][FUNC];
             double fhi = flo;
             double pavg,sterr;
             for (i=1; i<NPTS; i++)
             {
               if (p[i][FUNC] < flo)
                 {flo=p[i][FUNC]; ilo=i;}
               if (p[i][FUNC] > fhi)
                 {fhi=p[i][FUNC]; ihi=i;}
             }
             double fnhi = flo;
             inhi = ilo;
             for (i=0; i<NPTS; i++)
               if ((i != ihi) && (p[i][FUNC] > fnhi))
                 {fnhi=p[i][FUNC]; inhi=i;}
             ////////// exit criteria //////////////
             if ((iter % 4*NDIMS) == 0)
             {
                // calculate the avarage (including maximum value)
                pavg=0;
                for(i=0;i<NPTS;i++)
```

```
      pavg+=p[i][FUNC];
    pavg/=NPTS;
    double tot=0;
    if(printlist!=0)
    {  System.out.print(iter);
       for (j=0; j<=NDIMS; j++)
          { System.out.print(p[ilo][j]+" ");}
       System.out.println("");
    }
    for(i=0;i<NPTS;i++)
    { tot=(p[i][FUNC]-pavg)*(p[i][FUNC]-pavg);}
    sterr=Math.sqrt(tot/NPTS);
    //if(sterr < tolerance)
      { for (j=0; j<NDIMS; j++)
          { z[j]=p[ilo][j];}
        //break;
      }
    best = p[ilo][FUNC];
}

///// calculate avarage without maximum value //////

double ave[] = new double[NDIMS];
for (j=0; j<NDIMS; j++)
  ave[j] = 0;
for (i=0; i<NPTS; i++)
  if (i != ihi)
    for (j=0; j<NDIMS; j++)
      ave[j] += p[i][j];
for (j=0; j<NDIMS; j++)
   ave[j] /= (NPTS-1);


///////// reflect ////////////////

double r[] = new double[NDIMS];
for (j=0; j<NDIMS; j++)
  r[j] = 2*ave[j] - p[ihi][j];
double fr = fnelder.func(r);

if ((flo <= fr) && (fr < fnhi))  // in zone: accept
{
    for (j=0; j<NDIMS; j++)
      p[ihi][j] = r[j];
    p[ihi][FUNC] = fr;
    continue;
}

if (fr < flo)  //// expand
{
    double e[] = new double[NDIMS];
    for (j=0; j<NDIMS; j++)
      e[j] = 3*ave[j] - 2*p[ihi][j];
    double fe = fnelder.func(e);
    if (fe < fr)
    {
       for (j=0; j<NDIMS; j++)
         p[ihi][j] = e[j];
       p[ihi][FUNC] = fe;
       continue;
```

```
    }
    else
    {
      for (j=0; j<NDIMS; j++)
        p[ihi][j] = r[j];
      p[ihi][FUNC] = fr;
      continue;
    }
  }

  ////////////// shrink:

  if (fr < fhi)
  {
    double c[] = new double[NDIMS];
    for (j=0; j<NDIMS; j++)
      c[j] = 1.5*ave[j] - 0.5*p[ihi][j];
    double fc = fnelder.func(c);
    if (fc <= fr)
    {
      for (j=0; j<NDIMS; j++)
        p[ihi][j] = c[j];
      p[ihi][FUNC] = fc;
      continue;
    }
    else   /////// daralt
    {
      for (i=0; i<NPTS; i++)
        if (i != ilo)
        {
          for (j=0; j<NDIMS; j++)
            p[i][j] = 0.5*p[ilo][j] + 0.5*p[i][j];
          p[i][FUNC] = fnelder.func(p[i]);
        }
      continue;
    }
  }

  if (fr >= fhi)   ///
  {
    double cc[] = new double[NDIMS];
    for (j=0; j<NDIMS; j++)
      cc[j] = 0.5*ave[j] + 0.5*p[ihi][j];
    double fcc = fnelder.func(cc);
    if (fcc < fhi)
    {
      for (j=0; j<NDIMS; j++)
        p[ihi][j] = cc[j];
      p[ihi][FUNC] = fcc;
      continue;
    }
    else    /////////
    {
      for (i=0; i<NPTS; i++)
        if (i != ilo)
        {
          for (j=0; j<NDIMS; j++)
            p[i][j] = 0.5*p[ilo][j] + 0.5*p[i][j];
          p[i][FUNC] = fnelder.func(p[i]);
        }
```

```
                }
            }
        }
     return z;
}
    public static double[] nelder(f_xj fnelder,double a[],double da[],double tolerance)
    {return nelder(fnelder,a,da,500,tolerance,0);}

    public static double[] nelder(f_xj fnelder,double a[],double da[])
    {return nelder(fnelder,a,da,500,1.0e-10,0);}

    public static double[] nelder(f_xj fnelder,double a[])
    {
                double [] da=new double[a.length];
        for(int i=0;i<a.length;i++) da[i]=0.1*a[i];
        return nelder(fnelder,a,da);
    }
}
```

Program 5.21.2 Ateşböceği simülasyonu test programı

```
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
class fa1 extends f_xj//Rosenbrock function
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=(1.0-x[0])*(1.0-x[0])+100*(x[1]-x[0]*x[0])*(x[1]-x[0]*x[0]);
return ff;
}
}

public class atesbocegitesti
{
public static void main(String arg[])
{
fa1 ff=new fa1();
double xmin[]={0,0};
double xmax[]={2.0,2.0};
double gamma=1.0;
double alpha0=0.5;
double alphan=0.01;
atesbocegi  ffly=new atesbocegi(ff,1000,2,xmin,xmax,gamma,alpha0,alphan,1000);
double x0[]=ffly.xmin();
double r1[]= atesbocegi.nelder(ff,x0);
String s="atesbocegi search : \n"+ffly.toString()+"\n Nelder-Mead: \n"+ffly.toString(ff,r1);
    String s2="atesbocegi-Nelder-Mead simplex optimizasyon metodu ";
    System.out.println(s2+s);
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" atesbocegitesti
atesbocegi-Nelder-Mead simplex optimizasyon metodu atesbocegi search :
ymin = 1.5798355951986852E-7
xmin = 0.9996026695684972 0.9992065557233554
```

```
 Nelder-Mead:
ymin = 0.0
xmin = 1.0 1.0

> Terminated with exit code 0.
```

Ateşböceği simülasyon programıyla geometrik optimizasyonlar için çözümü zor olan bazı özel fonksiyonları test edelim. İlk örneğimiz Michaelewicz fonksiyonu olsun. Bu fonksiyonun genel şekli:

$f(x) = -\sum_{i=0}^{n}\left[\sin(x_i)\left[\sin\left(\frac{(i+1)x_i^2}{\pi}\right)\right]^{2m}\right]$ şeklindedir. Biz n=2 ve m=10 için deneyelim

```
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class fa extends f_xj
{
int n;
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0;
int n=x.length;
for(int i=0;i<n;i++)
{ff+=-Math.sin(x[i])*Math.pow(Math.sin((i+1)*x[i]*x[i]/Math.PI),20);}
return ff; //maksimum testi
}
}

public class atesbocegitesti1
{
public static void main(String arg[])
{
fa ff=new fa();
double xmin[]={0,0};
double xmax[]={2.5,2.5};
double gamma=1.0;
double alpha0=0.5;
double alphan=0.01;
atesbocegi  ffly=new atesbocegi(ff,1000,2,xmin,xmax,gamma,alpha0,alphan,1000);
double x0[]=ffly.xmin();
double r1[]= atesbocegi.nelder(ff,x0);
String s="atesbocegi search : \n"+ffly.toString()+"\n Nelder-Mead: \n"+ffly.toString(ff,r1);
    String s2="atesbocegi-Nelder-Mead simplex optimizasyon metodu ";
    System.out.println(s2+s);
}
}
```

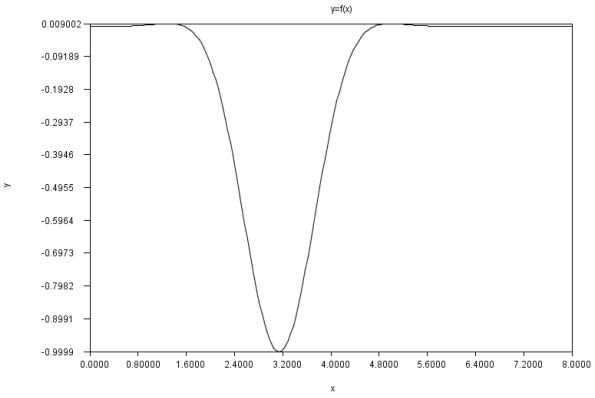
```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" atesbocegitesti1
atesbocegi-Nelder-Mead simplex optimizasyon metodu atesbocegi search :
ymin = -1.8012832380012231
xmin = 2.2021437805760473 1.570281169861501
 Nelder-Mead:
ymin = -1.8013034100985534
xmin = 2.2029055181769603 1.5707963290643197
```

```
> Terminated with exit code 0
```

Diğer ilginç bir test fonksiyonu Easom fonksiyonudur. $f(x) = \cos(x)e^{-(x-\pi)^2}$ Bu fonksiyonun tek bir minimumu olup x=π, y=f(x)=1noktasındadır.

```
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
class fa extends f_xj
{
int n;
public double func(double x[])
{

//çözümü istenen fonksiyon
double ff=Math.cos(x[0])*Math.exp(-(x[0]-Math.PI)*(x[0]-Math.PI));
return ff; //maksimum testi
}
}


public class atesbocegitesti2
{
public static void main(String arg[])
{
fa ff=new fa();
double xmin[]={0};
double xmax[]={4.0};
double gamma=1.0;
double alpha0=0.5;
double alphan=0.01;
atesbocegi  ffly=new atesbocegi(ff,1000,1,xmin,xmax,gamma,alpha0,alphan,1000);
double x0[]=ffly.xmin();
double r1[]= atesbocegi.nelder(ff,x0);
String s="atesbocegi search : \n"+ffly.toString()+"\n Nelder-Mead: \n"+ffly.toString(ff,r1);
    String s2="atesbocegi-Nelder-Mead simplex optimizasyon metodu ";
    System.out.println(s2+s);
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" atesbocegitesti2
atesbocegi-Nelder-Mead simplex optimizasyon metodu atesbocegi search :
ymin = -0.9999999999896446
xmin = 3.1415900261204532
 Nelder-Mead:
ymin = -1.0
xmin = 3.1415926570301265

> Terminated with exit code 0.
```



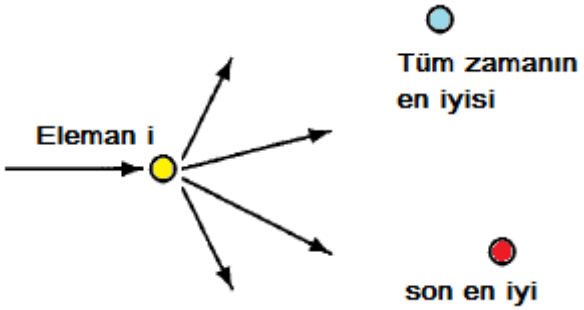
(bolum5_029.jpg,Resimaltı **:Easom fonksiyonu**)

Görüldüğü gibi metodumuz, kompleks test fonksiyonlarını başarı ile çözmektedir.

5.22 Global tesadüfi çözüm metodları : sürü elemanları algoritması

Sürü elemanları algoritması (Particle Swarm Algorithm-PSO) Kennedy ve Eberhart (1995) tarafından doğadaki kuş ve balık sürülerinin davranışlarından esinlenerek oluşturulmuştur. PSO algoritmasında sürü içindeki elemanların hareketleri izlenir. Bu hareketler temelde hem tesadüfi hem de sürüye uyum gösterme amaçlı olarak kararlı bir davranış sergiler. Sürüdeki her eleman o andaki en iyi konum ve yakın geçmişteki (tüm anlardaki) en iyi konumun cazibesiyle hareket ederler. Sürüdeki her elemanın aynı zamanda tesadüfi hareket etme olasılığı da mevcuttur. Herhangi bir eleman o ana kadar bulunan en iyi konumdan daha iyi bir konum yakaladığında bu konum en iyi konum olarak update edilir. Sürünün amacı tüm iyi konumlar arasından en iyi konumu yakalamaktır. Bu eylem artık daha iyi bir konum yakalamak mümkün olmayıncaya kadar sürer.

(bolum5_030.jpg,Resimaltı **:Balık sürüsünde balıkların toplu olarak hareketi**)



(bolum5_031.jpg,Resimaltı **:sürüdeki her eleman tüm zamanların en iyi konumu, en son iyi konum ve tesadüfi hareket çekim alanlarının etkisiyle hareket eder**)

Sürü algoritması Tablo 5.22.1 de verilmiştir. Eğer x_i ve v_i i'inci sürü elemanının pozisyon ve hız vektörleri ise, yeni hız vektörü

$$v_i^{t+1} = v_i^t + \alpha * \varepsilon_1 * (g^* - x_i^t) + \beta * \varepsilon_2 * (x_i^* - x_i^t)$$

Denklemiyle belirlenir. Buradaki ε_1 ve ε_2 , 0 ile 1 arasındaki tesadüfi sayılardır, g^* tüm zamanların en iyi konumu ve x_i^* en iyi son konumdur. α ve β parametreleri öğrenme parametreleri eya ivme sabitleri adını alır. Elemanların ilk başlama hızlarını sıfır olarak kabul edebiliriz, $v_i^{t=0} = 0$ veya tesadüfi olarak saptayarak başlarız . Elemanların yeni pozisyonları her birim zamanlık adım için

$x_i^{t+1} = x_i^t + v_i^{t+1}$. Formülüyle belirlenebilir. v_i değeri olarak herhangi bir değer alınabilirsede elemanların maximum bir hızla $[0, v_{max}]$ sınırlandırılmış olmalarında yarar vardır. PSO algoritmalarının bir çok versiyonları varsada ivmelendirilmiş PSO adını verdiğimiz versiyonu en çok kullanılanlardan biridir. Bu versiyonda bir önceki hız vektörü belli bir katsayı ile yeni vektörü etkiler.

$$v_i^{t+1} = \theta v_i^t + \alpha * \varepsilon_1 * (g^* - x_i^t) + \beta * \varepsilon_2 * (x_i^* - x_i^t)$$

Bu etki parçacıklar için suni bir kütle tanımlaması yapma gibi bir etkidir.

Table 5.22.1 Sürü elemanları algoritması (PSO)

Optimizasyon fonksiyonu $f(P_i)$, $P_i = (x_0, ...,x_m)_i$ i'inci eleman için (toplam n eleman) ve m değişken boyutu mevcuttur.

Eleman konumları P_i ve hızları V_i n eleman ve m boyut için tesadüfi olarak sapta

Hız vektörü formülü :

$$V_i^{t+1} = \theta V_i^t + \alpha * random() * (GP\min - x_i^t) + \beta * random() * (P\min - x_i^t)$$

Burada θ atalet fonksiyonu(0 ile 1 arasında tesadüfi bir sayı)

α ve β öğrenim katsayıları veya ivme sabitleri

$\alpha = \alpha_0 \gamma^t$ parametresi için: $0 < \gamma < 1 \quad \alpha_0 \cong 0.5..1$ alınır

t sanal zamandır

Tüm set taranarak tüm zamanların en iyi değeri bulunur GPmin from min{ $f(P_0)…f(P_n)$} (t = 0)

while (durdurma kriteri)

t = t + 1 (sanal zaman sayıcısı her adımda bir sanal zaman ilerler)

toplam m eleman ve m değişken boyutu için döngü

Yeni hız vektörü oluştur V_i^{t+1}

Yeni eleman konumu belirle $P_i^{t+1} = P_i^t + V_i^{t+1}$

Yeni konumdaki fonksiyon değerlerini hesapla

Her eleman için en son en iyi değeri sapta Pmin

Her eleman için tüm zamanların en iyi değerini sapta GPmin

Döngü sonu

Dış while döngüsü sonu

Sonuçları çıktı olarak ver Pmin and GPmin

Aşağıda geliştirdiğimiz java kodu ve örnek problemler verilmiştir.

Program 5.22.1 sürü elemanları algoritması

```
 import java.io.*;
import java.text.*;
import java.util.Locale;

abstract class f_xj
{
  abstract double func(double x[]);
}

public class suruelemanlari
{
int m;
int n;
double alpha,beta,beta0,teta,gamma;

// m sürüdeki eleman sayısı
// n değişken (boyut) sayısı
double x[][];  //her elemanın konumu
double y[];    //her elemanın fonksiyon değeri
double yxmin;  //tüm zamanların en iyi değeri
double ygmin;  //en son en iyi değerler
double xmin[]; //tüm zamanların en iyi x konumu
double gmin[]; //en son en iyi  x konumu
```

```
double v[][];  //hız vektörü
int ngeneration;  //döngüdeki nesil sayısı (durdurma kriteri)
f_xj ff;

public suruelemanlari(f_xj ffi,int mi, double xxmin[],double xxmax[],double alphai,double
betai,int ngenerationi)
{
teta=0.5;
gamma=0.5;
ff=ffi;
//ilk nüfusu oluştur
n=xxmin.length;
m=mi;
alpha=alphai;
beta0=betai;
beta=beta0;
ngeneration=ngenerationi;
x=new double[m][n];
v=new double[m][n];
y=new double[m];
xmin=new double[n];
gmin=new double[n];

ngeneration=ngenerationi;
yxmin=1.0e50;
for(int i=0;i<m;i++)
{for(int j=0;j<n;j++)
  {x[i][j]=xxmin[j]+Math.random()*(xxmax[j]-xxmin[j]);
   v[i][j]=Math.random();
  }

  y[i]=ff.func(x[i]);
  if(y[i]<yxmin)
    {yxmin=y[i];
     ygmin=y[i];
    for(int k=0;k<n;k++)
    {xmin[k]=x[i][k];
     gmin[k]=x[i][k];
    } //end for(int k=0;k<n;k++)
    }//end if(y[i]<yxmin)
}
//ilk nufus oluşumu sonu
int t=0;

while(t<ngeneration)
{
yxmin=1.0e50;
for(int i=0;i<m;i++)
{  for(int j=0;j<n;j++)
   {x[i][j]=x[i][j]+v[i][j];
   if(x[i][j]<xxmin[j]) x[i][j]=xxmin[j];
   if(x[i][j]>xxmax[j]) x[i][j]=xxmax[j];
   }
```

```
    y[i]=ff.func(x[i]);
    if(y[i]<yxmin)
    {yxmin=y[i];
     for(int k=0;k<n;k++)
     {xmin[k]=x[i][k];}
    }
}
for(int i=0;i<m;i++)
{
    if(y[i]<ygmin)
    {ygmin=y[i];
    for(int k=0;k<n;k++)
    {gmin[k]=x[i][k];}
    }
}
for(int i=0;i<m;i++)
{
    for(int j=0;j<n;j++)
    {v[i][j]=teta*v[i][j]+alpha*Math.random()*(gmin[j]-
xmin[j])+beta*Math.random()*(xmin[j]-x[i][j]);}
}//end for int i=..
t++;
beta=beta0*Math.pow(gamma,0.01*t);
}}


public String toString()
{ String s="ymin = "+ff.func(xmin)+"¥nxmin = ";
  for(int k=0;k<n;k++)
  {s+=xmin[k]+" ";}
  s+="¥ny global min = "+ff.func(gmin)+"¥nxmin = ";
  for(int k=0;k<n;k++)
  {s+=gmin[k]+" ";}
  return s;
}

public String toString(double x[])
{ String s="ymin = "+ff.func(x)+"¥nxmin = ";
  for(int k=0;k<n;k++)
  {s+=x[k]+" ";}
  return s;
}

public double[] xmin()
{
  return xmin;
}
public double[] gmin()
{
  return gmin;
}
```

```
public static double[] nelder(f_xj fnelder,double a[],double da[],int maxiteration,double
tolerance,int printlist)
{
       int i,j;
double x[][]=new double[a.length+1][a.length];
double p[][]=new double[a.length+1][a.length+1];
for(i=0;i<x.length;i++)
   {for(j=0;j<x[0].length;j++)
     {if(i==j){x[i][j]=a[j]+da[j];p[i][j]=x[i][j];}
      else    {x[i][j]=a[j];p[i][j]=x[i][j]; }
     }
     p[i][j] = fnelder.func(p[i]);
   }


   // Inlet variable definitions
   // fnelder : abstract multivariable function f(x)
   // x : independent variable set of n+1 simplex elements
   // maxiteration : maximum iteration number
   // tolerance :
         int NDIMS = x.length-1;
         int NPTS = x.length;
         int FUNC = NDIMS;
         int ncalls = 0;
            ////// construct the starting simplex //////////////////
            //double p[][]=new double[NPTS][NPTS]; // [row][col] = [whichvx][coord,FUNC]
            double z[]=new double[NDIMS];
            double best = 1E99;
            ///////////////// calculate the first function values for the simplex
////////////////

            int iter=0;

            for (iter=1; iter<maxiteration; iter++)
            {
                  ///////////  define lo, nhi, hi (low high next_to_high //////////////
               int  ilo=0, ihi=0, inhi = -1; // -1 means missing
               double flo = p[0][FUNC];
               double fhi = flo;
               double pavg,sterr;
               for (i=1; i<NPTS; i++)
               {
                  if (p[i][FUNC] < flo)
                    {flo=p[i][FUNC]; ilo=i;}
                  if (p[i][FUNC] > fhi)
                    {fhi=p[i][FUNC]; ihi=i;}
               }
               double fnhi = flo;
               inhi = ilo;
               for (i=0; i<NPTS; i++)
                 if ((i != ihi) && (p[i][FUNC] > fnhi))
                   {fnhi=p[i][FUNC]; inhi=i;}
               ////////// exit criteria //////////////
```

```
if ((iter % 4*NDIMS) == 0)
{
    // calculate the avarage (including maximum value)
    pavg=0;
    for(i=0;i<NPTS;i++)
      pavg+=p[i][FUNC];
    pavg/=NPTS;
    double tot=0;
    if(printlist!=0)
    {  System.out.print(iter);
       for (j=0; j<=NDIMS; j++)
          { System.out.print(p[ilo][j]+" ");}
       System.out.println("");
    }
    for(i=0;i<NPTS;i++)
    { tot=(p[i][FUNC]-pavg)*(p[i][FUNC]-pavg);}
    sterr=Math.sqrt(tot/NPTS);
    //if(sterr < tolerance)
      { for (j=0; j<NDIMS; j++)
           { z[j]=p[ilo][j];}
        //break;
      }
    best = p[ilo][FUNC];
}

///// calculate avarage without maximum value //////

double ave[] = new double[NDIMS];
for (j=0; j<NDIMS; j++)
  ave[j] = 0;
for (i=0; i<NPTS; i++)
  if (i != ihi)
    for (j=0; j<NDIMS; j++)
       ave[j] += p[i][j];
for (j=0; j<NDIMS; j++)
   ave[j] /= (NPTS-1);


///////// reflect ////////////////

double r[] = new double[NDIMS];
for (j=0; j<NDIMS; j++)
  r[j] = 2*ave[j] - p[ihi][j];
double fr = fnelder.func(r);

if ((flo <= fr) && (fr < fnhi))  // in zone: accept
{
    for (j=0; j<NDIMS; j++)
      p[ihi][j] = r[j];
    p[ihi][FUNC] = fr;
    continue;
}
```

```
                if (fr < flo)  //// expand
                {
                    double e[] = new double[NDIMS];
                    for (j=0; j<NDIMS; j++)
                      e[j] = 3*ave[j] - 2*p[ihi][j];
                    double fe = fnelder.func(e);
                    if (fe < fr)
                    {
                       for (j=0; j<NDIMS; j++)
                         p[ihi][j] = e[j];
                       p[ihi][FUNC] = fe;
                       continue;
                    }
                    else
                    {
                       for (j=0; j<NDIMS; j++)
                         p[ihi][j] = r[j];
                       p[ihi][FUNC] = fr;
                       continue;
                    }
                }

                ////////////// shrink:

                if (fr < fhi)
                {
                    double c[] = new double[NDIMS];
                    for (j=0; j<NDIMS; j++)
                      c[j] = 1.5*ave[j] - 0.5*p[ihi][j];
                    double fc = fnelder.func(c);
                    if (fc <= fr)
                    {
                        for (j=0; j<NDIMS; j++)
                          p[ihi][j] = c[j];
                        p[ihi][FUNC] = fc;
                        continue;
                    }
                    else   /////// daralt
                    {
                        for (i=0; i<NPTS; i++)
                          if (i != ilo)
                          {
                              for (j=0; j<NDIMS; j++)
                                p[i][j] = 0.5*p[ilo][j] + 0.5*p[i][j];
                              p[i][FUNC] = fnelder.func(p[i]);
                          }
                        continue;
                    }
                }

                if (fr >= fhi)   ///
                {
                    double cc[] = new double[NDIMS];
```

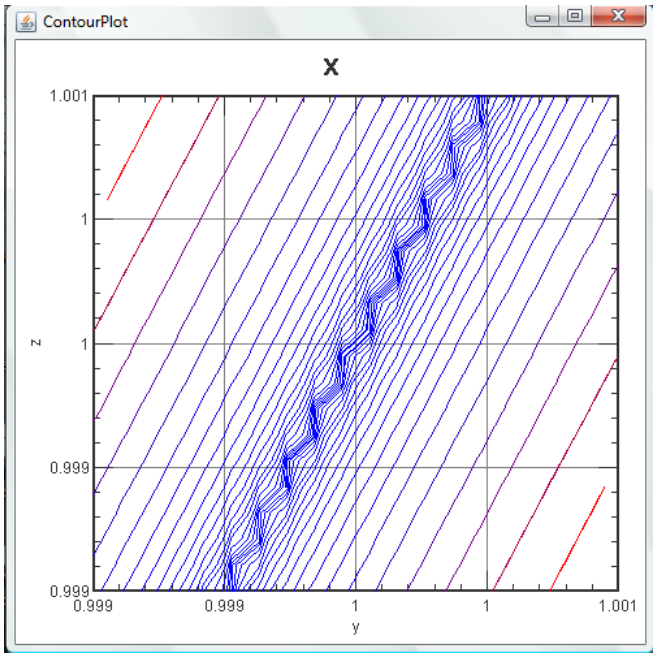
```
                    for (j=0; j<NDIMS; j++)
                      cc[j] = 0.5*ave[j] + 0.5*p[ihi][j];
                    double fcc = fnelder.func(cc);
                    if (fcc < fhi)
                    {
                        for (j=0; j<NDIMS; j++)
                          p[ihi][j] = cc[j];
                        p[ihi][FUNC] = fcc;
                        continue;
                    }
                    else    /////////
                    {
                        for (i=0; i<NPTS; i++)
                          if (i != ilo)
                          {
                              for (j=0; j<NDIMS; j++)
                                p[i][j] = 0.5*p[ilo][j] + 0.5*p[i][j];
                              p[i][FUNC] = fnelder.func(p[i]);
                          }
                    }
                }
            }
        return z;
}
      public static double[] nelder(f_xj fnelder,double a[],double da[],double tolerance)
      {return nelder(fnelder,a,da,500,tolerance,0);}

      public static double[] nelder(f_xj fnelder,double a[],double da[])
      {return nelder(fnelder,a,da,500,1.0e-10,0);}

      public static double[] nelder(f_xj fnelder,double a[])
      {
                 double [] da=new double[a.length];
            for(int i=0;i<a.length;i++) da[i]=0.1*a[i];
            return nelder(fnelder,a,da);
      }
}
```

Program 5.22.2 sürü elemanları test programı

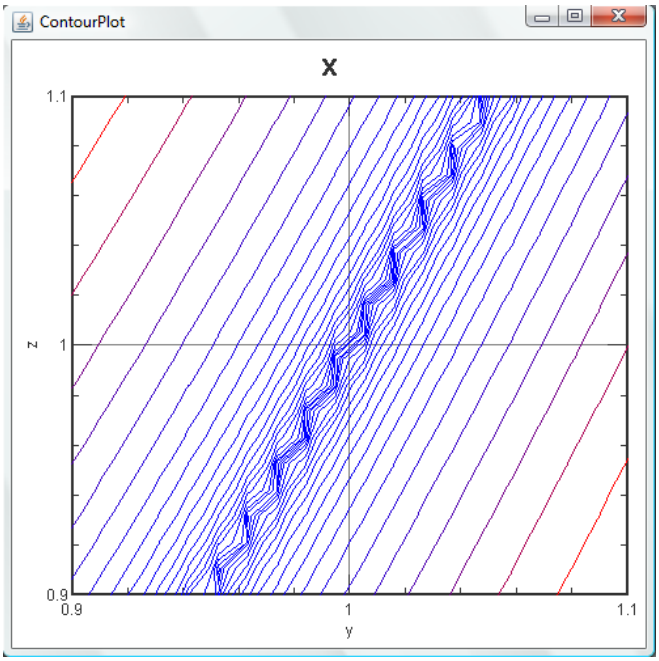
```
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class fa extends f_xj
{
int n;
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=(1.0-x[0])*(1.0-x[0])+100.0*(x[1]-x[0]*x[0])*(x[1]-x[0]*x[0]);
return ff; //maksimum testi
```

```
}}

public class suruelemanlaritesti1
{
public static void main(String arg[])
{
fa ff=new fa();
double xmin[]={0.0,0.0};
double xmax[]={5.0,5.0};
suruelemanlari ps=new suruelemanlari(ff,500,xmin,xmax,2.0,0.7,500);
double r1[]= suruelemanlari.nelder(ff,ps.gmin());
String s="Sürü elemanları (PSO) optimizasyonu  : ¥n"+ps.toString()+"¥n Nelder-Mead:
¥n"+ps.toString(r1);
     String s2="Sürü elemanları (PSO)-Nelder-Mead simpleks optimization metodu ";
     System.out.println(s2+s);
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" suruelemanlaritesti1
Sürü elemanları (PSO)-Nelder-Mead simpleks optimization metodu Sürü elemanları (PSO) optimizasyonu  :
ymin = 0.0
xmin = 1.0 1.0
y global min = 0.0
xmin = 1.0 1.0
 Nelder-Mead:
ymin = 0.0
xmin = 1.0 1.0

> Terminated with exit code 0.
```



(bolum5_032.jpg,Resimaltı **:sürü elemanları test fonksiyonu çözüm seti**)



(bolum5_033.jpg,Resimaltı **:sürü elemanları test fonksiyonu çözüm seti**)

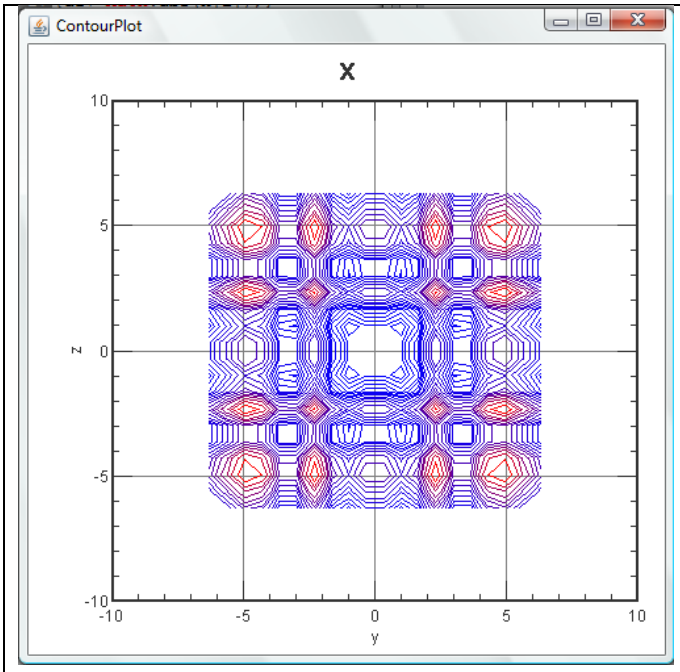
Diğer bir örnek fonksiyon:

$f(x) = \left(\sum_{i=0}^{d} |x_i| \right) \exp\left(- \sum_{i=0}^{d} \sin(x_i^2) \right) \quad -2\pi \le x_i \le 2\pi$ Bu fonksiyonun global minimum değeri [0,0] noktasındadır. Ve fonksiyon değeri de 0 dır.

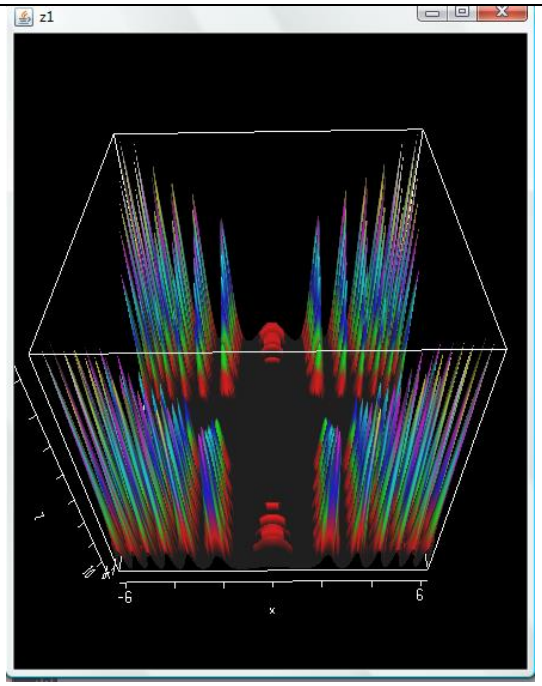
Program 5.22.3 sürü elemanları test programı:

```
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class fa extends f_xj
{
int n;
public double func(double x[])
{ //Michaelewicz function
int n=x.length;
double ff=0;
double a1=0,a2=0;
for(int i=0;i<n;i++)
{a1+=Math.abs(x[i]);}
for(int i=0;i<n;i++)
{a2+=Math.sin(x[i]*x[i]);}
ff=a1*Math.exp(-a2);
return ff;
}}

public class suruelemanlaritesti2
{
public static void main(String arg[])
{
fa ff=new fa();
double xmin[]={0.0,0.0};
double xmax[]={2.5,2.5};
suruelemanlari ps=new suruelemanlari(ff,500,xmin,xmax,2.0,0.7,500);
double r1[]= suruelemanlari.nelder(ff,ps.gmin());
String s="Sürü elemanları (PSO) optimizasyonu  : \n"+ps.toString()+"\n Nelder-Mead: \n"+ps.toString(r1);
    String s2="Sürü elemanları (PSO)-Nelder-Mead simpleks optimization metodu \n";
    System.out.println(s2+s);
}
}
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" suruelemanlaritesti2
Sürü elemanları (PSO)-Nelder-Mead simpleks optimization metodu
Sürü elemanları (PSO) optimizasyonu  :
ymin = 0.0
xmin = 0.0 0.0
y global min = 0.0
xmin = 0.0 0.0
 Nelder-Mead:
ymin = 0.0
xmin = 0.0 0.0

> Terminated with exit code 0.
```



(bolum5_034.jpg,Resimaltı **:sürü elemanları test fonksiyonu Contour grafiği)**

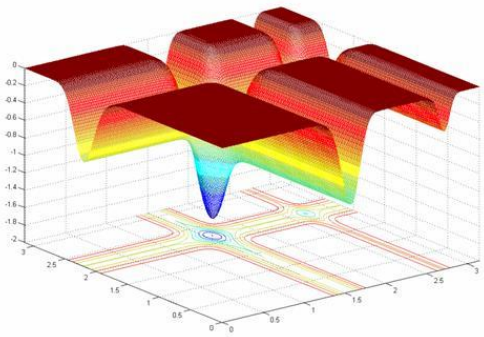


(bolum5_035.jpg,Resimaltı **:sürü elemanları testi 3D grafiği)**

Diğer bir test fonksiyonu Michaelewicz fonksiyonudur. Bu fonksiyon

$$f(x) = -\sum_{i=0}^{n}\left[\sin(x_i)\left[\sin\left(\frac{(i+1)x_i^2}{\pi}\right)\right]^{2m}\right]$$

Şeklinde verilmiştir. Minimum noktası f = −1.801 , P(2.2029, 1.5707963267949) noktası olarak verilmiştir.



(bolum5_036.jpg,Resimaltı **:sürü elemanları Michaelewic test fonksiyonu 3D grafiği)**

```
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class fa extends f_xj
{
int n;
```

```
public double func(double x[])
{

//çözümü istenen fonksiyon
double ff=0;
int n=x.length;
for(int i=0;i<n;i++)
{ff+=-Math.sin(x[i])*Math.pow(Math.sin((i+1)*x[i]*x[i]/Math.PI),20);}
return ff; //maksimum testi
}
}



public class suruelemanlaritesti3
{
public static void main(String arg[])
{
fa ff=new fa();
double xmin[]={0.0,0.0};
double xmax[]={2.5,2.5};
suruelemanlari ps=new suruelemanlari(ff,500,xmin,xmax,2.0,0.7,500);
double r1[]= suruelemanlari.nelder(ff,ps.gmin());
String s="Sürü elemanları (PSO) optimizasyonu  : \n"+ps.toString()+"\n Nelder-Mead: \n"+ps.toString(r1);
    String s2="Sürü elemanları (PSO)-Nelder-Mead simpleks optimization metodu \n";
    System.out.println(s2+s);
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" suruelemanlaritesti3
Sürü elemanları (PSO)-Nelder-Mead simpleks optimization metodu
Sürü elemanları (PSO) optimizasyonu  :
ymin = -1.8013034100985532
xmin = 2.2029055222041776 1.5707963296373977
y global min = -1.8013034100985537
xmin = 2.2029055208904484 1.5707963268735161
 Nelder-Mead:
ymin = -1.8013034100985537
xmin = 2.2029055208904484 1.5707963268735157

> Terminated with exit code 0.
```

5.23 Global tesadüfi metodlar : Yarasa simülasyonu

Xin-She Yang (2010) [51] Yarasa algoritmasını (BA) geliştirmiştir. Bu algoritma yarasaların sosyal davranışları konusundaki araştırmalardan şekillenmiştir. Bu metod temel olaral yarasaların ses yansıtma ile avlarının yerini saptama davranışlarını kullanmaktadır. Mikroyarasalar sonar sistemine benzer bir mekanizmayla avlarını , önlerine çıkan engelleri saptamaktadırlar. Yarasalar çok yüksek ses seviyesinde kısa süreli sesler yaratırlar ve bu seslerin etraftaki cisimlerden geri yansımalarını dinlerler. Gönderilen seslerin özellikleri kullanma gayesine ve yarasa türüne göre değişebilir. Bu temel özellikten yararlanarak Xin-She Yang aşağıda tablo 5.23.1 de verilen algoritmayı geliştirmiştir.

Table 5.23.1 Yarasa arama algoritması

Optimizasyon fonksiyonu $f(x) =, \ x = (x_1, x_2, ..., x_d)^T$

Yarasa sürüsünü oluştur konum x_i (i = 1, 2, …, n) ve hız v_i
x_i konumundaki ses frekansını ver f_i at
ses oluşturma zaman aralığını r_i ve ses gürültü seviyesini A_i *tanımla*
while (t < Max döngü sayısı için)
{
Frekansı ayarlayarak yeni çözüm oluşturun
Bu yeni frekansa göre hız ve konumu tekrar oluşturun

$$f_i = f_{min} + (f_{max} - f_{min})\beta$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*) f_i$$

$$x_i^t = x_i^{t-1} + v_i^t$$

If (rand >*ri*)
{ çözümler arasında en iyi olanını seçin
En iyi çözüm civarındaki yerel çözümleri oluşturun
}
End if
Tesadüfi bir uçuşla en iyi çözümü saptayın
If (rand < A_i and $f(x_i) < f(x_*)$)
{ yeni çözümü kabul edin
r_i *ses oluşturma zaman aralığı değerini arttırın ve* A_i *değerini azaltın*
}
End if
Yarasalatı sıralayın ve en iyi olan değeri saptayın *x**
} **End while**
Sonuçalrı çıktı olarak verin

Yarasalar avlarını aramak için *vi* hızıyla *xi* konumunda tesadüfi olarak fmin sabit frekansı, değişken λ dalga boyu ve A0 gürültü seviyesi yayarak uçarken yaydıkları sesin dalgaboylarını (veya frekanslarını) değiştirebilirler. Zaman içindeki ses oluşturma aralıklarını da değiştirebilirler. Ava yaklaştıkça bu aralıklar azalır. $r \in [0,1]$, Çıkan sesin gürültü seviyesi de yüksek bir A0 seviyesinden minimum bir Amin seviyesine değişebilir. . $\beta \in [0,1]$ tesadufi olarak oluşturulan bir sayıdır. Bu temele göre yukardaki algoritma kullanılarak yarasa simülasyon kodu geliştirilmiştir.

```
import java.io.*;
import java.text.*;
import java.util.Locale;

abstract class f_xj
{
  abstract double func(double x[]);
}

//Program to find minimum of the function
//for maximum finding change the sign of the function

public class yarasa
{
int m,n;
// n  yarasa sayısı
// m değişken(boyut) sayısı
double x[][]; //yarasa konumu
double y[]; // fonksiyon değeri;
double f[];  //ses frekansı
double v[][]; //hız
double ri[];  //ses oluşturma aralığı
double ymin;
double xmin[];
double A[];
double beta;
double fmin,fmax;//minimum ve maksimum freakanslar
double A0,Amin,r0;
double alpha,gamma;
double t;
int ngeneration;
int imin;
double xxmin[];
double xxmax[];

public yarasa(f_xj ff,int ni, double xxmini[],double xxmaxi[],int ngenerationi)
{
```

```
// orijinal yarasa populasyonu
xxmin=xxmini;
xxmax=xxmaxi;
fmin=0;
fmax=2;
A0=0.5;
Amin=0;
r0=0.5;
alpha=0.9;
gamma=0.9;
n=ni;
m=xxmin.length;
x=new double[n][m];
v=new double[n][m];
y=new double[n];
ri=new double[n];
f=new double[n];
xmin=new double[m];
A=new double[n];
double eps;
double AA=0;
double xold;
ngeneration=ngenerationi;
// initilize yarasa population
for(int i=0;i<n;i++)
{ ri[i]=Math.random();
  for(int j=0;j<m;j++)
  {
  x[i][j]=xxmin[j]+Math.random()*(xxmax[j]-xxmin[j]);
  }
  y[i]=ff.func(x[i]);
  f[i]=fmin+(fmax-fmin)*Math.random();
  checkmin(i);
}
//initilize yarasa velocities
for(int i=0;i<n;i++)
{for(int j=0;j<m;j++)
  {
  v[i][j]= v[i][j]+(x[i][j]-xmin[j])*(f[i]-fmin);
  }
}
//initilize Ai and ri
for(int i=0;i<n;i++)
{ri[i]=r0*Math.random();
 A[i]=A0;
}
int l;
t=0;l=0;
while(l<ngeneration)
{
for(int i=0;i<n;i++)
{for(int j=0;j<m;j++)
  {xold=x[i][j];
   x[i][j]= x[i][j]+v[i][j];
   //if out of range
   int ic1=0;
   int ic2=0;
   while(x[i][j]<xxmin[j]&&ic1<5)  {x[i][j]= xold+Math.random()*v[i][j];ic1++;}
   while(x[i][j]>xxmax[j]&& ic2<5) {x[i][j]= xold-Math.random()*v[i][j];ic2++;}
   if(x[i][j]<xxmin[j])  x[i][j]=xxmin[j]+(xxmax[j]-xxmin[j])*Math.random();
```

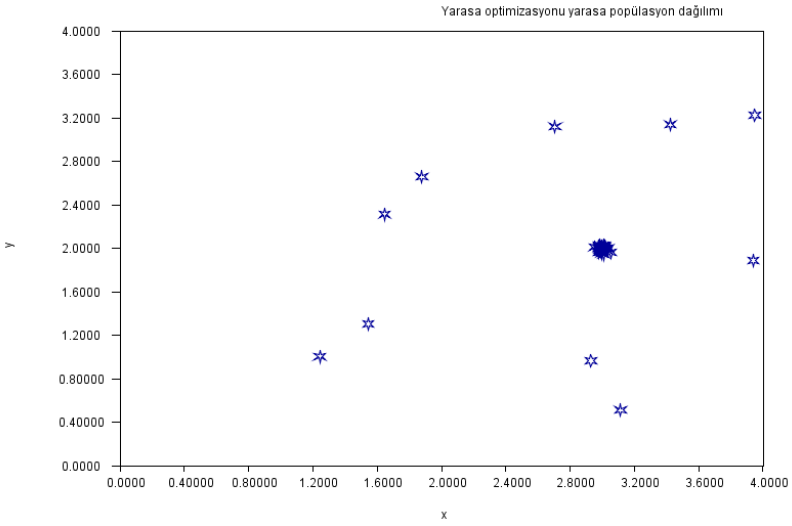
```
   if(x[i][j]>xxmax[j])  x[i][j]=xxmin[j]+(xxmax[j]-xxmin[j])*Math.random();
  }
  y[i]=ff.func(x[i]);
  f[i]=fmin+(fmax-fmin)*Math.random();
  checkmin(i);
}
for(int i=0;i<n;i++)
{for(int j=0;j<m;j++)
  {v[i][j]= v[i][j]+(x[i][j]-xmin[j])*f[i];}
}
for(int i=0;i<n;i++)
{ri[i]=r0*Math.random()*(1-Math.exp(-gamma*t));
 A[i]=alpha*A[i];
}
//random walk
for(int i=0;i<n;i++)
{if(Math.random()>ri[i])
 {x[i]=local_solution(xmin,xxmin,xxmax,0.05);
  y[i]=ff.func(x[i]);
   checkmin(i);
  }
}

for(int i=0;i<n;i++)
{ if(Math.random()<A[i] && y[i]<ymin)
  AA=avg(A);
 for(int j=0;j<m;j++)
 {eps=-1+2.0*Math.random();
 x[i][j]=x[i][j]+eps*AA;}
 y[i]=ff.func(x[i]);
 checkmin(i);
  ri[i]=r0*Math.random()*(1-Math.exp(-gamma*t));
  A[i]=alpha*A[i];
}

for(int i=0;i<n;i++)
{  checkmin(i);}
   l++;
   t=l;
}//end of while

}

// if minimum is found replace ymin and xmin
// with the new founded value
public void checkmin(int i)
{
 if(y[i]<ymin)
   {ymin=y[i];imin=i;
   for(int j=0;j<m;j++)
   {xmin[j]=x[i][j];}
   }
}

public static double avg(double xx[])
{ double toplam=0;
  int nn=xx.length;
  for(int i=0;i<nn;i++)
  {toplam+=xx[i];}
  toplam/=nn;
```

```
  return toplam;
}

//Gauss random tesadüfi sayı oluşturma
public static double  gaussrandom(double xavg,double r)
{ int iset=0;
 double gset=0;
 double fac,rsq,v1,v2;
 double aret=0;
 if(iset==0)
 {
   do{
            v1=2.0*Math.random()-1.0;
            v2=2.0*Math.random()-1.0;
            rsq=v1*v1+v2*v2;
     }while(rsq>=1.0 || rsq==0.0);
   fac=Math.sqrt(-2.0*Math.log(rsq)/rsq);
   gset=v1*fac;
   iset=1;
   aret=v2*fac;
 }
 else
 {
 iset=0;
 aret=gset;
 }
 return aret/8.0*r+xavg;
}

//seçilen en iyi çözüm çevresinde gauss dağılımını kullanarak yeni iyi çözümler arar
public static double[] local_solution(double selected_best[],double xxmin[],double xxmax[],double ratio)
{
int m=xxmin.length;
double dx=1;
  double xx[]=new double[m];
  for(int j=0;j<m;j++)
  {
  dx=(xxmax[j]-xxmin[j])*ratio;
  xx[j]=gaussrandom(selected_best[j],dx);
  //we will not permit data out of range
  int ic1=0;
  int ic2=0;
  while(xx[j]<xxmin[j]) {xx[j]=xxmin[j]+0.001*Math.random()*dx;ic1++;}
  while(xx[j]>xxmax[j]) {xx[j]=xxmax[j]-0.001*Math.random()*dx;ic2++;}
  }
  return xx;
}

//çıktı fonksiyonu
public double[] xmin()
{ return xmin;}

//çıktı fonksiyonu
public String toString()
{ String s="ymin = "+ymin+"\nxmin = ";
  for(int k=0;k<xmin.length;k++)
  {s+=xmin[k]+" ";}
  return s;
}
```

```
public void plot()
{
// her yarasa için değişken sayısı (boyut)
if(m==2)
{   double z[][]=Matrix.T(x);
            Plot pp=new Plot(z[0],z[1]);
            pp.setPlotType(0,35);
            pp.setMinMax(xxmin[0],xxmax[0],xxmin[1],xxmax[1]);
            pp.plot();
}}
}
```

```
import javax.swing.*;

class fa extends f_xj
{
//çözümü istenen fonksiyon
 double func(double x[])
 {
   double ff= 1.0+Math.cos(Math.PI*(x[0]-3.0))*Math.cos(2.0*Math.PI*(x[1]-2.0))/
   (1+(x[0]-3.0)*(x[0]-3.0)+(x[1]-2.0)*(x[1]-2.0));
   return -ff; //maksimum test;
  }
}

public class yarasatest
{
public static void main(String arg[])
{
fa ff=new fa();
double xmin[]={0,0};
double xmax[]={4,4};
yarasa  b=new yarasa(ff,40,xmin,xmax,100);
double x0[]=b.xmin();
String s="yarasa optimizasyonu : \n"+b.toString()  ;
    String s2="yarasa arama optimizasyon metodu \n";
    System.out.println(s2+s);
b.plot();
    }

}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" yarasatest
yarasa arama optimizasyon metodu
yarasa optimizasyonu :
ymin = -1.9999885757031817
xmin = 3.000651801109898 1.9993448038893664
```



(bolum5_037.jpg,Resimaltı **:yarasa sürü elemanları çözüm dağılımı)**

5.24 Global tesadüfi metodlar : Ahenk ve Melodi arama metodları

Ahenk (Hahenk) arama algoritması(HS) Z. W. Geem *et cd.* in 2001[53] tarafından geliştirilmiş göreceli olarak yeni bir algoritmadır. Hahenk algoritmasının temelindekikulağa hoş gelen ahenkli müzik yatar. Müziğin temeli kulağa hoş gelen mükemmel bir ahengi yakalamaktır. Bu müzikteki ahenk optimizasyon prosesi olarak bakıldığında optimum çözüme karşı gelir. Optimizasyon prosesi bir müzisyenin ahenkli parça oluşturma çabasıyla aynı şekilde oluşturulabilir. Ahenkli bir parçayı oluşturma ses estetik standartlarını kullanarak yapılır. Bir müzik aletinin estetik kalitesi temel olarak aletin çıkardığı frekansa veya perdeye(pitch), ses kalitesine (timbre) ve ses yüksekliğine (amplitude) bağlıdır. Ses kalitesi genellikle çıkan seslerin hahenkc karekterleriyle oluşur. Buda ses dalga biçimlerinin ve modülasyonlarının fonksiyonudur. Ancak bir bir müzik enstrümanının çıkarabildiği sesler her şeyden önce perde aralığı veya freakansının fonksiyonudur. Perde aralığı ve frekans birbirine dönüştürülebilen fonksiyonlardır.

$$p = 69 + 12 * \log_2\left[\frac{\mathrm{f}}{440\ \mathrm{Hz}}\right]$$ veya ters fonksiyonu

$f = 440\mathrm{x}2^{(\mathrm{p}-69)/12}$ bu fonksiyonu programlarımızda genellikle perde aralığı olarak kullanacağız.

Buna göre do (A) notası frekans f_0=440 Hz veyaperde aralığı p=69 ile ifade edilebilir. Buna göre Ahenk algoritmasını şu şekilde oluşturabiliriz:

Tablo 5.24.1 Ahenk (Harmony) Algoritması

Ahenk (Harmony) Algoritması
Optimizasyon fonksiyonu *f(x), x = (x_0, ...,x_p)T*
İlk ahenkleri oluştur (gerçek zaman boyutlu değişkenleri)
Perde ayarlama oranını tanımla (r_{pa}) ve perde sınırlarını tanımla
Ahenk hafızası kabul oranını tanımla (T_{accept})
while (*t* <Maksimum döngü sayısı)
En iyi ahen oranlarını seçerek yeni ahenkler yarat
Perde sınırlarını ayarlayarak yeni ahenkler oluştur (çözümler)
if (rand> r_{accept}),
mevcut ahenklerin içinden tesadüfi seçim yap
else i f (rand> r_{pa}),
tanımlanan bölgede yeni bir perde tanımla
$\mathrm{x}_{new} = \mathrm{x}_{old} + b_p(2\mathrm{rand} - 1)$
else
ahengi tesadüfi değiştirerek yeni ahenkler oluştur
$\mathrm{p}_a = \mathrm{p}_{lowerlimit} + p_{range}\mathrm{rand}$
end if

```
Eğer yeni ahenkler daha iyiyse kabul et
end while
En iyi ahengi seç
```

Tablo 5.24.1 Ahenk (Harmony) arama

```
import java.io.*;
import java.text.*;
import java.util.Locale;

abstract class f_xj
{
  abstract double func(double x[]);
}

public class Ahenk {
        public int m;       // değişken(boyut) sayısı
        public int HMS;     // Ahenk hafıza boyutu(popülastyon sayısı)
        public int maxIter;  // maksimum iterasyon sayısı
        public double PAR;   // perde ayar oranı
        public double BW;    // perde genişliği
        public double HMCR;  // Ahenk hafızası ayar oranı 0< HMCR< 1
        public double xmin[]; // bağımsız değişkenlerin minimum sınırı
        public double xmax[]; //bağımsız değişkenlerin maksimum sınırı
        public double NCHV[];
        public double bestFitHistory[];  // en iyi uyum tarihçesi
        public double bestHarmony[];     // En iyi ahenk
        public double worstFitHistory[]; // en kötü ahenk tarihçesi
        public double HM[][];            //Ahenk hafızası
        static int generation = 0;
        public boolean terminationCriteria = true;
        f_xj ff;

   public Ahenk(f_xj ffi,double xxmin[],double xxmax[],int maxIteration,double BWi,double HMCRi,int
HMSi,double PARi)
   {       ff=ffi;
             maxIter=maxIteration;
             m=xxmin.length;
             HMCR=HMCRi;
             HMS=HMSi;
             BW=BWi;
             xmin = xxmin;
                    xmax = xxmax;
                    NCHV = new double[m];
                    bestHarmony = new double[m + 1];
                    bestFitHistory = new double[maxIter + 1];
                    worstFitHistory = new double[maxIter + 1];
                    HM = new double[HMS][m + 1];
             xmin=xxmin;
             xmax=xxmax;
   }

   public double random(double low,double high)
        {return low+Math.random()*(high-low);}

        public int random(int low, int high)
        {return (int)(Math.floor(Math.random()*(high-low)+low+.5));}

        public void initiator()
        {
                int i;
```

```
		double curFit;
		for (i = 0; i < HMS; i++)
		{
			for (int j = 0; j < m; j++) {HM[i][j] = random(xmin[j], xmax[j]);}
			curFit = ff.func(HM[i]);  // uyum(HM[i]);
			HM[i][m] = curFit;	// uyum ahenk hafızasının son kolonuna yüklendi of HM
		}
	}

	public boolean stopCondition() {
		if (generation > maxIter)
			terminationCriteria = false;
		return terminationCriteria;
	}

	public void updateHarmonyMemory(double newFitness) {

		// find worst harmony
		double worst = HM[0][m];
		int worstIndex = 0;
		for (int i = 0; i < HMS; i++)
			if (HM[i][m] > worst) {
				worst = HM[i][m];
				worstIndex = i;
			}
		worstFitHistory[generation] = worst;
		// Uyumu değiştir
		if (newFitness < worst) {
			for (int k = 0; k < m; k++)
				HM[worstIndex][k] = NCHV[k];
			HM[worstIndex][m] = newFitness;
		}
		// en iyi uyumu bul
		double best = HM[0][m];
		int bestIndex = 0;
		for (int i = 0; i < HMS; i++)
			if (HM[i][m] < best) {
				best = HM[i][m];
				bestIndex = i;
			}
		bestFitHistory[generation] = best;
		if (generation > 0 && best != bestFitHistory[generation - 1]) {
			for (int k = 0; k < m; k++)
				bestHarmony[k] = HM[bestIndex][k];
			bestHarmony[m] = best;
		}
	}

	public void memoryConsideration(int varIndex) { NCHV[varIndex] = HM[random(0, HMS -
1)][varIndex];}

	public void pitchAdjustment(int varIndex) {

		double rand = Math.random();
		double temp = NCHV[varIndex];
		if (rand < 0.5) {
			temp += rand * BW;
			if (temp < xmax[varIndex])
				NCHV[varIndex] = temp;
		} else {
```

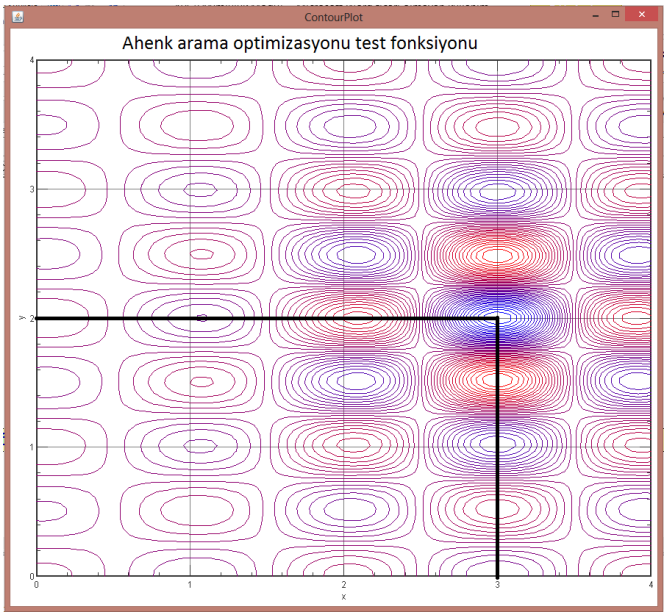
```
                        temp -= rand * BW;
                        if (temp > xmin[varIndex])
                                NCHV[varIndex] = temp;
                }

        }

        public void randomSelection(int varIndex) {NCHV[varIndex] = random(xmin[varIndex],
xmax[varIndex]);}

        public double[]  xmin()
        {
                initiator();
                while (stopCondition()) {
                        for (int i = 0; i < m; i++) {
                                if (Math.random() < HMCR) {
                                        memoryConsideration(i);
                                        if (Math.random() < PAR)
                                                pitchAdjustment(i);
                                } else
                                        randomSelection(i);
                        }
                        double currentFit;
                        currentFit = ff.func(NCHV);//fitness(NCHV);
                        updateHarmonyMemory(currentFit);
                        generation++;
                }
                return bestHarmony;
        }

public String toString()
{ double x[]=xmin();
  double y=x[x.length-1];
        String s="ymin = "+y+"\nxmin = ";
  for(int k=0;k<x.length-1;k++)
  {s+=x[k]+" ";}
  return s;
}

public void plot()
{
if(m==2)
{   double z[][]=Matrix.T(HM);
    for(int i=0;i<z[0].length;i++)
    {System.out.println(z[0][i]+"   "+z[1][i]);}
        Plot pp=new Plot(z[0],z[1]);
        pp.setPlotType(0,35);
        pp.setMinMax(xmin[0],xmax[0],xmin[1],xmax[1]);
        pp.plot();
}
}}
```

Program 5.24.2 Ahenk arama test programı

```
 import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;

class fa extends f_xj
{
```

```
//çözümü istenen fonksiyon
 double func(double x[])
 {
   double ff= 1.0+Math.cos(Math.PI*(x[0]-3.0))*Math.cos(2.0*Math.PI*(x[1]-2.0))/
   (1+(x[0]-3.0)*(x[0]-3.0)+(x[1]-2.0)*(x[1]-2.0));
   return -ff;
  }
}
public class Ahenktesti {

            public static void main(String[] args) {

                        fa ff=new fa();
                        double xmin[]={0,0};
      double xmax[]={4,4};
      int maxIter=1000000;
      double BW=0.2;
      double HMCR=0.9;
      int HMS=10;
      double PAR=0.4;
                        Ahenk hs=new Ahenk(ff,xmin,xmax,maxIter,BW,HMCR,HMS,PAR);
                        String s="Ahenk arama : \n"+hs.toString()  ;
      String s2="Ahenk arama optimizasyonu \n";
      System.out.println(s2+s);
      ContourPlot pp = new ContourPlot(ff,xmin[0],xmax[0],100,xmin[1],xmax[1],100,40, false,"Ahenk arma
optimizasyonu", "x", "y");
                        pp.plot();
            }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" Ahenktesti
Ahenk arama optimizasyonu
Ahenk arama :
ymin = -1.9999999887931241
xmin = 3.0000011052150914 2.000023238370813
```



(bolum5_038.jpg,Resimaltı **:ahenk arama test fonksiyonu contour grafiği)**

Ahenk algoritmasının hesaplama verimini daha arttırmaya çalışan araştırıcılar(Ashrafi and Dariane[54]) melodi arama (melody search-MA) algoritmasını geliştirdiler. Melodi arama bir orkestra tarafından icra edilen bir eserin müzik performansını taklit eder. Müzisyenler melodiye uyan en iyi ahengi yakalamaya çalışır. Değişik deneyimleri ve zevkleri

olan değişik sanatçılar bir arada çalınca tüm bu deneyimlerin kombinasyonu olan bir ahenk yakalarlar. Bu algoritma temel olarak yine bir önceki ahenk algoritmasında bulunan temel kavramları kullanmakla birlikte, yapısal olarak oldukça deişik bir forma sahiptir. Melodi arama algoritması her bir çalgıcıya özgü Ahenk algoritmaları dizinini barındırır. Değişik çalgıcıların hafızaları birbirinden etkilenerek tıpkı bir orkestradaki değişik çalgıcıların birbirinden etkilenmesi gibi ortak bir altyapı oluşturur. Ayrıyeten melodi algoritmasında her perdenin sınırları değişik iterasyonlarda değişebilir. Melodi arama algoritması altta verilmiştir.

Table 5.24.2 MELODİ arama algoritması

Step 1. Otimizasyon prosesi başlangıç değerlerini ata
Step 2. Başlangıç fazı;
2.1. çalgıcı hafizalarına ilk değerlerini ata (PM)
2.2. her çalgıcı Ahenk hafızasından PM yeni bir melodi oluştur
2.3. çalgıcı ahenk hafızalarını yenile PM
2.4. sonuçlar başlangıç kriterine uyana kadar 2.2 ve 2.3 steplerini tekrarla
Başlangıç kriterlerine ulaşıldığında dur
Step 3. İkinci faz;
3.1. Her çalgıcı ahenk hafızasından,PM, mümkün olan perdeleri kullanarak yeni melodiler oluştur
3.2. Tüm çalgıcı ahenk hafizalarını , PM, yenile
3.3. Bir sontraki ahenk gurupları için perde sınırlarını tesadüfi olarak yeniden tanımla
3.4. 3.1, 3.2 ve 3.3 steplaerini başlangıç kriterine uyana kadar tekrarla
Başlangıç kriterine ulaşıldığında dur .

Program 5.24.3 Melodi arama programı

```
//      Referans:
//      Performance evaluation of an improved harmony search algorithm for numerical
//      optimization : Melody Search (MS)   S.M. Ashrafi , A.B.Dariane
//      Programlayan : Oğuz Emrah Turgut

abstract class f_xj
{
abstract double func(double x[]);
}

public class melodi_arama
{
int D;// Dimension(Number of pitch)
int PMN;//number of player memory
int PMS;//player memory size
int Maxiter1;//maximum of iteration for  first phase
int Maxiter2;//maximum number of iteration for second phase
double PMCR;//player memory considering rate
double PARmax;// Maximum value of pitch adjustment rate
double PARmin;// Minimum value of pitch adjustment rate
double PAR;//Pitch adjustment rate(chenges within every iteration)
double Xnew[][];//Holds new pitches for stage 1
double Xnew2[];//Holds new pitches for stage 2
double[][] fitness;// Holds fitness values for stage 1
double[][] fitness2;// Holds fitness value for stage 2
double[] fitness3;
double[][][] MM;//Melody memory first stage
double[][] MM2;//Melody memory second stage
double[] Lower1;// Lower bound of first stage
double[] Upper1;// Upper bound of first stage
double[] Lower2;// Lower bound of second stage (determined by first stage)
double[] Upper2;// Upper bound of second stage (determined by first stage)
double[] maxval;
int[] maxindex;
double[] minval;
int[] minindex;
double[] maxval2;
int[] maxindex2;
```

```
double[] minval2;
int[] minindex2;
double[] bw;//bandwith
double[][] worst;
double[][] best;
double[][] best2;
double[] best3;
double[] worst3;
int L;//random integer between 1 and PMS
int H;//random integer between 1 and D
double min2;
int mindex2;
double max2;
int maxdex2;
f_xj ff;

	public melodi_arama(f_xj iff,int iPMN,int iPMS,int iMaxiter1,int iMaxiter2,double iPMCR,double[]
iLower,double[] iUpper)
	{
		ff=iff;
		PMN=iPMN;
		PMS=iPMS;
		Maxiter1=iMaxiter1;
		Maxiter2=iMaxiter2;
		PMCR=iPMCR;
		Lower1=iLower;
		Upper1=iUpper;
		D=Upper1.length;
		Lower2=new double[D];
		Upper2=new double[D];
		maxval=new double[PMN];
		maxindex=new int[PMN];
		minval=new double[PMN];
		minindex=new int[PMN];
		maxval2=new double[PMN];
		maxindex2=new int[PMN];
		minval2=new double[PMN];
		minindex2=new int[PMN];
		fitness=new double[PMN][PMS];
		fitness2=new double[PMN][PMS];
		fitness3=new double[PMN];
		worst=new double[PMN][D];
		best=new double[PMN][D];
		best2=new double[PMN][D];
		best3=new double[D];
		worst3=new double[D];
		MM=new double[PMN][PMS][D];
		MM2=new double[PMN][D];
		Xnew=new double[PMN][D];
		Xnew2=new double[D];
		PARmax=2.0;
		PARmin=0.0;

		}

		void initialize()
		{
			// Improvisation a new melody
			for(int i=0;i<PMN;i++)
			{for(int j=0;j<PMS;j++)
```

```
{for(int k=0;k<D;k++)
 {MM[i][j][k]=Lower1[k]+(Math.random()*(Upper1[k]-Lower1[k]));}}}

 //Calculate fitness of each player
 for(int i=0;i<PMN;i++)
 {for(int j=0;j<PMS;j++)
  {fitness[i][j]=ff.func(MM[i][j]);}}

 // Calculate best and worst melody for each player
  for(int i=0;i<PMN;i++)
  {
          double dd[]=getmaxval_index(fitness[i]);
          maxval[i]=dd[0];
          maxindex[i]=(int)dd[1];
         }

         for(int i=0;i<PMN;i++)
  {
          double dd2[]=getminval_index(fitness[i]);
          minval[i]=dd2[0];
          minindex[i]=(int)dd2[1];
         }
         for(int i=0;i<PMN;i++)
         {
           for(int k=0;k<D;k++)
           {
                   best[i][k]=MM[i][minindex[i]][k];
                   }
         }
         for(int i=0;i<PMN;i++)
         {
           for(int k=0;k<D;k++)
           {
                   worst[i][k]=MM[i][maxindex[i]][k];
                   }
         }
}


double[][] solution()
{
initialize();

/// FIRST STAGE /// Determine best melody for each music player

int iter1=1;
        while(iter1<Maxiter1)
        {
            for(int i=0;i<PMN;i++)
            {
                    for(int k=0;k<D;k++)
                    {
                            if(Math.random()<=PMCR) // memory consideration
                   {
       L=(int)(((double)PMS)*Math.random());
                           H=(int)(((double)D)*Math.random());
       PAR=PARmax-((PARmax-PARmin)*((double)iter1)/((double)Maxiter1));
                    if((iter1%2)==1)
                    {Xnew[i][k]=MM[i][L][k]+(((Math.random()-0.5)*2.0)*((Upper1[k]-
Lower1[k])/200.0));   }
```

```
                                    else
                                    {Xnew[i][k]=MM[i][L][H]+(((Math.random()-0.5)*2.0)*((Upper1[k]-
Lower1[k])/200.0));  }
                                        if(Math.random()<=PAR) // pitch adjustment
                                        {Xnew[i][k]=best[i][k];  }
                                      }
                                        else // randomization
                                   {Xnew[i][k]=Lower1[k]+(Math.random()*(Upper1[k]-Lower1[k]));    }
                                         }    }
                                    for(int i=0;i<PMN;i++)   // Xnew is better than worst , exclude worst
                                    {if(ff.func(Xnew[i])<ff.func(worst[i]))
                                       {for(int k=0;k<D;k++)
                                              {worst[i][k]=Xnew[i][k];}}
                                    }
                                    // Update  worst and best melody
                                    for(int i=0;i<PMN;i++)
                              {for(int j=0;j<PMS;j++)
                                {fitness[i][j]=ff.func(MM[i][j]);}}
                           for(int i=0;i<PMN;i++)
                           {
                                 double dd3[]=getmaxval_index(fitness[i]);
                                 maxval[i]=dd3[0];
                                 maxindex[i]=(int)dd3[1];
                                 }
                                 for(int i=0;i<PMN;i++)
                           {
                                 double dd4[]=getminval_index(fitness[i]);
                                 minval[i]=dd4[0];
                                 minindex[i]=(int)dd4[1];
                                 }
                                       for(int i=0;i<PMN;i++)
                                 {
                                   for(int k=0;k<D;k++)
                                   {
                                       best[i][k]=MM[i][minindex[i]][k];
                                      }
                                 }

                                      for(int i=0;i<PMN;i++)
                                 {
                                   for(int k=0;k<D;k++)
                                   {
                              MM[i][maxindex[i]][k]=worst[i][k]; // worst is now  new maxindexed MM
                                       }
                                 }

                                 iter1++;
                            }


                            /// SECOND STAGE /// Determine best harmony from selected melodies

                            // Determine new upper and lower bound
                            for(int i=0;i<PMN;i++)
                            {
                              for(int j=0;j<PMS;j++)
                               {fitness2[i][j]=ff.func(MM[i][j]);}
                      }

                             for(int i=0;i<PMN;i++)
```

```
{
 double dd6[]=getminval_index(fitness2[i]);
 minval2[i]=dd6[0];
 minindex2[i]=(int)dd6[1];
}
for(int i=0;i<PMN;i++)
{
  for(int k=0;k<D;k++)
  {
        best2[i][k]=MM[i][minindex[i]][k];
        }
}

for(int i=0;i<PMN;i++)
{
   for(int k=0;k<D;k++)
   {
          MM2[i][k]=best[i][k];
    }
}

// MM2 now is new harmony matrix


// Procedure for determining lower and upper bounds for every column
double min=0.0;
for(int k=0;k<D;k++)
{
        min=MM2[0][k];
   for(int i=0;i<PMN;i++)
   {
        if(MM2[i][k]<min)
        {min=MM2[i][k];}
        }
        Lower2[k]=min;
}



double max=0.0;
for(int k=0;k<D;k++)
{
        max=MM2[0][k];
   for(int i=0;i<PMN;i++)
   {
        if(MM2[i][k]>max)
        {max=MM2[i][k];}
        }
        Upper2[k]=max;
}


// determine best and worst vector in MM2
for(int i=0;i<PMS;i++)
{fitness3[i]=ff.func(MM2[i]);}

double dd7[]=getminval_index(fitness3);
min2=dd7[0];
mindex2=(int)dd7[1];
```

```
                        double dd8[]=getmaxval_index(fitness3);
                        max2=dd8[0];
                        maxdex2=(int)dd8[1];

                        for(int i=0;i<D;i++)
                        {best3[i]=MM2[mindex2][i];
                         worst3[i]=MM2[maxdex2][i];}




                        int iter2=1;
                        while(iter2<Maxiter2)
                        {

                                for(int k=0;k<D;k++)
                                {
                                        if(Math.random()<PMCR)//Memory consideration
                                        {
                                                L=(int)(((double)PMS)*Math.random());

                                    H=(int)(((double)D)*Math.random());
                                    PAR=PARmax-((PARmax-
PARmin)*((double)iter2)/((double)Maxiter2));
                                                if((iter2%2)==1)
                                                {
                                                Xnew2[k]=MM2[L][k]+(((Math.random()-
0.5)*2.0)*((Upper2[k]-Lower2[k])/200.0));
                                                }
                                                else
                                                {
                                                Xnew2[k]=MM2[L][H]+(((Math.random()-
0.5)*2.0)*((Upper2[k]-Lower2[k])/200.0));
                                                }
                                                 if(Math.random()<PAR)// pitch adjustment
                                                 {
                                                        for(int kk=0;kk<D;kk++)
                                                   {Xnew2[kk]=best3[kk];}

                                                 }

                                    }
                                        else // Randomization
                                        {
                                          Xnew2[k]=Lower2[k]+(Math.random()*(Upper2[k]-
Lower2[k]));
                                        }

                                   }

                        if(ff.func(Xnew2)<ff.func(worst3))
                        {
                           for(int i=0;i<D;i++)
                           {worst3[i]=Xnew2[i];}

                        }

                        for(int i=0;i<PMN;i++)
                    {fitness3[i]=ff.func(MM2[i]);}
```

```
                // get minimum index and best solution
                double[] dd9=getminval_index(fitness3);
                min2=dd9[0];
                mindex2=(int)dd9[1];
                for(int i=0;i<D;i++)
                {best3[i]=MM2[mindex2][i];}

                // get maximum index and replace new worst value
                double[] dd10=getmaxval_index(fitness3);
                max2=dd10[0];
                maxdex2=(int)dd10[1];
                for(int i=0;i<D;i++)
                {MM2[maxdex2][i]=worst3[i];}

                        // Procedure for determining lower and upper bounds for every column
                         min=0.0;
                        for(int k=0;k<D;k++)
                        {
                                min=MM2[0][k];
                           for(int i=0;i<PMN;i++)
                           {
                                 if(MM2[i][k]<min)
                                 {min=MM2[i][k];}
                                }
                                Lower2[k]=min;
                        }



                         max=0.0;
                        for(int k=0;k<D;k++)
                        {
                                max=MM2[0][k];
                           for(int i=0;i<PMN;i++)
                           {
                                 if(MM2[i][k]>max)
                                 {max=MM2[i][k];}
                                }
                                Upper2[k]=max;
                        }

                        iter2++;

                        }
                        double[][] dp=new double[2][D];
                        dp[0][0]=min2;
                        for(int i=0;i<D;i++)
                        {dp[1][i]=best3[i];}

                        return dp;

        }

        double[] getmaxval_index(double a[])
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double maxval=a[0];
```

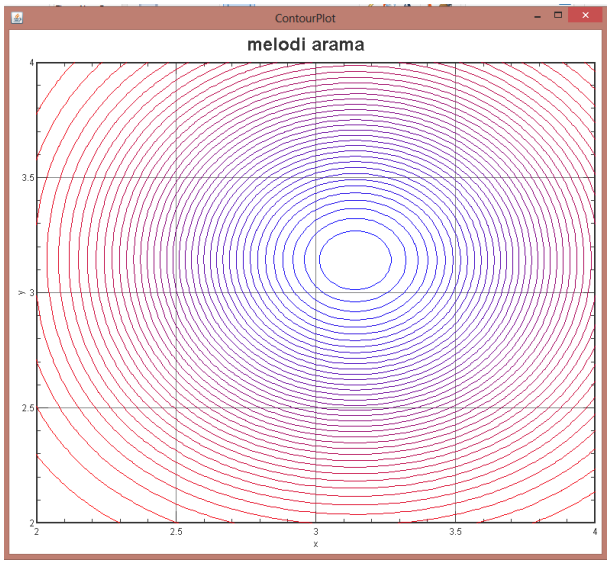
```
            for(int j=0;j<a.length;j++)
            {if(a[j]>maxval){maxval=a[j];}}
            for(int i=0;i<b.length;i++)
            {if(b[i]==maxval){m=i;break;}}
            double dep2[]=new double[2];
            dep2[0]=maxval;
            dep2[1]=m;
            return dep2;
            }



            double[] getminval_index(double[] a)
            {
            double m=0.0;
            double b[]=new double[a.length];
            for(int i=0;i<a.length;i++)
            {b[i]=a[i];}
            double minval=a[0];
            for(int i=0;i<a.length;i++)
            {if(a[i]<minval){minval=a[i];}}
            for(int i=0;i<a.length;i++)
            {if(b[i]==minval){m=i;break;}};
            double[] dep=new double[2];
            dep[0]=minval;
            dep[1]=m;
            return dep;
            }

            public void toString(double out[][])
            {
                System.out.println("Optimized value = "+out[0][0]);
                for(int i=0;i<out[1].length;i++)
                {System.out.println("x["+i+"] = "+out[1][i]);}
            }
                    public void toString(double[] out)
                    {
                     for(int i=0;i<out.length;i++)
                     {System.out.println("x["+i+"] = "+out[i]);}
                    }

}
```

Program 5.24.4 Melodi arama test programı

```
 //http://extreme.adorio-research.org
// Programmed by : Oğuz Emrah Turgut

class f9 extends f_xj// Easom  f(x)=-1.0  @x=(pi,pi)   -100.0<=x[i]<=100.0 i=0,1
{
double func(double x[])
{
double first=0.0;
first=-Math.cos(x[0])*Math.cos(x[1])*Math.exp(-(x[0]-Math.PI)*(x[0]-Math.PI)-(x[1]-Math.PI)*(x[1]-
Math.PI));
return first;
}
}
```

```
public class melodi_arama_testi
{

public static void main(String args[])
                {
                        int PMN=100;// çalgıcı sayısı
        int PMS=100;// çalgıcı hafızası
                        int Maxiter1=5000;// birinci sahne için maksimum döngü sayısı
                        int Maxiter2=1000;// ikinci sahne için maksimum döngü sayısı
        double PMCR=0.99;// çalgıcı hafıza oranı
        double[] xmin={-100.0,-100.0};
        double[] xmax={100.0,100.0};
        f9 ff=new f9();
                        melodi_arama ms=new
melodi_arama(ff,PMN,PMS,Maxiter1,Maxiter2,PMCR,xmin,xmax);
                        double out[][]=ms.solution();
                        ms.toString(out);
                        double r1[]= suruelemanlari.nelder(ff,out[1]);
                        System.out.println("Nelder mead metodu optimum değer : "+ff.func(r1));
                        ms.toString(r1);
                        ContourPlot pp = new ContourPlot(ff,2,4,100,2,4,100,40, false,"melodi arama", "x", "y");
                            pp.plot();
                    }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" melodi_arama_testi
Optimum değer = -0.9999997319746594
x[0] = 3.141891555208508
x[1] = 3.1418915539173806
Nelder mead metodu optimum değer : -1.0
x[0] = 3.141592649926962
x[1] = 3.141592656443677
```



(bolum5_039.jpg,Resimaltı **:melodi arama test fonksiyonu contour grafiği)**

Son yıllarda Ahenk algoritmalarının çok çeşitli şekilleri çeşitli makalelerde yayınlanmıştır. Diğer bir ilginç algoritma Zeki Ahenk algoritması **(**Intelligent tuned Harmony Search**)** [55]'dır. Bu algoritma altta tanımlanmıştır.

Table 5.24.3 Zeki ahenk algoritması

Step 1: IHTS algoritması giriş değerlerini ata
F(x) optimizasyon fonksiyonu
N: değişken(boyut) sayısı
x_{i_lower} : alt sınır
x_{i_upper} : üst sınır
HMS: Ahenk hafıza boyutu=10
HMCR: Ahenk hafıza değişim oranı=0.99
PAR_{min}=0
PAR_{max}=1
Maxiter=m
Step 2: Ahenk hafızasının ilk değerlerini ata(HM)
Ahenk hafızasındaki en iyi ve en kötü değerleri sapta HM
Step3: Yeni ahenk hafızası oluştur (HM)
while (not Maxiter)
for i = 1 to değişken(boyut) sayısı N do
{
if rand [0, 1] <=HMCR
{ // (hafıza değişkenleri) //
d = int(1 + (HMS _ 1) rand [0, 1])
x_i'=HM(d,i)

$$PAR(iter) = PAR_{max} - (PAR_{max} - PAR_{min})\left[\frac{iter}{\max iter}\right]$$

if rand [0, 1] $\le$ PAR(iter)
{// (perde ayarları) //
HM^{mean}=mean(HM(N+1));
if (HM(d, N + 1)) $\le$ mean(HM(N + 1))
{
/ / (Group A, Perdeleri sıkılaştır veya gevşet) //
if rand [0, 1] $\le$ 0.5

$$y_i = x_i^{Best} - (x_i^{Best} - x_i)rand[0,1]$$

else

$$y_i = x_i^{Best} - (x_i^{Worst} - x_i)rand[0,1]$$

Endif
}
Else
{
/ / (Group B, Perdeleri sıkılaştır) //
m = int(1 + (N _ 1)/ rand [0, 1])

$$x_m^{Best} = x_m^{Best}(x_{i\_upper} / x_{m\_upper})$$

$$y_i = x_i' - (x_m^{Best} - x_i')rand[0,1]$$

}
Endif

$$if\ (x_{i\_lower} \le y_i \le x_{i\_upper})$$

x_i'= y_i
endif
}Endif // (perde ayarı) //
Else
{// (hafıza değişkenleri) //

$$x_i' = x_{i\_lower} + (x_{i\_upper} - x_{i\_lower})rand[0,1]$$

}Endif
}endfor
Step.4: Ahenk hafızasını yenile
Yeni ahenk vektörünün uygunluğunu hesapla.

$$if\ (f(x') \le f(x^{worst})$$

X^{worst}=x'
Solution=f(x^{best})
End procedure

Program 5.24.5 zeki ahenk arama programı

```
////////////////////////////////////////////////////////////////////////////////////////////////
//Referans : An Intelligent Tuned Harmony Search algorithm for optimisation
//Parikshit Yadav, Rajesh Kumar, S.K. Panda, C.S. Chang
```

```
// Programlayan : Oğuz Emrah Turgut
////////////////////////////////////////////////////////////////////////////////////////////////////

abstract class f_xj
{
abstract double func(double x[]);
}

public class zeki_ahenk_arama
{
int N;
double Lower[];
double Upper[];
int HMS;
double HMCR;
double PARmin;
double PARmax;
double PAR;
int Maxiter;
double best[];
double worst[];
double y[];
int d;
int m;
double[][] HM;
double[] fitness;
double[] xdot;
int indexmin;
double minval;
int indexmax;
double maxval;
double fitnessmean;
f_xj ff;
        public zeki_ahenk_arama(f_xj iff,double[] iLower,double[] iUpper,int iHMS,double iHMCR,int iMaxiter)
        {
                ff=iff;
                Maxiter=iMaxiter;
                Lower=iLower;
                Upper=iUpper;
                N=Upper.length;
                HMS=iHMS;
                HMCR=iHMCR;
                PARmin=0.0;
                PARmax=2.0;
                HM=new double[HMS][N];
                fitness=new double[HMS];
                best=new double[N];
                worst=new double[N];
                y=new double[N];
                xdot=new double[N];
                }

                public void initialize()
                { //IHTS algoritması giriş değerlerini ata
                        // iAhenk hafizasını tanımla(improvisation)
                 for(int i=0;i<HMS;i++)
                   {for(int j=0;j<N;j++)
                                {HM[i][j]=Lower[j]+(Upper[j]-Lower[j])*Math.random();}}

            // fonksiyon değerlerini hesapla
                                  for(int i=0;i<HMS;i++)
                                {fitness[i]=ff.func(HM[i]);}

                                // minimumu bul
                                double[] dd=getminval_index(fitness);
                                minval=dd[0];
                                indexmin=(int)dd[1];
                                for(int i=0;i<N;i++)
                                {best[i]=HM[indexmin][i];}

                                // maksimumu bul
                                 double[] dd2=getmaxval_index(fitness);
                                 maxval=dd2[0];
                                 indexmax=(int)dd2[1];
                                 for(int i=0;i<N;i++)
                                 {worst[i]=HM[indexmax][i];}
```

```
                }

        public double[] getminval_index(double[] a)
        {
        double m=0.0;
        double b[]=new double[a.length];
        for(int i=0;i<a.length;i++)
        {b[i]=a[i];}
        double minval=a[0];
        for(int i=0;i<a.length;i++)
        {if(a[i]<minval){minval=a[i];}}
        for(int i=0;i<a.length;i++)
        {if(b[i]==minval){m=i;break;}};
        double[] dep=new double[2];
        dep[0]=minval;
        dep[1]=m;
        return dep;
        }

        public double[] getmaxval_index(double a[])
        {
        double m=0.0;
        double b[]=new double[a.length];
        for(int i=0;i<a.length;i++)
        {b[i]=a[i];}
        double maxval=a[0];
        for(int j=0;j<a.length;j++)
        {if(a[j]>maxval){maxval=a[j];}}
        for(int i=0;i<b.length;i++)
        {if(b[i]==maxval){m=i;break;}}
        double dep2[]=new double[2];
        dep2[0]=maxval;
        dep2[1]=m;
        return dep2;
        }

                public double mean(double x[]) // ortalama vektör
                {
                        int n=x.length;double s1=0.0;
                        for(int i=0;i<n;i++)
                        {s1+=x[i];}
                        return s1/(double)n;
                        }

                public double[][] solution()
                {
                         initialize() ;
                         int iter=0;
                         while(iter<Maxiter)
                         {
                                 for(int i=0;i<N;i++)
                                 {
                                         if(Math.random()<HMCR)
                                         {
                                                 d=(int)(((double)HMS)*Math.random());
                                      xdot[i]=HM[d][i];
                                     PAR=PARmax-((PARmax-PARmin)*((double)iter)/((double)Maxiter));
                                                 if(Math.random()<PAR)
                                                 {
                                                         fitnessmean=mean(fitness);
                                                         if(fitness[d]<fitnessmean)
                                                         {
                                                         // Group A, perdeleri hep sıkılaştır hem gevşet
                                                                 if(Math.random()<=0.5)
                                                                 {
                                                                 y[i]=best[i]-(best[i]-HM[d][i])*
Math.random();

                                                                 }
                                                                 else
                                                                 {
                                                                 y[i]=best[i]-(worst[i]-

HM[d][i])*Math.random();

                                                                   }
                                                           }
                                                          else
                                                         { // perdeleri hep sıkılaştır
```

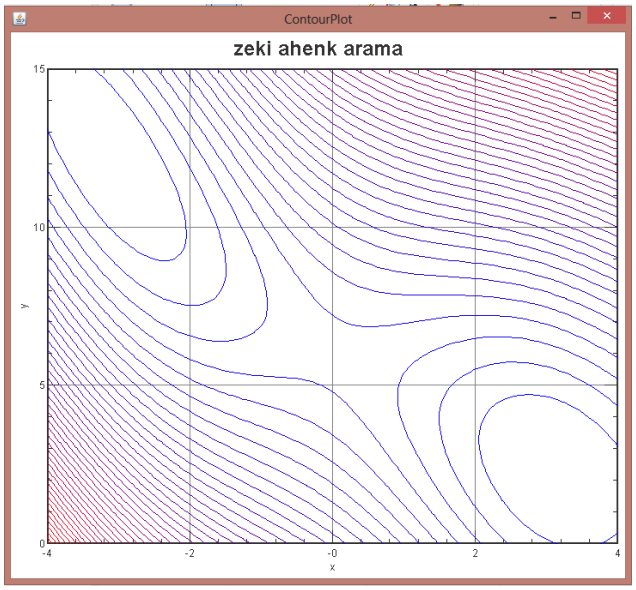
```
                                                    m=(int)(((double)N)*Math.random());
                                                    y[i]=xdot[i]+(best[m]-xdot[i])*Math.random();
                                                   }

                                                        if((Lower[i]<=y[i])&&(Upper[i]>y[i]))
                                                        {xdot[i]=y[i];}

                                                       }
                                            }
                                                else
                                                {
                                                xdot[i]=Lower[i]+(Upper[i]-Lower[i])*Math.random();

                                            }
                                    }
                                    if(ff.func(xdot)<=ff.func(worst))
                                    {
                                            for(int i=0;i<N;i++)
                                            {worst[i]=xdot[i];}
                                    }
                                    for(int i=0;i<HMS;i++)
                            {fitness[i]=ff.func(HM[i]);}

                            // minimum indeks ve en iyi çözümü bul
                            double[] dd=getminval_index(fitness);
                            minval=dd[0];
                            indexmin=(int)dd[1];
                            for(int i=0;i<N;i++)
                            {best[i]=HM[indexmin][i];}

                            // maksimum indeks ve en kötü çözümü bul
                             double[] dd2=getmaxval_index(fitness);
                             maxval=dd2[0];
                             indexmax=(int)dd2[1];
                             for(int i=0;i<N;i++)
                             {HM[indexmax][i]=worst[i];}

                             iter++;
                        }
                        double[][] dep=new double[2][N];
                        dep[0][0]=minval;
                        for(int i=0;i<N;i++)
                        {dep[1][i]=best[i];}
         return dep;
}
        public double[] optimum()
        {   double[][] out=solution();
                    return out[1];
                }

                        void toString(double[][] out)
                        {
                         System.out.println("Optimum değer = "+out[0][0]);
                         for(int i=0;i<N;i++)
                         {System.out.println("x["+i+"] = "+out[1][i]);}
                        }
                        void toString(double[] out)
                        {
                         for(int i=0;i<N;i++)
                         {System.out.println("x["+i+"] = "+out[i]);}
                        }
                        }
```

Program 5.24.6 zeki ahenk arama test programı

```
class f7 extends f_xj// Branin  f(x)=0.397887  @x=(-pi,12.275),(pi,2.275),(9.42478,2.475)   -5.0<=x[0]<=10.0,
0.0<=x[1]<=15.0
{
double func(double x[])
{
double first=0.0;
first=((x[1]-(5.1*x[0]*x[0]/(4.0*Math.PI*Math.PI))+(5.0*x[0]/Math.PI)-6.0)*(x[1]-
(5.1*x[0]*x[0]/(4.0*Math.PI*Math.PI))+(5.0*x[0]/Math.PI)-6.0))+(10.0*(1.0-
```

```
(1.0/(8.0*3.1415)))*Math.cos(x[0]))+10.0;
return first;
}
}

public class zeki_ahenk_arama_testi
{
	public static void main(String args[])
		{
			double[] xmin={-10.0,-10.0};
			double[] xmax={20.0,20.0};
			int HMS=100;
			double HMCR=0.99;
			int Maxiter=50000;
			f7 ff=new f7();
			zeki_ahenk_arama iths=new zeki_ahenk_arama(ff,xmin,xmax,HMS,HMCR,Maxiter);
			double out[][]=iths.solution();
			iths.toString(out);
			//Nelder-Mead ile değerleri kontrol edelim
			double r1[]= suruelemanlari.nelder(ff,out[1]);
			System.out.println("Nelder mead metodu optimum değer : "+ff.func(r1));
			iths.toString(r1);
			ContourPlot pp = new ContourPlot(ff,-4,4,100,0,15,100,40, false,"zeki ahenk arama",
"x", "y");
			pp.plot();
			}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" zeki_ahenk_arama_testi
Optimum değer = 0.4036672982519569
x[0] = -3.167222987220524
x[1] = 12.387812886787845
Nelder mead metodu optimum değer : 0.39789909279006785
x[0] = -3.141592658279003
x[1] = 12.274999994445498
```



(bolum5_040.jpg,Resimaltı **:zeki ahenki arama test fonksiyonu contour grafiği)**

5.25 Global tesadüfi metodlar : Guguk kuşu arama metodu

Bu algoritma Yang and Deb[56] tarafından oluşturulmuştur. Temel mekanizma guguk kuşlarının geliştirdiği ilginç bir çoğalma stratejisine dayanmaktadır. Guguk kuşları yumurtalarını diğer kuşların yuvalarına bırakırlar. Eğer bu kuşlar yumurtanın yabancı olduğunu keşfederlerse ya yuvayı terkederler, ya da yumurtayı atarlar. Keşfedilmenin engellenmesi için yumurtaların bırakılan yumurtaya mümkün olduğunca benzemesini sağlayacak bir evrimden geçmişlerdir. Bu kavram içinde alttaki kuralları belirleyebiliriz.

- Guguk kuşları her sefer sadece bir yumurta yumurtlama hakına sahiptirler ve yumurtayı tesadüfi bir yuvaya bırakırlar.
- Yüksek kaliteli yumurtaları barındıran yuvalar glecek neslin oluşmasını sağlarlar (en iyi uyum)
- Toplam yuva sayısı sınırlıdır ve yuva sahibi kuşun belli bir olasılıkla, $p_a \in [0,1]$, yabancı yumurtayı keşfetmesi mümkündür. Eğer bu durum olursa yabancı yumurta atılacak veya yuva başka bir yerde tekrar oluşturulacaktır.

Bu temel kurallar kullanılarak guguk kuşu arama algoritması oluşturulmuştur. Bu algoritmanın verimli çalışabilmesi için " Lévy uçuşu" adını verdiğimiz bir istatistiksel kavram kullanılmıştır. Yerel ve global arama işlemi "Lévy uçuşu" ile yapılmaktadır. Lévy uçuşu bir rasgele yürüme algoritmasıdır. Her adım tesadüfi olarak atılır, ancak başlangıç noktası bir önceki adımdır. Levy uçuşu şu şekilde tanımlanabilir.

$$x_i^{t+1} = x_i^t + \alpha \Theta Lévy(\lambda)$$

Bu denklemde Lévy(λ) Levy tesadüfi dağılım fonksiyonudur. Θ genişleme daralma kararı alan akıllı bir çarpım faktörü, α adım genişliğidir. " Lévy uçuşu" yardımıyla yeni bir yumurta yerleştirildiğinde daha önceki pozisyonuna göre daha iyi olup olmadığı kararı oluşturulur. Bir çok araştırmada adım genişliği α parametresinin 1 olarak alınabileceği bildirilmiştir.

```
Optimizasyon fonksiyonu f(x), x = (x1, ..., xd) ;
Yuvalardaki orijinal yumurtalar  xi (i = 1, 2, ..., n);
while (t <MaxksimumNesil) or (durdurmakriteri);
Guguk kuşu (i) Lévy uçuşuyla tesadüfi uçsun;
Kaliteyi kontrol et/i noktasında fonksiyon değeri Fi;
n yuvadan birini tesadüfi olarak seç(örneğin  j);
if (Fi > Fj),
yuva j yi yeni çözümle değiştir;
end
en kötü (pa) yuvayı terket ve
[Lévy uçuşu kullanarak yeni bir noktada yeni yuva oluştur];
En iyi çözümleri sakla (veya kaliteli yuvaları gözönünde bulundur);
Bütün çözümleri karşılaştır ve en iyi çözümü sapta;
end while
sonuçları değerlendir;
```

Bu algoritma kullanılarak java kodlarımız geliştirilmiştir.

Program 5.25.1 Guguk kuşu arama programı

```
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.Random;
import java.util.ArrayList;

////////////////////////////////////////////////////////////////////////
// Enginering Optimisation by Cuckoo Search       //
//                                                //
//          Xin-She Yang          Suash Deb       //
////////////////////////////////////////////////////////////////////////
// Programmed by : Oğuz Emrah Turgut

//REFERANCE
```

```
//Papers -- Citation Details:
// 1) X.-S. Yang, S. Deb, Cuckoo search via Levy flights,
// in: Proc. of World Congress on Nature & Biologically Inspired
// Computing (NaBIC 2009), December 2009, India,
// IEEE Publications, USA, pp. 210-214 (2009).
//http://arxiv.org/PS_cache/arxiv/pdf/1003/1003.1594v1.pdf
// 2) X.-S. Yang, S. Deb, Engineering optimization by cuckoo search,
//Int. J. Mathematical Modelling and Numerical Optimisation,
//Vol. 1, No. 4, 330-343 (2010).
//http://arxiv.org/PS_cache/arxiv/pdf/1005/1005.2908v2.pdf
abstract class f_xj
{
abstract double func(double x[]);
}

public class gugukkusu_arama
{
int n;
int nd;
double pa;
double tol;
double nest[][];
double xmin[];
double xmax[];
f_xj ff;

    public gugukkusu_arama(f_xj iff,int in,double ipa,double itol,double ixmin[],double ixmax[])
    {
            n=in;
            nd=ixmin.length;
            pa=ipa;
            tol=itol;
            nest=new double[n][nd];
            xmin=new double[nd];
            xmax=new double[nd];
            ff=iff;
            double[] one=ones();
            for(int i=0;i<nd;i++)
            {xmin[i]=ixmin[i];xmax[i]=ixmax[i];}
            }

            double[] randn()
            {
            double a[]=new double[nd];
            for(int i=0;i<nd;i++)
            {a[i]=Math.random();}
            return a;
            }

            double[] ones()
    {
            double v[]=new double[nd];
            for(int i=0;i<nd;i++)
            {v[i]=1.0;}
            return v;
            }

            double [][][] initialize()
            {
            for(int i=0;i<n;i++)
```

```
            {for(int j=0;j<nd;j++)
               {nest[i][j]=xmin[j]+((xmax[j]-xmin[j])*Math.random());}
            }
            double[] fitness=new double[n];
            for(int i=0;i<n;i++)
            {fitness[i]=Math.pow(10,7);}
            double dp [][][]=new double [2][n][nd];
            for(int i=0;i<n;i++)
            {  for(int j=0;j<nd;j++)
              {dp[0][i][j]=nest[i][j];}
            }
            for(int i=0;i<n;i++)
            {dp[1][i][0]=fitness[i];}
            return dp;
            }

            double logGamma(double x)
            {
    double tmp =(x-0.5)*Math.log(x+4.5)-(x+4.5);
    double ser = 1.0 + 76.18009173    / (x + 0)   - 86.50532033    / (x + 1)+ 24.01409822    / (x + 2)   -
1.231739516   / (x + 3)+  0.00120858003 / (x + 4)   -  0.00000536382 / (x + 5);
    return tmp + Math.log(ser * Math.sqrt(2 * Math.PI));
    }
    double gamma(double x) { return Math.exp(logGamma(x)); }


   double[] getminval_index(double[] a)
            {
            double m=0.0;
            double b[]=new double[a.length];
            for(int i=0;i<a.length;i++)
            {b[i]=a[i];}
            double minval=a[0];
            for(int i=0;i<a.length;i++)
            {if(a[i]<minval){minval=a[i];}}
            for(int i=0;i<a.length;i++)
            {if(b[i]==minval){m=i;break;}};
            double[] dep=new double[2];
            dep[0]=minval;
            dep[1]=m;
            return dep;
            }

            double[][][] get_best_nest(double[][] nest,double[][] newnest,double[] fitness)
            {
            double fnew=0.0;
              for(int j=0;j<n;j++)
              {
                fnew=ff.func(newnest[j]);
                if(fnew<fitness[j])
                {fitness[j]=fnew;nest[j]=newnest[j];}
              }
              double[] dep=getminval_index(fitness);
              double fmin=dep[0];
              int index=(int)dep[1];
              double[] best=new double[nd];
              for(int i=0;i<nd;i++)
              {best[i]=nest[index][i];}
              double depp[][][]=new double[4][n][nd];
              depp[0][0][0]=fmin;
```

```
            for(int i=0;i<n;i++)
            {depp[1][i][0]=fitness[i];}
            for(int i=0;i<nd;i++)
            {depp[2][0][i]=best[i];}
            for(int i=0;i<n;i++)
            {for(int j=0;j<nd;j++)
               {depp[3][i][j]=nest[i][j];}
            }
            return depp;
          }
          // Random permutation of matrix
          double[][] randperm(double a[][])
          {
          double y[][]=new double[a.length][a[0].length];
          double[] sw=new double[a[0].length];
          for(int i=0;i<a.length;i++)
          {y[i]=a[i];}
          int m=a.length;
          int rp=0;
          for(int i=0;i<m-1;i++)
                    {rp=(int)((m-i)*Math.random()+i);sw=y[i];
                    y[i]=y[rp];y[rp]=sw;
                    }
          return y;
          }
          double[][] get_cuckoos(double[][] nest,double best[])
          {
          double[] s=new double[nd];
          double[] u=new double[nd];
          double[] v=new double[nd];
          double[] step=new double[nd];
          double[] stepsize=new double[nd];
          double beta=1.5;
          double
sigma=Math.pow((gamma(1+beta)*Math.sin(3.1415*beta/2.0)/(gamma((1.0+beta)/2.0)*beta*Math.pow(2.0,(beta-
1.0)/2.0))),(1.0/beta));
          double[] sk=new double[nd];
          Random rndm=new Random();
          for(int j=0;j<n;j++)
          {
          s=nest[j];
            for(int i=0;i<nd;i++)
            {u[i]=(rndm.nextGaussian())*sigma;
                    v[i]=(rndm.nextGaussian());
                    step[i]=u[i]/Math.pow((Math.abs(v[i])),(1/beta) );
                    stepsize[i]=0.01*step[i]*(s[i]-best[i]);
                    }
                    for(int k=0;k<nd;k++)
                    {sk[k]=stepsize[k]*rndm.nextGaussian();
                     s[k]=sk[k]+s[k];
                    }
                    nest[j]=simplebounds(s);
          }
          return nest;
          }
          double[] simplebounds(double s[])
          {
            for(int i=0;i<nd;i++)
            {if(s[i]<xmin[i])
             {s[i]=xmin[i];}
```

```
  if(s[i]>xmax[i])
  {s[i]=xmax[i];}
 }
 return s;
}
double[][] emptynests(double[][] nest)
{
        double[][] K=new double[n][nd];
   double[][] randmat=new double[n][nd];
        double[] perm1=new double[n];
        double[] perm2=new double[n];
        double[][] nest1=new double[n][nd];
        double[][] nest2=new double[n][nd];
        double[][] nest3=new double[n][nd];
        double[][] nest4=new double[n][nd];
        double[][] stepsize=new double[n][nd];
   double[][] randmat2=new double[n][nd];
        double[][] newnest=new double[n][nd];
   for(int i=0;i<n;i++)
   {for(int j=0;j<nd;j++)
          {randmat[i][j]=Math.random();
          if(randmat[i][j]<pa){K[i][j]=0.0;}
          if(randmat[i][j]>pa){K[i][j]=1.0;}
          }
        }
        for(int i=0;i<n;i++)
        {for(int j=0;j<nd;j++)
          {nest1[i][j]=nest[i][j];nest2[i][j]=nest[i][j];}}
        nest3=Matrix.substract(randperm(nest1),randperm(nest2));
        for(int i=0;i<n;i++)
        {for(int j=0;j<nd;j++)
         {stepsize[i][j]=nest3[i][j]*Math.random();}}
         nest4=Matrix.dotproduct(stepsize,K);
         newnest=Matrix.add(nest,nest4);
         return newnest;
}
void solution()
{
double nestt[][]=new double[n][nd];
double fitness[]=new double[n];
double bestnest[]=new double[nd];
double best[]=new double[nd];
double newnest[][]=new double[n][nd];
double d1[][][]=initialize();
for(int i=0;i<n;i++)
{for(int j=0;j<nd;j++)
  {nestt[i][j]=d1[0][i][j];}
}
for(int i=0;i<n;i++)
{fitness[i]=d1[1][i][0];}
double d2[][][]=get_best_nest(nestt,nestt,fitness);
double fmin=d2[0][0][0];
  for(int i=0;i<n;i++)
  {fitness[i]=d2[1][i][0];}
  for(int i=0;i<nd;i++)
  {bestnest[i]=d2[2][0][i];}
        for(int i=0;i<n;i++)
        {   for(int j=0;j<nd;j++)
          {nestt[i][j]=d2[3][i][j];}}
        int iter=0;
```

```
                    double fnew=0.0;
                    nestt=emptynests(nestt);
                    //System.out.println(Matrix.toString(nestt));
                    while(fmin>tol)
                    {
                            newnest=get_cuckoos(nest,bestnest);
                            double d3[][][]=get_best_nest(nestt,newnest,fitness);
                fnew=d3[0][0][0];
                for(int i=0;i<n;i++)
                {fitness[i]=d3[1][i][0];}
                for(int i=0;i<nd;i++)
                {best[i]=d3[2][0][i];}
                        for(int i=0;i<n;i++)
                        {for(int j=0;j<nd;j++)
                          {nestt[i][j]=d3[3][i][j];}
                        }
                        iter+=n;
                            newnest=emptynests(nestt);
                            //System.out.println(Matrix.toString(nestt));
                            double d4[][][]=get_best_nest(nestt,newnest,fitness);
                 fnew=d4[0][0][0];
                for(int i=0;i<n;i++)
                {fitness[i]=d4[1][i][0];}
                for(int i=0;i<nd;i++)
                {best[i]=d4[2][0][i];}
                        for(int i=0;i<n;i++)
                        {for(int j=0;j<nd;j++)
                          {nestt[i][j]=d4[3][i][j];}
                        }
                            iter+=n;
                            if(fnew<fmin)
                            {fmin=fnew;bestnest=best;}
                    }
                    System.out.println("fonksiyonun optimum değeri = "+fmin);
                    for(int i=0;i<bestnest.length;i++)
                    {System.out.println("x["+i+"] = "+bestnest[i]);}
            }
}
```

Program 5.25.2 Guguk kuşu arama test programı

```
class fa extends f_xj
{
double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=(1.0-x[0])*(1.0-x[0])+100*(x[1]-x[0]*x[0])*(x[1]-x[0]*x[0]);
return ff; //maksimum testi
}}

public class gugukkusu_arama_testi
{
public static void main(String args[])
{
fa ff=new fa();
int n=50;//Number of nests
double pa=0.25;// Discovery rate of alien eggs/solutions
double tol=1e-5;//minimum values of tolerance leads to better results
double xmin[]={0.0,0.0};//Lower bounds
double xmax[]={2.0,2.0};//Upper bounds
```

```
gugukkusu_arama cc=new gugukkusu_arama(ff,n,pa,tol,xmin,xmax);
cc.solution();
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" gugukkusu_arama_testi
fonksiyonun optimum değeri = 9.517096302689688E-6
x[0] = 1.0025499488817973
x[1] = 1.0049327665670904

> Terminated with exit code 0.
```

5.26 Global tesadüfi metodlar : av arama metodu

Bu global tesadüfi optimizasyon metodu bir hayvan gurubunun birlikte avlanmasını simule eder. Çeşitli avcı hayvanların avlarını yakalamak için kullandıkları çeşitli stratejiler mevcuttur. Örneğin kurtlar ve aslanlar avlarının etrafında bir çember oluşturarak avlamaya çalışırlar. Gerçek avda avlana hayvan dinamiktir, hareket eder. Bu yüzden avcılar sürekli olarak avlarının yeni pozisyonunu saptayarak pozisyonlarını düzeltir. Matematiksel optimizasyon probleminde optimumun yeri sabittir. Bu yüzden statik pozisyondan dolayı, daha kolay bir problemdir. Matematiksel optimizasyonda suni avcılar lidere(en iyi çözümü veren sete) doğru hareket ederler. Bu esnada herhangi bir suni avcının pozisyonu liderden daha iyi duruma gelirse bu avcı lider olur. Lidere doğru harekette bu tür değişimlere göre her avcı pozisyonunu değiştirir. Bunu gurup avlama ayar oranı (HGCR) parametresini kullanarak gerçekleştireceğiz. Algoritma adımları şu şekilde gerçekleştirilecektir:

1. Adım : Parametrelerin ilk değerlerinin tanımlanması
2. Avcı gurubunun oluşturulması
3. Lidere doğru hareket
4. Pozisyonun değiştirilmesi (eğer gerekliyse)
5. Avcı gurubunun tekrar organize edilmesi
6. 3 4 ve 5inci adımların maksimum iterasyon sayısına kadar tekrarı

1.Adım İlk önce alt ve üst limitler avın etrafında bir daire oluşturaak şekilde oluşturulurlar. Avcı gurubu boyutu(HGS), Lidere doğru maksimum hareket sayısı, Avcı gurubu gurup avlama oranı(HGCR) tanımlanır.

2.Adım

Avcı gurubualt ve üst liitlere bağlı kalarak tesadufi pozisyonlara yerleştirilir. Fonksiyon değerleri her avcı(pozisyon) için hesaplanır

3.Adım

Lidere doğru hareket edilerek yeni bir çözüm vektörü oluşturulur. $x^{new} = (x_1, x_2, ..., x_i, ..x_N)$. Lidere doğru hareket $x_i^{new} = x_i + rand \times MML \times (x_i^{Leader} - x_i)$ formülü ile hesaplanır. Burada rand 0-1 arasında tesadüfi bir sayıdır.

MML parametresi çeşitli durumlara göre değişebilen bir parametredir. Daha sonraki adımlarda detaylı tanımını vereceğimiz "epochs" parametresine göre değişir. Epochs parametresi büyük iterasyon belirtiyorsa MML değerinin 0.05 civarında olması önerilebilir. Küçük iterasyon sayılarında bu değer 0.4 civarında alınabilir.

4. Adım

Bu adımda posizyon düzeltmelerinin etkileri artarak bu stepte bir avlanma organizasyonuna dönüşür. Lidere doğru hareket eden avcılar bundan sonra çözümlerini iki türlü gerçekleştirirler 1) gerçek değer düzeltmesi 2) digital değer düzeltmesi. Burada kullandığımız algoritmada gerçek değer düzeltmesi kullanılmıştır. Yeni pozisyon olarak mevcut pozisyonlardan biri HGCR seçilebilir veya

New hunter position is selected from HG or real value correction by HGCR parameter which varies from 0.0 to 1.0(proposed value lies between 0.1 – 0.4). Değerin yenilenme prosesi matematiksel olarak şu şekilde tanımlanabilir:

$$x_i^j \leftarrow \begin{Bmatrix} x_i^j \in \{x_i^1, x_i^2, x_i^3, ..., x_i^{HGS}\} \text{ with probability HGCR i=1,...,N , j=1,...,HGS} \\ x_i^j = x_i^j \pm Ra \text{ with probability (1.0-HGCR)} \end{Bmatrix}$$

Ra dizayn değişkenleri arasındaki suni mesafedir ve optimizasyon prosesi sırasında sürekli yenilenir. Ra matematiksel olarak exponensiyel bir fonksiyon olarak tanımlanır.

$$Ra = Ra_{\min}[\max(x_i) - \min(x_i)]\exp\left(\frac{Ln\left(\frac{Ra_{\min}}{Ra_{\max}}\right)\times iter}{\max iter}\right)$$

Bu tanımda iter iterasyon sayısını, max(i) ve min(i) , (x_i). avcı vektörünün maksimum ve minimum değer alan koordinatlarını belirler. Ra_{min} ve Ra_{max} suni göreceli arama yarıçapıdır. Maxiter maksimum iterasyon sayısıdır.

5. Adım

İterasyon prosesi sırasında avcı yerel extremuma takılıp kalabilir. Bunu engellemek için avcıların yeri yeni bir optimum nokta bulmaları için değiştirilir. Belli bir iterasyon sayısından sonra avcı konumunu değiştirir. Epoch adını verdiğimiz yeni bir terim bu gaye için kullanılır. Epoch terimi yer değiştirmeden önce ne kadar aynı noktada kalınacağını belirler. Yer değiştirme prosesi

$$x_i^{new} = x_i^{Leader} \pm rand \times [\max(x_i) - \min(x_i)] \times \alpha \exp(-\beta \times EN)$$

Şeklinde oluşturulur. Burada α ve β algoritmanın dönüşümünü belirleyen gerçek rakamlardır. Birden fazla yerel optimumun olduğunu düşündüğümüz problemlerde α'nın büyük değerleri ve β'nın küçük değerleri kullanılır. EN sayıcı bir değişkendir ve en fazla epoch değişkenine kadar sayar. Bu algoritmaya göre geliştirilen java programı altta verilmiştir.

Program 5.26.1 avcı arama programı

```
// Programlayan: Oğuz Emrah Turgut
abstract class f_xj
{abstract double func(double x[]);}
public class av_arama
{
int N;
int HGS;//guruptaki çözüm vektörü sayısı
double MML;//lidere doğru maksimum hareket sayısı
double HGCR;//avcı gurubu oranı
double HG[][];
double Lower[];
double Upper[];
double dummyHG[][];
double fitness[];
double best[];
double worst[];
double alfa;
double beta;
double Ramax;
double Ramin;
int maxiter;
int epochiter;
double min;
double max;
int indexmin;
int indexmax;
int d;
double EN;
double[] ENcount;
f_xj ff;

    public av_arama(f_xj iff,int iHGS,double iMML,double iHGCR,double iLower[],double[] iUpper,double
ialfa,double ibeta,double iRamax,double iRamin,int imaxiter,int iepochiter)
    {
            ff=iff;
            HGS=iHGS;
            HGCR=iHGCR;
            MML=iMML;
```

```
 Ramax=iRamax;
 Ramin=iRamin;
 Lower=iLower;
 Upper=iUpper;
 epochiter=iepochiter;
 maxiter=imaxiter;
 N=Upper.length;
 HG=new double[HGS][N];
 dummyHG=new double[HGS][N];
 fitness=new double[HGS];
 best=new double[N];
 worst=new double[N];
 alfa=ialfa;
 beta=ibeta;
 EN=(double)maxiter/((double)HGS*(double)epochiter);
 EN=Math.ceil(EN);

  //System.out.println(EN);
  ENcount=new double[(int)EN];
  for(int i=0;i<(int)EN;i++)
  {ENcount[i]=i+1;}
 // System.out.println(Matrix.toString(ENcount));
}

double[] getminval_index(double[] a)
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double minval=a[0];
for(int i=0;i<a.length;i++)
{if(a[i]<minval){minval=a[i];}}
for(int i=0;i<a.length;i++)
{if(b[i]==minval){m=i;break;}};
double[] dep=new double[2];
dep[0]=minval;
dep[1]=m;
return dep;
}

double[] getmaxval_index(double a[])
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double maxval=a[0];
for(int j=0;j<a.length;j++)
{if(a[j]>maxval){maxval=a[j];}}
for(int i=0;i<b.length;i++)
{if(b[i]==maxval){m=i;break;}}
double dep2[]=new double[2];
dep2[0]=maxval;
dep2[1]=m;
return dep2;
}

void init()
{
```

```
    for(int i=0;i<HGS;i++)
    {
            for(int j=0;j<N;j++)
            {HG[i][j]=Lower[j]+(Math.random()*(Upper[j]-Lower[j]));}
          }

          //uyum fonksiyonunu hesapla
          for(int i=0;i<HGS;i++)
          {fitness[i]=ff.func(HG[i]);}

          // minimum index ve en iyi çözüm
          double[] dd=getminval_index(fitness);
          indexmin=(int)dd[1];
          for(int i=0;i<N;i++)
          {best[i]=HG[indexmin][i];}

          // maksimum indeks
          double[] dd2=getmaxval_index(fitness);
          indexmax=(int)dd2[1];
          for(int i=0;i<N;i++)
          {worst[i]=HG[indexmax][i];}
}

double max(double x[])
{
 int n=x.length;
 double maxv=x[0];
 for(int i=0;i<n;i++)
 {if(x[i]>maxv)
          {maxv=x[i];}}
 return maxv;
}

double min(double x[])
{
 int n=x.length;
 double minv=x[0];
 for(int i=0;i<n;i++)
 {if(x[i]<minv)
   {minv=x[i];}}
 return minv;

}
double[][] iterations()
{
          init();
          //System.out.println(Matrix.toString(HG));
          int iter=1;
          while(iter<maxiter)
          {

          for(int i=0;i<HGS;i++)
          {for(int j=0;j<N;j++)
            {dummyHG[i][j]=HG[i][j]+(2.0*(Math.random()-0.5)*MML*(best[j]-HG[i][j]));}}

          for(int i=0;i<HGS;i++)
          {
            if(ff.func(dummyHG[i])<ff.func(HG[i]))
            {
                    for(int j=0;j<N;j++)
```

```
                                {HG[i][j]=dummyHG[i][j];}
                            }
                        }

                        //Calculate fitness
                        for(int i=0;i<HGS;i++)
                        {fitness[i]=ff.func(HG[i]);}

                        //minimum indeks ve en iyi çözüm
                        double[] dd=getminval_index(fitness);
                        indexmin=(int)dd[1];
                        for(int i=0;i<N;i++)
                        {best[i]=HG[indexmin][i];}


                        for(int i=0;i<HGS;i++)
                        {
                          if(Math.random()<HGCR)
                          {
                            d=(int)(((double)HGS)*Math.random());
                            for(int j=0;j<N;j++)
                            {HG[i][j]=HG[d][j];}
                          }
                          else
                          {
                                    for(int j=0;j<N;j++)
                                    {
                                    HG[i][j]=HG[i][j]+(2.0*(Math.random()-0.5))*Ramin*(max(HG[i])-
min(HG[i]))*Math.exp(Math.log(Ramin/Ramax)*(double)iter/(double)maxiter);
                              }
                                  }
                        }
                        int tt=(int)((double)iter/((double)HGS*(double)epochiter));
                        for(int i=0;i<HGS;i++)
                        {
                           for(int j=0;j<N;j++)
                           {HG[i][j]=best[j]+(2.0*(Math.random()-0.5))*(max(HG[i])-min(HG[i]))*alfa*Math.exp(-
beta*ENcount[tt]);                  }
                        }

                        //uyumu hesapla
                        for(int i=0;i<HGS;i++)
                        {fitness[i]=ff.func(HG[i]);}

                        // minimum indeks ve en iyi çözüm
                        double[] dd3=getminval_index(fitness);
                        min=dd3[0];
                        indexmin=(int)dd3[1];
                        for(int i=0;i<N;i++)
                        {best[i]=HG[indexmin][i];}

                        iter++;
                        }
                        double[][] out=new double[2][N];
                        for(int i=0;i<N;i++)
                        {out[0][i]=best[i];}
                        out[1][0]=min;
                        return out;
}
            public void toStringnew()
```

```
        {
            double[][] in=iterations();
            System.out.println("Optimum değer = "+in[1][0]);
            for(int i=0;i<N;i++)
            {System.out.println("x["+i+"] = "+in[0][i]);}
           }
}
```

Program 5.26.2 avcı arama test programı

```
class f9 extends f_xj// Easom  f(x)=-1.0  @x=(pi,pi)   -100.0<=x[i]<=100.0 i=0,1
{
double func(double x[])
{
double first=0.0;
first=-Math.cos(x[0])*Math.cos(x[1])*Math.exp(-(x[0]-Math.PI)*(x[0]-Math.PI)-(x[1]-Math.PI)*(x[1]-
Math.PI));
return first;
}}
public class av_arama_testi
{
    public static void main(String args[])
        {
            int maxiter=20000;
            int epochiter=40;
            int HGS=100;
            double MML=0.3;
            double HGCR=0.2;
            double[] Lower={-5.0,-5.0};
            double[] Upper={5.0,5.0};
            double alfa=-0.1;
            double beta=1.0;
            double Ramax=1.0e-2;
            double Ramin=1.0e-7;
            f9 ff=new f9();
            av_arama hunt=new
av_arama(ff,HGS,MML,HGCR,Lower,Upper,alfa,beta,Ramax,Ramin,maxiter,epochiter);
            hunt.toStringnew();
        }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" av_arama_testi
Optimum değer = -0.9999999999999999
x[0] = 3.1415926477365184
x[1] = 3.141592647891917

> Terminated with exit code 0.
```

5.27 Global tesadüfi metodlar : yerçekimi arama metodu

Yerçekimi arama metodu Newton'un çekim yasasına dayanan bir istatiksel arama metodudur. Kütlesi M_1 ve M_2 olan iki cisim için Newtonun çekim kanunu

$F = G\frac{M_1 M_2}{R^2}$ şeklinde ifade edilir. Buradaki R cisimler arasındaki mesafe, G çekim sabiti adını alır.

Newtonun ikinci kanununda M kütlesindeki bir cismin a ivmesiyle hareket ettiğinde oluşacak kuvvetin F olduğunu söyler.

$F = Ma$

Eğer iki cismin arasındaki mesafe artıyorsa, bu aralarındaki çekim kuvvetinin azalması anlamına gelir. Çekim kuvvetinin azalması aynı anda aralarındaki çekim sabitininde zaman içinde değişeceği anlamına gelir.

$$G(t) = G(t_0)\left[\frac{t_0}{t}\right]^{\beta} \qquad \beta < 1$$

Bu tanımlardan yola çıkarak teorik fizik üç türlü kütle tanımı yapar.

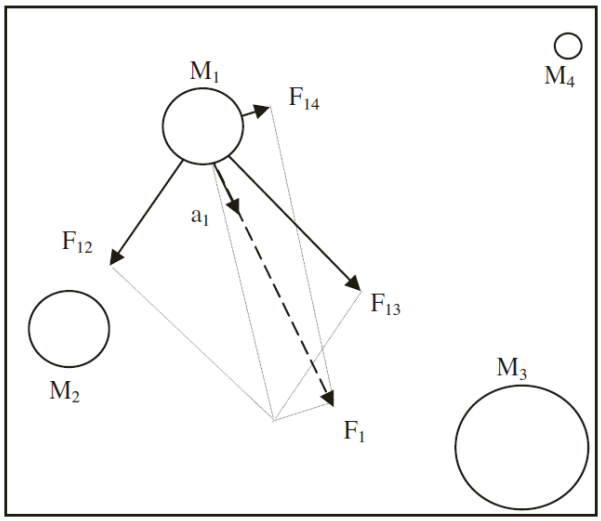
Aktif çekimli kütle: M_a belli bir nesnenin çekim alanının bu nesne tarafından oluşturulmuş gücüdür. Daha küçük kütleli bir cismin oluşturduğu çekim gücü daha büyük kütleli bir cismin oluşturduğuna göre daha azdır.

Pasif çekimli kütle: M_p bir çekim alanının etkisinde kalan bir cismin oluşturduğu kuvvet etkisidir.

Atalet kütlesi: M_i bir cismin kuvvet altında hareket etmeye karşı oluşturduğu direnç kuvvetidir. Bu tanımlar ışığında Newton yasasını aşağıdaki gibi tekrar yazabiliriz.

$$F_{ij} = G\frac{M_{aj}M_{pi}}{R^2} \qquad a_i = G\frac{F_{ij}}{M_{ii}}$$

Burada M_{aj} j kütlesinin aktif çekim kütlesi ve M_{pi} i kütlesinin pasif çekim kütlesini bildiri. Ve M_{ii} i atalet kütlesidir. Bir çok kütleden oluşan sistemlerde çekim tüm bu kütlelerin oluşturduğu bileşke kuvvetler şeklinde gerçekleşir.



(bolum5_041.jpg,Resimaltı **:çeşitli kütlelerin çekimsel etkileşimleri)**

Şimdi bu kavramı kullanarak çekim arama algoritmasını nasıl oluşturabileceğimizi irdeleyelim. N kütleli bir sistemi göz önüne alalım.

$X_i = (x_i^1, ..., x_i^d, ..., x_i^n) \quad i = 1,2,...N$ burada X_i^d i'inci kütlenin yerini belirtir. Belirli bir t zamanında j kütlesinin i kütlesine etkilediği kuvveti

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t)M_{aj}(t)}{R_{ij}^{\ 2} + \varepsilon}(X_j^d(t) - X_i^d(t))$$

İfadesiyle yazabiliriz. Burada ε küçük bir sabit, M_{aj} j'nin aktif çekimsel kütlesi, M_{pi} i'nin pasif çekimsel kütlesi, G(t) zamana bağlı çekimi değişkenidir R_{ij} ütleler arasındaki N boyutlu mesafe olup 2 normuyla hesaplanabilir.

$$R_{ij}(t) = \left\| X_i(t), X_j(t) \right\|_2$$

Algoritmada istatistiksel karekter oluşturabilmek için d boyutunda i kütlesine etki eden kuvveti

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^d(t)$$

İfadesiyle hesaplayacağız. Buradaki $rand_j$ tesadüfi sayıdır(0-1). Aynı şekilde ivme içinde

$$a_i^d(t) = \frac{F_i^d}{M_{ii}(t)}$$

Şeklinde hesaplanır. Buradan yeni hız ve konum vektörlerine kolayca geçebiliriz.

$$v_i^d(t+1) = rand_i\, v_i^d(t) + a_i^d(t)$$

$$X_i^d(t+1) = X_i^d(t) + v_i^d(t+1)$$

Çekim ve atalet kütleleri temel olarak fonksiyon değerlerinden hesaplanır. Daha büyük fonksiyon değerleri daha büyük kütlelere karşı gelir. Daha büyük kütle daha fazla çekim alanı yaratır.

$$M_{ai} = M_{pi} = M_i, \quad i = 1,2,....,N$$

$$m_i(t) = \frac{uyum_i(t) - en\_kötü(t)}{en\_iyi(t) - en\_kötü(t)}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)}$$

Bu bilgiler ışığında algoritmamızı şu şekilde tanımlayabiliriz.

a) Arama sınırlarını ve fonksiyonunu belirle
b) Tesadüfi olarak kütle pozisyonlarını (koordinatları) oluştur
c) Fonksiyonları değerlendirerek kütleleri hesapla
d) G(t), en_iyi(t), en_kötü(t) ve $M_i(t)$, i=1,..,N değerlerini oluştur
e) Değişik yönlerdeki toplam kuvvetleri hesapla
f) İvme ve hızları hesapla
g) Pozisyonları yenile
h) C den deye durdurma kriterine ulaşıncaya kadar tekrarla
i) Bitir

Bu algoritmaya göre oluşturulmuş örnek problemimiz ve kodumuz altta verilmiştir.

Program 5.27-1 yerçekimi arama

```
import java.util.Vector;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;

abstract class f_xj
{
abstract double func(double x[]);
}


public class yercekimsel_arama
{
int D;//boyut sayısı
int N;//kütle sayısı
int maxiter;//maksimum iterasyon
double fmin;//global minimum değer
double fsol[]; //kütle için minimum değer
double Masses[];
double Lowerbound[];
double Upperbound[];
double Lbvec[];
double Ubvec[];
double X[][];
double V[][];
double A[][];
double fitness[];
double Lb;
double Ub;
int index;
double min;
double[] dep;
double G;
f_xj ff;

    public yercekimsel_arama(f_xj iff,int iN,int imaxiter,double[] iLowerbound,double[] iUpperbound)
    {
                ff=iff;
                Lowerbound=iLowerbound;
                Upperbound=iUpperbound;
                D=Upperbound.length;
                N=iN;
                maxiter=imaxiter;
                Lbvec=new double[D];
                Ubvec=new double[D];
                X=new double[N][D];
                V=new double[N][D];
                fitness=new double[N];
                fsol=new double[D];
                Masses=new double[N];
                A=new double[N][D];
                }

                void initialize()
                {
                for(int i=0;i<N;i++)
                {for(int j=0;j<D;j++)
                 {X[i][j]=Lowerbound[j]+((Upperbound[j]-Lowerbound[j])*Math.random());}}
                }
```

```
        // sınırlardan çıkmışsa sınırlara geri çek
    double[][] simplebounds(double s[][])
        {
          for(int i=0;i<N;i++)
          {for(int j=0;j<D;j++)
          {if(s[i][j]<Lowerbound[j]){s[i][j]=Lowerbound[j];}
           if(s[i][j]>Upperbound[j]){s[i][j]=Upperbound[j];}}}
          return s;
        }

        //sınır dışına çıkmışsa  Lb ve Ub tesadüfi sayılarını oluştur
        double[][] randomsimplebounds(double s[][])
        {
          for(int i=0;i<N;i++)
          {for(int j=0;j<D;j++)
                    {if((s[i][j]<Lowerbound[j])||(s[i][j]>Upperbound[j]))
                     {s[i][j]=Lowerbound[j]+(Upperbound[j]-Lowerbound[j])*Math.random();}}}
           return s;
        }
        // fonksiyonları hesapla
        double[] Fitcalc(double a[][])
        {
                    for(int i=0;i<N;i++)
                    {fitness[i]=ff.func(a[i]);}
                    return fitness;
        }


        // minimum değerleri hesapla
        double[] getminval_index(double[] a)
        {
        double m=0.0;
        double b[]=new double[a.length];
        for(int i=0;i<a.length;i++)
        {b[i]=a[i];}
        double minval=a[0];
        for(int i=0;i<a.length;i++)
        {if(a[i]<minval){minval=a[i];}}
        for(int i=0;i<a.length;i++)
        {if(b[i]==minval){m=i;break;}};
        double[] dep=new double[2];
        dep[0]=minval;
        dep[1]=m;
        return dep;
        }

        // maksimum değerleri hesapla
        double[] getmaxval_index(double a[])
        {
        double m=0.0;
        double b[]=new double[a.length];
        for(int i=0;i<a.length;i++)
        {b[i]=a[i];}
        double maxval=a[0];
        for(int j=0;j<a.length;j++)
        {if(a[j]>maxval){maxval=a[j];}}
        for(int i=0;i<b.length;i++)
        {if(b[i]==maxval){m=i;break;}}
        double dep2[]=new double[2];
        dep2[0]=maxval;
        dep2[1]=m;
        return dep2;
        }

        // minimum çözümleri kütlelerini heapla
        double[] Calcmass(double fit[])
        {
        double[] dpmin=new double[2];
        double[] dpmax=new double[2];
        double FFmin=0.0;double FFmax=0.0;
        double best=0.0;double worst=0.0;
        double[] Masss =new double[fit.length];
        double[] Massss =new double[fit.length];
        dpmin=getminval_index(fit);
        FFmin=dpmin[0];
```

```
dpmax=getmaxval_index(fit);
FFmax=dpmax[0];
best=FFmin;
worst=FFmax;
for(int i=0;i<fit.length;i++)
{Masss[i]=(fit[i]-worst)/(best-worst);}
double t=0.0;
for(int i=0;i<fit.length;i++)
{t+=Masss[i];}
for(int i=0;i<fit.length;i++)
{Massss[i]=Masss[i]/t;}

return Massss;
}

int[] descendingsort_index(double[] Mass)
{
        Vector<Double> vv=new Vector<Double>();
        for(int i=0;i<Mass.length;i++)
        {vv.add(Mass[i]);}
        Object[] vold=vv.toArray();
        String str="";
        double[] voldarray1=new double[Mass.length];
        for(int i=0;i<Mass.length;i++)
        {voldarray1[i]=Double.parseDouble(vold[i].toString());}
        // voldarray1 holds unsorted values

        Collections.sort(vv,Collections.reverseOrder());//sorts in descending order
        Object[] vsort=vv.toArray();
        String strsorted="";
        double[] vsorted=new double[Mass.length];
        for(int i=0;i<Mass.length;i++)
        {vsorted[i]=Double.parseDouble(vsort[i].toString());}
        int[] indexed=new int[Mass.length];
  for(int i=0;i<Mass.length;i++)
        {for(int j=0;j<Mass.length;j++)
        {if(voldarray1[i]==vsorted[j])
          {indexed[j]=i;}}}

        return indexed;

}
// çekim değişkenlerini hesapla
double Gconst(int iterr)
{return 100.0*Math.exp(-20.0*(double)iterr/(double)maxiter);}

public double[][] Gravfield(double[] Mass,double[][] XX,double GG)
{
double epsilon=0.001;
double[][] dummy=new double[N][D];
int[] xx=new int[N];
double R=0.0;
int[] indexed=descendingsort_index(Mass);
int j=0;

for(int i=0;i<N;i++)
{for(int tt=0;tt<N;tt++)
  {j=indexed[tt];
          if(j!=i)
          {R=Matrix.norm(Matrix.substract(X[i],X[j]));//Equation 8
          for(int k=0;k<D;k++)
          {dummy[i][k]+=Math.random()*Mass[j]*((X[j][k]-X[i][k])/(R*R+epsilon));} //Equation 7
          }
  }
}
return  Matrix.multiply(GG,dummy);
}

double[][][] movements(double x[][],double a[][],double[][] v)
{
double[][] dummy=new double[N][D];
for(int i=0;i<N;i++)
{for(int j=0;j<D;j++)
 {dummy[i][j]=Math.random();}};
 v=Matrix.add(Matrix.dotproduct(dummy,v),a);//Equation (11)
 x=Matrix.add(x,v);//Equation (12)
```

```
                for(int i=0;i<N;i++)
                {for(int j=0;j<D;j++)
                 {X[i][j]=x[i][j];V[i][j]=v[i][j];}
                }
                double d3[][][]=new double[2][N][D];
                for(int i=0;i<N;i++)
                {for(int j=0;j<D;j++)
                  {d3[0][i][j]=X[i][j];d3[1][i][j]=V[i][j];}}
                return d3;
                }



    double[][] solution()
    {
            initialize();
            //System.out.println(Matrix.toString(X));
            double[][][] d4=new double[2][N][D];
            int iter=1;
               while(iter<maxiter)
               {
                        X=randomsimplebounds(X);
                        fitness=Fitcalc(X);
                   dep=getminval_index(fitness);
                        min=dep[0];
                        index=(int)dep[1];
                        if(iter==1)
                        {fmin=min;
                        for(int i=0;i<D;i++)
                        {fsol[i]=X[index][i];}
                        }
                        if(min<fmin)
                        {fmin=min;
                        for(int i=0;i<D;i++)
                        {fsol[i]=X[index][i];}
                        }

                        Masses=Calcmass(fitness);
                        G=Gconst(iter);
                        A=Gravfield(Masses,X,G);
                        d4=movements(X,A,V);
                        for(int i=0;i<N;i++)
                        {for(int j=0;j<D;j++)
                         {X[i][j]=d4[0][i][j];V[i][j]=d4[1][i][j];}}
                        iter++;
                        }
               double[][] d5=new double[2][D];
               d5[0][0]=fmin;
               for(int i=0;i<D;i++)
               {d5[1][i]=fsol[i];}
               return d5;
               }
               void toStringnew()
               {
                        double[][] dd=solution();
                        System.out.println("Optimizasyon değeri = "+dd[0][0]);
                        for(int i=0;i<D;i++)
                        {System.out.println("x["+i+"] ="+dd[1][i]);}
               }
}
```

Program 5.27-1 yerçekimi arama test programı

```
class fc extends f_xj //Axis parallel hyper-ellipsoid 2.2
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0;
int n=x.length;

for(int i=0;i<n;i++)
{ff+=i*x[i]*x[i];}
return ff;
}}

public class yercekimsel_arama_testi
```

```
{
   public static void main(String args[])
   {
     int N=20;
                int maxiter=1000;
                double[] Lowerbound={-5.0,-5.0,-5.0};
                double[] Upperbound={5.0,5.0,5.0};
                fc ff=new fc();
                yercekimsel_arama gsa=new yercekimsel_arama(ff,N,maxiter,Lowerbound,Upperbound);
                gsa.toStringnew();
   }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" yercekimsel_arama_testi
Optimizasyon değeri = 7.885392446131609E-10
x[0] =1.5355264124400567
x[1] =2.0299276077735885E-5
x[2] =-1.3720033442616322E-5

> Terminated with exit code 0.
```

5.28 Global tesadüfi metodlar : Büyük Patlama – Büyük Kırılma algoritması

Büyük Patlama – Büyük Kırılma algoritması (BP-BK) Erol ve Eksin (2006) tarafından önerilmiş evrenin oluşumu teorisini matematiksel yollarla taklit ederek bir optimizasyon yöntemine dönüştüren bir yöntemdir.Algoritma büyük patlama ve büyük kırılma olmak üzere iki kısıma ayrılmıştır. Büyük patlama, düzensizlik ve tesadüfi dağılım ilkelerini kullanarak evrendeki enerji salınımı prosesini içermektedir. Büyük kırılma fazında ise parçacıklar tesadüfi olarak dağılmakta ve ardından dağılan parçıcaklar bir düzen içerisine sokulmaktadır. Metasezgisel algoritmaların başarısı çeşitlendirme (diversification) ve yoğunlaştırma (intensification) fazlarının bir arada etkili bir biçimde uygulanabilmesiyle ölçülmektedir. BP-BK algoritmasında büyük patlama(enerji dağıtımı) fazının görevi çeşitlendirme aşamasını sağlamak iken, büyük kırılma fazı ise yoğunlaştırma aşaması ile görevlendirilmiştir. Büyük kırılma fazında yoğunlaşma prosesinin düşük hesaplama zamanı ve yüksek kesinlik ile sağlanması istenmektedir. Bu etkiyi sağlamak için "kütle merkezi" kavramı önerimiştir. Kütle merkezi " $\vec{x}^c$ " tanımı

$$\vec{x}^c = \frac{\sum_{i=1}^{N} \frac{\vec{x}_i}{f_i}}{\sum_{i=1}^{N} \frac{1}{f_i}} \qquad \text{(x1)}$$

ile ifade edilmektedir. Bu tanımda $\vec{x}_i$ D boyutlu arama uzayında bir nokta olarak tanımlanmıştır. f_i ise bu noktanın fonksiyon değerini (fitness) belirtmektedir. N populasyon büyüklüğünü göstermektedir. Kütle merkezi kavramı bu algoritmada mevcut iterasyon sayısına kadar elde edilen en iyi çözüm vektörü ile gösterilmektedir. Böylece olası çözümler "kütle merkezi " etrafında toplanarak fazında optimum çözüme yakınsama kolaylaşacaktır.

BP-BK algoritmasının adımları

Adım 1 Tanımlanan üst ve alt limitlere sadık kalarak N (populasyon büyüklüğü) adet D (tasarım değişkeni sayısı) boyutlu çözüm vektörleri tesadüfi sayılarla oluşturulur.

Adım 2 N adet olası çözüm vektörünün fonksiyon değeri hesaplanır.

Adım 3 "Kütle merkezi" (x1) denklemi ile elde edilir. Ayrıca fonksiyon değeri en iyi olan çözüm vektörü de kütle merkezi olarak alınabilir.

Adım 4 Yeni olası çözümleri oluşturmak için

$$\vec{x}_{yeni} = \beta * \vec{x}_c + \left((1.0-\beta) * best\right) + \left(randn * (\text{üstlimit - altlimit})/\text{iter}\right) \qquad \text{(x2)}$$

denklemi kullanılmıştır. Bu denklemde β 0.0 ile 1.0 arası değişen bir katsayıdır. Mevcut kitap kapsamında bu değer 0.4 olarak alınmıştır. "*best*" ile tanımlanan değer bir önceki en iyi çözüm vektörüdür. "randn" normal dağılımlı pseourandom sayı olarak tanımlanmıştır. "üstlimit" ve "altlimit" karşılıklı olarak arama uzayının üst ve alt limitlerini temsil etmektedir. "iter" ise mevcut iterasyon sayısını göstermektedir.

Adım 5 Yakınsama kriteri sağlanıncaya kadar ilk 4 adım uygulanır.

Program 5.28.1 Büyük Patlama – Büyük Kırılma algoritması

```
 import java.util.Random;
import java.io.*;
// Büyük patlama Büyük kırılma : Big Bang-Big Crunch //
// Osman K. Erol, Ibrahim Eksin                    //
// Programlayan: Oğuz Emrah Turgut                 //
abstract class f_xj
{
abstract double func(double x[]);
}

public class Bpatlama_Bkirilma
{
int N;
int maxiter;
int D;
double Lower[];
double Upper[];
double X[][];
double fitness[];
double Xcenter[];
double best[];
double worst[];
double Xnew[];
double BESTVAL[];
double iterdep[];
int iter;
double beta;
double minval;
double maxval;
int mindex;
int maxdex;
f_xj ff;

public Bpatlama_Bkirilma(f_xj iff,int iN,int imaxiter,double[] iLower,double[] iUpper,double ibeta)
{
ff=iff;
N=iN;
maxiter=imaxiter;
Lower=iLower;
Upper=iUpper;
beta=ibeta;
D=Lower.length;
Xnew=new double[D];
X=new double[N][D];
fitness=new double[N];
best=new double[D];
worst=new double[D];
Xcenter=new double[D];
}

void initialize()
{
for(int i=0;i<N;i++)
{for(int j=0;j<D;j++)
{X[i][j]=Lower[j]+((Upper[j]-Lower[j])*Math.random());}}
for(int i=0;i<N;i++)
{fitness[i]=ff.func(X[i]);}
 double dd[]=getminval_index(fitness);
minval=dd[0];
mindex=(int)dd[1];
for(int i=0;i<D;i++)
{best[i]=X[mindex][i];}
```

```
double dd1[]=getmaxval_index(fitness);
maxval=dd1[0];
maxdex=(int)dd1[1];
 for(int i=0;i<D;i++)
{worst[i]=X[maxdex][i];}
}


double[] simplebounds(double s[])
{
for(int i=0;i<D;i++)
{if(s[i]<Lower[i])
{s[i]=Lower[i];}
if(s[i]>Upper[i])
{s[i]=Upper[i];}
}
return s;
}

double[] getminval_index(double[] a)
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double minval=a[0];
for(int i=0;i<a.length;i++)
{if(a[i]<minval){minval=a[i];}}
 for(int i=0;i<a.length;i++)
{if(b[i]==minval){m=i;break;}};
double[] dep=new double[2];
dep[0]=minval;
 dep[1]=m;
return dep;
}

double[] getmaxval_index(double a[])
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
 double maxval=a[0];
for(int j=0;j<a.length;j++)
{if(a[j]>maxval){maxval=a[j];}}
for(int i=0;i<b.length;i++)
{if(b[i]==maxval){m=i;break;}}
double dep2[]=new double[2];
dep2[0]=maxval;
dep2[1]=m;
 return dep2;
}

double[][] cozum()
{
initialize();
iter=0;
Random rnd=new Random();
  while(iter<maxiter)
  {
  double dd3[]=getminval_index(fitness);
  mindex=(int)dd3[1];
  for(int i=0;i<D;i++)
 {Xcenter[i]=X[mindex][i];}
  for(int i=0;i<D;i++)
```

```
 {Xnew[i]=(beta*Xcenter[i])+((1.0-beta)*best[i])+(rnd.nextGaussian()*((Upper[i]/Lower[i])/(double)iter));}
 Xnew=simplebounds(Xnew);
 if(ff.func(Xnew)<ff.func(worst))
 {
  for(int i=0;i<D;i++)
 {worst[i]=Xnew[i];}
 }

for(int i=0;i<N;i++)
{fitness[i]=ff.func(X[i]);}

 double dd4[]=getminval_index(fitness);
 minval=dd4[0];
 mindex=(int)dd4[1];
 for(int i=0;i<D;i++)
{best[i]=X[mindex][i];}
 double dd5[]=getmaxval_index(fitness);
 maxval=dd5[0];
 maxdex=(int)dd4[1];
 for(int i=0;i<D;i++)
 {X[maxdex][i]=worst[i];}
iter++;
}

double[][] dp=new double[2][D];
dp[0][0]=minval;
for(int i=0;i<D;i++)
{dp[1][i]=best[i];}
return dp;
}

void toStringnew()
{
double[][] out=cozum();
System.out.println("optimum değer = "+out[0][0]);
for(int i=0;i<D;i++)
{System.out.println("x["+i+"] = "+out[1][i]);}
}}
```

Program 5.28.2 Büyük Patlama – Büyük Kırılma test programı

```
class f6 extends f_xj// Booth
{
double func(double x[])
{
double first=0.0;
first=(x[0]+2.0*x[1]-7.0)*(x[0]+2.0*x[1]-7.0)+(2.0*x[0]+x[1]-5.0)*(2.0*x[0]+x[1]-5.0);
return first;
}}
class f11 extends f_xj// Griewank
{
double func(double x[])
{
  double s=0.0;
  double fact=1.0;
  int m=x.length;
  for(int i=0;i<m;i++)
  {s+=x[i]*x[i];}
  for(int i=0;i<m;i++)
  {fact*=Math.cos(x[i]/Math.sqrt(i+1));}
  return (s/4000.0)+1.0+(-fact);
}}

class f14 extends f_xj // Michalewicz
{
  double func(double x[])
```

```
  {
            int n=x.length;
            double m=10.0;
            double s=0.0;
            for(int i=0;i<n;i++)
            {s+=Math.sin(x[i])*Math.pow(Math.sin(((double)i+1.0)*x[i]*x[i]/3.1415),2.0*m);}
            return -s;
  }
}

class f30 extends f_xj  // Rosenbrock's valley
{
double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;
double ff=0.0;
for(int i=0;i<n-1;i++)
{ff+=(100.0*(x[i+1]-x[i]*x[i])*(x[i+1]-x[i]*x[i])+(1.0-x[i])*(1.0-x[i]));}
return ff;
}}

class f8 extends f_xj// Colville
{
double func(double x[])
{
double first=0.0;
first=(100.0*(x[0]-x[1]*x[1])*(x[0]-x[1]*x[1]))+((1.0-x[0])*(1.0-x[0]))+(90.0*(x[3]-x[2]*x[2])*(x[3]-x[2]*x[2]))+((1.0-x[2])*(1.0-
x[2]))+(10.1*((x[1]-1.0)*(x[1]-1.0)+(x[3]-1.0)*(x[3]-1.0)))+(19.8*(x[1]-1.0)*(x[3]-1.0));
return first;
}}


public class big_bang_big_crunch_test
{
   public static void main(String args[])
  {
  int N=25;
  int maxiter=1000;
  double beta=0.4;
   //double[] Lower={-10.0,-10.0}; //f6 Booth
   //double[] Upper={10.0,10.0}; //f6 Booth
   //double[] Lower={-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0}; //f11 Griewank
   //double[] Upper={5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0}; //f11 Griewank
    //double[] Lower={0.0,0.0}; //f14 Michalewicz
   //double[] Upper={Math.PI,Math.PI}; //f14 Michalewicz
   //double[] Lower={-2.048,-2.048,-2.0480,-2.048,-2.048,-2.048,-2.048,-2.048,-2.048,-2.048};
   //double[] Upper={2.048,2.048,2.048,2.048,2.048,2.048,2.048,2.048,2.048,2.048}; // f30 Rosenbrock
   double[] Lower={-10.0,-10.0,-10.0,-10.0}; //f8 Colville
   double[] Upper={10.0,10.0,10.0,10.0}; //f8 Colville
   f8 ff=new f8();
   big_bang_big_crunch bb=new big_bang_big_crunch(ff,N,maxiter,Lower,Upper,beta);
   bb.toStringnew();
    }
}
```

2 boyutlu Booth fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" big_bang_big_crunch_test
Optimized value = 1.0549186789908263E-8
x[0] = 1.0000299512800657
x[1] = 2.9999337673678155
> Terminated with exit code 0.
```

2 boyutlu Michalewicz fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" big_bang_big_crunch_test
Optimized value = -1.8013224715515057
x[0] = 2.2028780660247738
x[1] = 1.5707685925149248
```

10 boyutlu Rosenbrock fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" big_bang_big_crunch_test
Optimized value = 0.18352157676422381
x[0] = 0.9982573299847123
x[1] = 0.9967904179946093
x[2] = 0.9925649070454818
x[3] = 0.9862071020110906
x[4] = 0.9724474390588439
x[5] = 0.9463439882250473
x[6] = 0.8950032277965182
x[7] = 0.8020120630603955
x[8] = 0.6420029893800693
x[9] = 0.40953169136122175
> Terminated with exit code 0.
```

10 boyutlu Griewank fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" big_bang_big_crunch_test
Optimized value = 1.496741319773065E-7
x[0] = 2.5530577576752666E-4
x[1] = 2.2672282159293393E-4
x[2] = 3.645725689398019E-4
x[3] = -1.7522124753388402E-4
x[4] = 3.522892766694187E-4
x[5] = 5.496165672674181E-4
x[6] = -6.416978873368457E-4
x[7] = 1.0112875761316458E-4
x[8] = 3.408203420478673E-4
x[9] = 2.7653425740075137E-4
> Terminated with exit code 0.
```

4 boyutlu Colville fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" big_bang_big_crunch_test
Optimized value = 1.4289323747292848E-5
x[0] = 0.9996338358937211
x[1] = 0.9998953932666456
x[2] = 1.0005933335577277
x[3] = 1.0011584910009146
> Terminated with exit code 0.
```

5.29 Global tesadüfi metodlar : Arı kovanı optimizasyonu

Arı kovanı algoritması (Artificial Bee Colony – ABC) , Yang (2005) tarafından önerilen sanal arı kolonisi algoritmasından esinlenerek Karaboga (2005) tarafından bal arısı kovanındaki arıların davranışlarını taklit ederek oluşturulan metasezgisel bir algoritmadır. ABC algoritmasında arılar üç gruba ayrılmıştır. Bunlar çalışan (employed) arılar, seyirci(onlooker) arılar ve gözcü (scout) arılardır. Seyirci arılar dans bölgesinde bekleyerek besin kaynağının tayininden sorumludurlar. Önceden uğradığı besin kaynağına tekrar ziyaret etmekle görevli arılar çalışan arılar olarak nitelenmektedir. Tesadüfi olarak besin arama görevi ise gözcü arılarındır. ABC algoritmasında arı kolonisi çalışan arılar ve seyirci arılar olmak üzere ikiye ayrılmıştır. Her besin kaynağı için sadece bir çalışan arı bulunmaktadır. Çalışan ve seyirci arılar tarafından besin kaynağı tüketilen herhangi bir çalışan arının kolonideki yeni görevi gözcülüktür. ABC algoritması üç ana prosesten meydana gelmektedir. İlk aşama çalışan arıların besin kaynaklarına gönderilmesi ve bu kaynaklardaki nektar miktarlarının belirlenmesi ile sonuçlandırılır. İkinci aşama ise çalışan arılardan elde edilen bilgiler ışığında seyirci arıların besin kaynaklarını seçmesi ile sonuçlanır. Üçüncü ve son aşama ise gözcü arıların olası besin kaynaklarına gönderilmesi ile sonuçlanmaktadır.

Algoritma, besin kaynaklarının sorumlu arı grubu tarafından tesadüfi olarak elde edilmesiyle ve bu kaynaklardaki besin miktarının değerlendirilmesiyle başlamaktadır. Ardından görevli arılar arı kovanına gelerek elde ettikleri bilgileri kovandaki diğer arılarla paylaşır. Bilgi paylaşımı sonrasında, her çalışan arı daha önce ziyaret ettiği besin kaynağına uğrar ve elde ettiği görsel deneyimler sonucunda ziyaret edilen kaynak yakınlarında yeni bir besin kaynağı seçer. Bu aşamadan sonra, gözcü arılar çalışan arılardan aldıkları bilgiler(besin miktarı) ışığında yeni besin kaynağı alanları seçer. Besin kaynağındaki nektar miktarının çokluğu, seyirci arı tarafından bu kaynağın seçim olasılığını yükseltecektir. Yani yüksek miktarda nektar taşıyan çalışan arılar, besin topladıkları kaynaklara seyirci arıları gönderecektir. Seyirci arı bu kaynağa ulaştıktan sonra edindiği görsel deneyimlerden faydalanarak besin kaynağının çevresinden başka bir kaynak seçecektir. Besin kaynağı sorumlu arılar tarafından tüketildiğinde, yeni besin kaynağı tesadüfi olarak gözcü arı tarafından seçilecek ve tüketilmiş besin kaynağı ile yer değişecektir. Yapay arı kolonisi algoritmasında besin kaynağı posizyonu optimizasyon probleminin çözüm vektörünü, besin kaynağındaki nektar çokluğu ise çözümü kalitesini göstermektedir. Çalışan ya da seyirci arıların sayısı populasyon büyüklüğünü *(PB)* göstermektedir. Algoritma tesadüfü olarak oluşturulan $i=1,2,3,..,PB$ adet $j=1,2,3,...,D$ boyutlu besin kaynağı pozisyonunun oluşturulmasıyla başlamaktadır. Bu tanımda D tasarım parametresi sayısnı belirtmektedir. $Ç=1,2,3,..,Ç_{max}$ adet çevrimden sonra optimum çözüme ulaşılması hedeflenmektedir. Çalışan ya da seyirci arı (x1) denklemiyle besin kaynağı pozisyonunu modifiye etmekte ve elde edilen nektar miktarını test etmektedir. Eğer kaynaktaki nektar miktarı bir öncekine göre düşükse, sorumlu arı önceki kaynağı terkedecektir. Yeni besin kaynağının eldesini ifade eden denklem

$$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j})$$

ile ifade edilir. (x1) denkleminde $k \in \{1,2,...,PB\}$ ve $j \in \{1,2,...,D\}$ tesadüfi olarak seçilen posiyon indeksleridir.

Burada k indeksinin i indeksine eşit olmaması gerekmektedir. $\Phi_{i,j}$ ise $x_{i,j}$ civarındaki besin kaynağı pozisyonlarına ulaşmak için oluşturulan ve [-1,1] arasında değişen tesadüfi bir sayıdır. Seyirci arının yeni besin kaynağını elde etmesi olasılıklı bir yaklaşım ile temellendirilerek sonuçlanmaktadır. Bu yaklaşım

$$p_i = \frac{çözüm_i}{\sum_{n=1}^{PB} çözüm_n}$$

ile gösterilmektedir. Burada $çözüm_i$ i sıradaki çözümün kalitesini göstermektedir. p_i olasılığıyla çalışan arılar elde ettikleri bilgileri seyirci arılarla paylaşır.

Sorumlu arılar tarafından iptal edilen besin kaynaklarının gözcü arılar tarafından yeni besin kaynaklarıyla yer değiştirildiği bilinmektedir. Bu fenomen YAK algoritmasında, tesadüfi bir besin kaynağı oluşturma ve bunu terk edilmiş kaynakla değiştirme prosesiyle canlandırılır. Eğer "limit" olarak tanımlanmış maksimum çevrim sayısı içerisinde yeni bir besin kaynağı pozisyonu geliştirilemiyorsa bu besin kaynağı terkedilecektir.Algoritmada her $v_{i,j}$ besin kaynağı pozisyonu $x_{i,j}$ pozisyonu ile karşılaştırılacaktır. Eğer yeni besin kalitesi eskisine göre eşit ya da daha iyi ise , bu besin(çözüm) eskisiyle yer değişecek ve arı hafızasındaki yerini alacaktır. Bu algoritmanın örnek kodu ve değişik test problemlerine uygulanması aşağıda verilmiştir.

Program 5.29.1 arı kovanı algoritması

/ Referans:

```
// Bahriye Akay , Dervis Karaboga, A modified Artificial Bee Colony algorithm for real-parameter optimization
// http://mf.erciyes.edu.tr/abc/
// Programlayan:  Oğuz Emrah Turgut
import java.io.*;
abstract class f_xj
{
abstract double func(double x[]);
}
public class ari_kovani
{
int D;//Tasarım parametreleri
int N;//Koloni büyüklüğü
int FS;//Besim kaynağı sayısı
int limit;//Sınırlı besin kaynağı sayısı
int Cyclemax;//Maksimum çevrim sayısı
double Lb;//Lower bound
double Ub;//Upper bound
double Foods[][];
double Fitness[];
double Objval[];
int trial[];
double prob[];
double Lbvec[];
double Ubvec[];
double sol[];
double Objvalsol;
double BESTVAL[];
int randomjj;
int randomkk;
double FitnessSol;
double Globalmin;
double GlobalParams[];
double fmin;
int indexfmin;
double phi;
f_xj ff;

public ari_kovani(f_xj iff,int iN,int ilimit,int iCyclemax,double[] iLbvec,double[] iUbvec)
{
ff=iff;
Lbvec=iLbvec;
Ubvec=iUbvec;
D=Ubvec.length;
N=iN;
FS=N/2;
limit=ilimit;
Cyclemax=iCyclemax;

BESTVAL=new double[Cyclemax];
sol=new double[D];
Foods=new double[FS][D];
Objval=new double[FS];
Fitness=new double[FS];
trial=new int[FS];
prob=new double[FS];
GlobalParams=new double[D];
}


 void initialize()
 {
for(int i=0;i<FS;i++)
{for(int j=0;j<D;j++)
{Foods[i][j]=Lbvec[j]+((Ubvec[j]-Lbvec[j])*Math.random());}
Objval[i]=ff.func(Foods[i]);}
```

```
Fitness=Fitcalc(Objval);
double[] dep=getminval_index(Objval);
fmin=dep[0];
indexfmin=(int)(dep[1]);
Globalmin=Objval[indexfmin];
for(int i=0;i<D;i++)
{GlobalParams[i]=Foods[indexfmin][i];}
 }
 double[] getminval_index(double[] a)
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double minval=a[0];
for(int i=0;i<a.length;i++)
{if(a[i]<minval){minval=a[i];}}
for(int i=0;i<a.length;i++)
{if(b[i]==minval){m=i;break;}};
double[] dep=new double[2];
dep[0]=minval;
dep[1]=m;
return dep;
}
int getmaxval_index(int a[])
{
int m=0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double maxval=a[0];
for(int j=0;j<a.length;j++)
{if(maxval>a[j]){maxval=a[j];}}
for(int i=0;i<b.length;i++)
{if(b[i]==maxval){m=i;break;}}
return m;
}
double[] Fitcalc(double a[])
{
 double[] out=new double[FS];
for(int i=0;i<FS;i++)
{ if(a[i]>=0.0)
{out[i]=1.0/(1.0+a[i]);}
else
{out[i]=1.0+Math.abs(a[i]);}
}
return out;
}

double Fitcalc(double aa)
{
double out=0.0;
if(aa>=0.0)
{out=1.0/(1.0+aa);}
else{out=1.0+Math.abs(aa);}
return out;
}

double[] simpleboundss(double s[])
{
for(int i=0;i<D;i++)
{if(s[i]<Lbvec[i])
{s[i]=Lbvec[i];}
if(s[i]>Ubvec[i])
{s[i]=Ubvec[i];}
```

```
}
return s;
}

double[][] cozum()
{
initialize();
int iter=0;

while(iter<Cyclemax)
{
//ÇALIŞAN ARI FAZI
for(int i=0;i<FS;i++)
{
randomjj=(int)Math.floor(D*Math.random());
randomkk=(int)Math.floor(FS*Math.random());
phi=2.0*(Math.random()-0.5);
while(randomkk==i)
{randomkk=(int)Math.floor(FS*Math.random());}
for(int j=0;j<D;j++)
{sol[j]=Foods[i][j];}
//v[i][j]=x[i][j]+phi[i][j]*(x[i][j]-x[k][j]);    equation (2)
sol[randomjj]=Foods[i][randomjj]+((Foods[i][randomjj]-Foods[randomkk][randomjj])*phi);
sol=simpleboundss(sol);
Objvalsol=ff.func(sol);
FitnessSol=Fitcalc(Objvalsol);
if(FitnessSol>Fitness[i])
{for(int j=0;j<D;j++){Foods[i][j]=sol[j];}
 Fitness[i]=FitnessSol;
Objval[i]=Objvalsol;
trial[i]=0;
}else{trial[i]++;}
}

for(int i=0;i<FS;i++)
{double sum=0.0;
for(int k=0;k<FS;k++)
{sum+=Fitness[k];}
 prob[i]=Fitness[i]/sum;
}

/////// SEYİRCİ ARI FAZI //////////
int ii=0;
int tt=0;
while(tt<FS)
{
if(Math.random()<prob[ii])
{
 tt++;
randomjj=(int)Math.floor(D*Math.random());
randomkk=(int)Math.floor(FS*Math.random());
phi=2.0*(Math.random()-0.5);
while(randomkk==ii)
{randomkk=(int)Math.floor(FS*Math.random());}
for(int j=0;j<D;j++)
{sol[j]=Foods[ii][j];}
sol[randomjj]=Foods[ii][randomjj]+((Foods[ii][randomjj]-Foods[randomkk][randomjj])*phi);
sol=simpleboundss(sol);
Objvalsol=ff.func(sol);
FitnessSol=Fitcalc(Objvalsol);
if(FitnessSol>Fitness[ii])
{for(int j=0;j<D;j++){Foods[ii][j]=sol[j];}
Fitness[ii]=FitnessSol;
Objval[ii]=Objvalsol;
trial[ii]=0;
```

```
}else{trial[ii]++;}
}
ii++;
if(ii==(FS))
{ii=0;}
}

double[] dep=getminval_index(Objval);
double fminn=dep[0];
int index=(int)dep[1];
if(Objval[index]<Globalmin)
{Globalmin=Objval[index];
for(int j=0;j<D;j++)
{GlobalParams[j]=Foods[index][j];}}


// GÖZCÜ ARI  FAZI
int ind=getmaxval_index(trial);
if(trial[ind]>limit)
{
trial[ind]=0;
for(int j=0;j<D;j++)
{sol[j]=Lbvec[j]+((Ubvec[j]-Lbvec[j])*Math.random());}
Objvalsol=ff.func(sol);
FitnessSol=Fitcalc(Objvalsol);
for(int i=0;i<D;i++)
{Foods[ind][i]=sol[i];}
Fitness[ind]=FitnessSol;
Objval[ind]=Objvalsol;
}
iter++;
}
double[][] out=new double[2][GlobalParams.length];
out[0][0]=Globalmin;
out[0][1]=GlobalParams.length;
for(int i=0;i<D;i++)
{out[1][i]=GlobalParams[i];}
return out;
}
void toStringnew()
{
double[][] out=cozum();
System.out.println("Optimized value = "+out[0][0]);
for(int i=0;i<out[0][1];i++)
{System.out.println("x["+i+"] = "+out[1][i]);}
}
}
```

Yapay arı kolonisi algoritmasının etkinliği Booth , Griewank, Michalewicz ve Rosenbrock fonskiyonları ile test edilmiştir.Algoritmanın çok boyutlu optimizasyon yeteneğini değerlendirmek için Griewank ve Rosenbrock fonksiyonları 10 boyutlu olarak modellenmiştir. Algoritmanın optimum noktaya yakınsama performansı aşağıda verilen grafiklerde gösterilmektedir.

Program 5.29.2 Yapay arı kolonisi test programı

```
class f6 extends f_xj// Booth
{
double func(double x[])
{
double first=0.0;
first=(x[0]+2.0*x[1]-7.0)*(x[0]+2.0*x[1]-7.0)+(2.0*x[0]+x[1]-5.0)*(2.0*x[0]+x[1]-5.0);
return first;
}}

class f11 extends f_xj// Griewank
{
```

```
double func(double x[])
{
  double s=0.0;
  double fact=1.0;
  int m=x.length;
  for(int i=0;i<m;i++)
  {s+=x[i]*x[i];}
  for(int i=0;i<m;i++)
  {fact*=Math.cos(x[i]/Math.sqrt(i+1));}
  return (s/4000.0)+1.0+(-fact);
}}

class f14 extends f_xj // Michalewicz
{
  double func(double x[])
  {
            int n=x.length;
            double m=10.0;
            double s=0.0;
            for(int i=0;i<n;i++)
            {s+=Math.sin(x[i])*Math.pow(Math.sin(((double)i+1.0)*x[i]*x[i]/3.1415),2.0*m);}
            return -s;
  }
}

class f30 extends f_xj  // Rosenbrock's valley
{
double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;
double ff=0.0;
for(int i=0;i<n-1;i++)
{ff+=(100.0*(x[i+1]-x[i]*x[i])*(x[i+1]-x[i]*x[i])+(1.0-x[i])*(1.0-x[i]));}
return ff;
}
}

public class artificial_bee_colony_test
{
   public static void main(String args[])
  {
  int N=20;
  int limit=200;
  int Cyclemax=2000;
   //double[] Lower={-10.0,-10.0}; //f6 Booth
   //double[] Upper={10.0,10.0}; //f6 Booth

   //double[] Lower={-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0}; //f11 Griewank
   //double[] Upper={5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0}; //f11 Griewank

   double[] Lower={0.0,0.0}; //f14 Michalewicz
   double[] Upper={Math.PI,Math.PI}; //f14 Michalewicz

   //double[] Lower={-2.048,-2.048,-2.0480,-2.048,-2.048,-2.048,-2.048,-2.048,-2.048,-2.048};
   //double[] Upper={2.048,2.048,2.048,2.048,2.048,2.048,2.048,2.048,2.048,2.048}; // f30 Rosenbrock

   f14 ff=new f14();
   artificial_bee_colony abc=new artificial_bee_colony(ff,N,limit,Cyclemax,Lower,Upper);
   abc.toStringnew();
    }
}
```

Program 5.XX3 2 boyutlu Booth fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" artificial_bee_colony_test
Optimized value = 1.81055324926134E-17
x[0] = 1.000000000021341
x[1] = 2.999999999919823

> Terminated with exit code 0.
```

10 boyutlu Griewank fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" artificial_bee_colony_test
Optimized value = 1.11022302462515E-16
x[0] = 1.68757087045928E-8
x[1] = 2.83357995318991E-9
x[2] = 6.85026067626402E-8
x[3] = -1.07803473163533E-8
x[4] = 9.47756169202529E-9
x[5] = 1.926253796984856E-8
x[6] = -1.594572467237635E-8
x[7] = 1.466069457830233E-8
x[8] = 2.272585864660378E-9
x[9] = 1.488581709518502E-8

> Terminated with exit code 0.
```

2 boyutlu Michalewicz fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" artificial_bee_colony_test
Optimized value = -1.8013224727098471
x[0] = 2.202874531055706
x[1] = 1.570773450575022

> Terminated with exit code 0.
```

10 boyutlu Rosenbroc k fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" artificial_bee_colony_test
Optimized value = 2.8748316991897463E-5
x[0] = 1.000030166097196
x[1] = 0.999987600933795
x[2] = 0.999949551390554
x[3] = 0.999999165738219
x[4] = 1.000098890123579
x[5] = 0.999902014931874
x[6] = 0.999755858060295
x[7] = 0.999177239339061
x[8] = 0.998390560265534
x[9] = 0.996604835810059

> Terminated with exit code 0.
```

5.30 Global tesadüfi metodlar : Kuantum davranışlı parçacık sürü optimizasyonu

Kuantum davranışlı parçacık sürü optimizasyonu(Quantum behaved particle swarm optimization Q-PSO) Sun *vd.* (2004) tarafından klasik parçacık sürü optimizasyonuna alternatif olarak sunulan bir algoritmadır. Parçacık sürü optimizasyonu modelinde N adet D boyutlu hacimsiz parçacığın konum ve hız vektörü

$\vec{x}_i^t = (\vec{x}_{i,1}^t, \vec{x}_{i,2}^t, .., \vec{x}_{i,D}^t)$

$\vec{v}_i^t = (\vec{v}_{i,1}^{t^t}, \vec{v}_{i,2}^{t^t}, .., \vec{v}_{i,D}^{t^t})$

ile tanımlanmıştır. Ardından Newton mekaniği prensiplerine sadık kalarak parçacık aşağıda tanımlanan denklemlerle hareket ederek optimuma ulaşmaktadır.

$$\vec{v}_i^{t+1} = \vec{v}_i^t + \phi_1(\vec{p}_i - \vec{x}_i^t) + (\vec{p}_g - \vec{x}_i^t)$$

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \vec{v}_i^{t+1}\Delta t$$

Burada ϕ_1 ve ϕ_2 algoritma parametreleri olarak tanımlanmaktadır. $\vec{p}_i = (\vec{p}_{i,1}, \vec{p}_{i,2}, .., \vec{p}_{i,D})$ bir önceki en iyi (fonskiyon değeri baz alınarak) konum vektörü olarak tanımlanmış ve bu vektöre p_{eniyi} adı verilmiştir. $\vec{p}_g = (\vec{p}_{g1}, \vec{p}_{g,2}, .., \vec{p}_{g,D})$ ise $\vec{p}_{g\_eniyi}$ olaraka dlandırılmış ve populasyondaki bütün parçacıklar arasında en iyi pozisyon vektörü olarak tanımlanmıştır. Δt zaman sabitidir ve bu algoritma için 1.0 olarak alınmıştır. Parçacık sürü algoritmasının çeşitli versiyonları değişik araştırmacılar tarafından önerilmiştir. Van den bergh ve Engelbrecht (2002), Garantili yakınsamalı parçacık sürü optimizasyon yöntemini sunmuştur. He vd. (2004) pasif toplamalı parçacık sürü optimizasyonunu önermiştir. Parçacıkların hareketini inceleyen yörünge analizlerinin sonucunda , PSO algoritmasına uygulanan model ne olursa olsun, sürü algoritmasındaki her parçacığın belli bir yerel toplayıcı (attractor) noktaya yakınsadığı görülmüştür (Clerc ve Kennedy, 2002). Yerel toplayıcı $pt_i^t = (pt_{i,1}^t, pt_{i,2}^t, .., pt_{i,j}^t, ...pt_{i,D}^t)$ vektörü ile gösterilmekte ve

$$pt_{i,j}^t = \frac{\phi_1 r_{i,j}^t p_{i,j}^t + \phi_2 R_{i,j}^t p_{g,j}^t}{\phi_1 r_{i,j}^t + \phi_2 R_{i,j}^t}$$

ile ifade edilmektedir. Denkemindeki r ve R katsayıları 0 ile 1 arasında değişen üniform dağılımlı tesadüfi sayılardır. Diğer katsayılar önceki denklemlerde tanımlandığı gibidir.

Sun vd. (2004) tarafından önerilen Q-PSO algoritmasında populasyon içerisindeki her bireyin hareketi $\vec{x}$ ve $\vec{v}$ vektörü yerine dalga fonksiyonu $\Psi(\vec{x}, t)$ ile ifade edilmektedir. Kuantum dünyasında parçacıkların hızı ve konumu aynı anda belirlenemez. Parçacığın konumu ancak olasılıklı yoğunluk fonksiyonu $|\Psi(\vec{x}, t)|^2$ ile elde edilmektedir. Lokal toplayıcı ve olasılıklı yoğunluk fonksiyonunu birleştirilmesiyle yeni konum vektörünün tayini Monte Karlo tesadüfi metodunun eklenmesi ile

$$x_{i,j}^t = pt_{i,j}^t \pm \alpha^t \left| \ln(\frac{1}{u_{i,j}^{t+1}}) \right|$$

denklemini oluşturur. Denklemindeki $C^t = (C_1^t, C_2^t, ..., C_D^t)$ ortalama en iyi konum vektörü olarak ifade edilmekte ve bütün parçacıkların p_{eniyi} pozisyonundaki ortalama değeri olarak tanımlanmaktadır.

$$C^t = (C_1^t, C_2^t, ..., C_D^t)\left(\frac{1}{N}\sum_{i=1}^{N} p_{i,1}^t, \frac{1}{N}\sum_{i=1}^{N} p_{i,2}^t, ..., \frac{1}{N}\sum_{i=1}^{N} p_{i,D}^t \right)$$

denkleminde N populasyon büyüklüğünü ve p ise p_{eniyi} pozisyon vektörünü göstermektedir. α ise iterasyon sayısına bağlı bir katsayıdır ve

$$\alpha^t = 0.5 + (T - t)\frac{0.5}{T}$$

ile tanımlanmaktadır (Sun vd.). T maksimum iterasyon sayısı , t ise mevcut iterasyon değerini göstermeketdir. $u_{i,j}$ ise 0 ile 1 arasında üniform dağılımlı tesadüfi bir sayıdır.

Program 5.30.1 Kuantum davranışlı parçacık sürüsü optimizasyonu programı

```
import java.io.*;
////////////////////////////////
```

```
// Programlayan Oğuz Emrah Turgut
//////////////////////////////
abstract class f_xj
{
abstract double func(double x[]);
}

public class kuantum_davranısli_suru
{
double c1;
double c2;
double wmax;
double wmin;
double alfa0;
double alfa1;
int maxiter;
double V[][];
double X[][];
double P[][];
double FP[];
double pp[][];
double G[];
double FG;
double L[][];
double C[];
double Cdepo[];
double Lower[];
double Upper[];
double fitness[];
int N;
int D;
double fi;
f_xj ff;
public kuantum_davranısli_suru(f_xj iff,double iLower[],double iUpper[],int imaxiter,int iN)
{
Upper=iUpper;
Lower=iLower;
D=Upper.length;
N=iN;
maxiter=imaxiter;
X=new double[N][D];
V=new double[N][D];
P=new double[N][D];
pp=new double[N][D];
G=new double[D];
L=new double[N][D];
C=new double[D];
Cdepo=new double[N];
fitness=new double[N];
FP=new double[N];
ff=iff;
c1=2.0;
c2=2.0;
wmax=0.9;
wmin=0.4;
alfa1=1.0;
alfa0=0.5;
}

double[] getminval_index(double[]              a)
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double minval=a[0];
```

```
for(int i=0;i<a.length;i++)
{if(a[i]<minval){minval=a[i];}}
for(int i=0;i<a.length;i++)
{if(b[i]==minval){m=i;break;}};
double[] dep=new double[2];
dep[0]=minval;
dep[1]=m;
return dep;
}
public void initialize()
{
for(int i=0;i<N;i++)
{
for(int j=0;j<D;j++)
{X[i][j]=Lower[j]+(Upper[j]-     Lower[j])*Math.random();P[i][j]=X[i][j];}
fitness[i]=ff.func(X[i]);
FP[i]=fitness[i];
}
double[] dep=getminval_index(FP)            ;
int bestindex=(int)dep[1];
for(int i=0;i<D;i++)
{G[i]=P[bestindex][i];}
FG=FP[bestindex];
}

double mean(double xx[])
{
int nn=xx.length;
double sum=0.0;
for (int i=0;i<nn;i++)
{sum+=xx[i];}
return sum/(double)nn;
}

double[][] simplebounds(double s[]            [])
{
for(int i=0;i<N;i++)
{for(int j=0;j<D;j++)
{if(s[i][j]<Lower[j])
{s[i][j]=Lower[j];}
if(s[i][j]>Upper[j])
{s[i][j]=Upper[j];}}}
return s;
}


double alfa(int tt)
{return alfa0+(((double)maxiter-(double)tt)*((alfa1-alfa0)/(double)maxiter);}

double w(int tt)
{return wmax-(((double)tt)*(wmax            -wmin)/((double)maxiter));}


double[][] solution()
{
initialize();
int iter=0;

while (iter<maxiter)
{
for(int j=0;j<D;j++)
{
for(int i=0;i<N;i++)
{Cdepo[i]=P[i][j];}
C[j]=mean(Cdepo);
}
```

```
for(int i=0;i<N;i++)
{
for (int j=0;j<D;j++)
{
fi=(Math.random()*c1)/((Math.random()*c1)+(Math.random()*c2));
pp[i][j]=fi*P[i][j]+(1-fi)*G[j]     ;
if(Math.random()>0.5)
{X[i][j]=pp[i][j]+alfa(iter)*Math.abs(C[j]-X[i][j])*Math.log10(1.0/Math.random());}
else
{X[i][j]=pp[i][j]-alfa(iter)*Math.abs(C[j]-X[i][j])*Math.log10(1.0/Math.random());}
}
}
X=simplebounds(X);

for(int i=0;i<N;i++)
{
fitness[i]=ff.func(X[i])                ;
if(fitness[i]<FP[i])
{
for(int j=0;j<D;j++)
{P[i][j]=X[i][j];}
FP[i]=fitness[i];
}
if(FP[i]<FG)
{
for(int j=0;j<D;j++)
{G[j]=P[i][j];}
                        FG=FP[i];
}
}
iter++;
}

double out[][]=new double[2][G.length];
out[0][0]=FG;out[0][1]=G.length;
for(int i=0;i<G.length;i++)
{out[1][i]=G[i];}
return out;
}

public String toString()
{double[][] out=solution();
 String s="Kuantum davranışlı sürü Optimum değer = "+out[0][0]+"\n";
 for(int i=0;i<out[0].length;i++)
{s+="x["+i+"] = "+out[1][i]+"\n";}
return s;
}

void println()
{
System.out.println(toString());
}
}
```

Program 5.30.2 Kuantum davranışlı parçacık sürüsü optimizasyonu test programı

```
class f36 extends f_xj //Ackley's function 2.9        f(x)=0;      @x=(0,0,0...)     -32.768<x[i]<32.768
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double a=20.0;
double b=0.2;
double c=2.*Math.PI;
```

```
int n=x.length;

double r=Math.PI/180;
double top1=0.0;
for(int i=0;i<n;i++)
{top1+=x[i]*x[i];}
top1=Math.sqrt(top1/n);
double top2=0.0;
for(int i=0;i<n;i++)
{top2+=Math.cos(r*c*x[i]);}
top2=top2/n;
double top=-a*Math.exp(-b*top1)-Math.exp(top2)+a+Math.exp(1);
return top;
}}

public class kuantum_davranısli_suru_test
{

public static void main(String arg[])
{
double[] Lower={-32.768,-32.768,-32.768,-32.768,-32.768,-32.768,-32.768,-32.768,-32.768,-32.768};
double[] Upper={32.768,32.768,32.768,32.768,32.768,32.768,32.768,32.768,32.768,32.768};
int maxiter=1000;
int N=20;
f36 ff=new f36();
kuantum_davranısli_suru qbpso=new kuantum_davranısli_suru(ff,Lower,Upper,maxiter,N);
qbpso.println();
}}
```

10D Ackley fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "c:\java\bin\javaw.exe" kuantum_davranısli_suru_test
Kuantum davranışlı sürü Optimum değer = 0.09294865139397235
x[0] = 0.001889573434399352
x[1] = -1.2509146855824854E-4
x[2] = -0.0027029745779945345
x[3] = 6.01584012324425E-4
x[4] = -0.002129069493801229
x[5] = -9.94771712690065E-4
x[6] = -7.299081231928087E-4
x[7] = -0.0019762257391262588
x[8] = -0.02953426328493973
x[9] = 0.06730775995121223

> Terminated with exit code 0.
```

Sun *vd.* (2011) lokal toplayıcı vektörünün QPSO algoritmasında önemli bir rol oynadığını görmüştür. Hatalı olarak oluşturulan bir lokal toplayıcı vektörünün algoritmayı yerel minimum noktalara yakalatma olasılığı yüksektir. Bunda dolayı Sun vd. (2011) Gauss dağılımlı bir lokal toplayıcı önermiştir. Gauss dağılımının standart sapması ve ortalama değeri dağılımın şeklini belirleyen önemli unsurlardır. Sun vd. (2011) taraından önerilen yeni lokal toplayıcı vektörü

$$yp_i^t = N(pt_i^t, C^t - p_i)$$

ile gösterilmektedir. pt_i^t lokal toplayıcı vektörü olarak bilinmektedir ve normal dağılımın ortalama değeri olarak tanımlanmıştır. Dağılımın standart sapması ise $C^t - p_i^t$ ile ifade edilmektedir. C^t değeri ortalama en iyi parçacık vektörünü, p_i^t ise p_{eniyi} vektörünü temsil etmektedir. Önerilen lokal toplayıcı ifadesi ile parçacığın yeni konumu

$$x_{i,j}^t = yp_{i,j}^t \pm \alpha^t \left| C_j^t - x_{i,j}^t \right| \ln\left(\frac{1}{u_{i,j}^t}\right)$$

denklemi ile gösterilebilir. Bu değişiklikler program içinde yapılacak küçük değişilkliklerle sağlanabilir.

Program 5.30.3 Gauss dağılımlı Kuantum davranışlı parçacık sürüsü optimizasyonu

```
import java.io.*;
//////////////////////////////
// Programlayan Oğuz Emrah Turgut
//////////////////////////////Gauss dağlımlı yerel parçacıklı
abstract class f_xj
{
abstract double func(double x[]);
}

public class kuantum_davranısli_suru1
{
double c1;
double c2;
double wmax;
double wmin;
double alfa0;
double alfa1;
int maxiter;
double V[][];
double X[][];
double P[][];
double FP[];
double pp[][];
double G[];
double FG;
double L[][];
double C[];
double Cdepo[];
double Lower[];
double Upper[];
double fitness[];
int N;
int D;
double fi;
f_xj ff;
public kuantum_davranısli_suru1(f_xj iff,double iLower[],double iUpper[],int imaxiter,int iN)
{
Upper=iUpper;
Lower=iLower;
D=Upper.length;
N=iN;
maxiter=imaxiter;
X=new double[N][D];
V=new double[N][D];
P=new double[N][D];
pp=new double[N][D];
G=new double[D];
L=new double[N][D];
C=new double[D];
Cdepo=new double[N];
fitness=new double[N];
FP=new double[N];
ff=iff;
c1=2.0;
c2=2.0;
wmax=0.9;
wmin=0.4;
alfa1=1.0;
alfa0=0.5;
}

double[] getminval_index(double[]              a)
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
```

```
{b[i]=a[i];}
double minval=a[0];
for(int i=0;i<a.length;i++)
{if(a[i]<minval){minval=a[i];}}
for(int i=0;i<a.length;i++)
{if(b[i]==minval){m=i;break;}};
double[] dep=new double[2];
dep[0]=minval;
dep[1]=m;
return dep;
}

public void initialize()
{
for(int i=0;i<N;i++)
{
for(int j=0;j<D;j++)
{X[i][j]=Lower[j]+(Upper[j]-     Lower[j])*Math.random();P[i][j]=X[i][j];}
fitness[i]=ff.func(X[i]);
FP[i]=fitness[i];
}
double[] dep=getminval_index(FP)            ;
int bestindex=(int)dep[1];
for(int i=0;i<D;i++)
{G[i]=P[bestindex][i];}
FG=FP[bestindex];
}

double mean(double xx[])
{
int nn=xx.length;
double sum=0.0;
for (int i=0;i<nn;i++)
{sum+=xx[i];}
return sum/(double)nn;
}

double[][] simplebounds(double s[]            [])
{
for(int i=0;i<N;i++)
{for(int j=0;j<D;j++)
{if(s[i][j]<Lower[j])
{s[i][j]=Lower[j];}
if(s[i][j]>Upper[j])
{s[i][j]=Upper[j];}}}
return s;
}


double alfa(int tt)
{return alfa0+((double)maxiter-(double)tt)*((alfa1-alfa0)/(double)maxiter);}

double w(int tt)
{return wmax-(((double)tt)*(wmax            -wmin)/((double)maxiter));}


double[][] solution()
{
initialize();
int iter=0;
double np[][]=new double[N][D];
while (iter<maxiter)
{
for(int j=0;j<D;j++)
{
for(int i=0;i<N;i++)
```

```
{Cdepo[i]=P[i][j];}
C[j]=mean(Cdepo);
}

for(int i=0;i<N;i++)
{
for (int j=0;j<D;j++)
{
fi=(Math.random()*c1)/((Math.random()*c1)+(Math.random()*c2));
pp[i][j]=fi*P[i][j]+(1-fi)*G[j]     ;
np[i][j]=gaussrandom(pp[i][j],(C[j]-P[i][j]));
                if(Math.random()>0.5)
                {X[i][j]=np[i][j]+alfa(iter)*Math.abs(C[j]-X[i][j])*Math.log10(1.0/Math.random());}
                else
                {X[i][j]=np[i][j]-alfa(iter)*Math.abs(C[j]-X[i][j])*Math.log10(1.0/Math.random());}
}
}
X=simplebounds(X);

for(int i=0;i<N;i++)
{
fitness[i]=ff.func(X[i])                ;
if(fitness[i]<FP[i])
{
for(int j=0;j<D;j++)
{P[i][j]=X[i][j];}
FP[i]=fitness[i];
}
if(FP[i]<FG)
{
for(int j=0;j<D;j++)
{G[j]=P[i][j];}
                        FG=FP[i];
}
}
iter++;
}

double out[][]=new double[2][G.length];
out[0][0]=FG;out[0][1]=G.length;
for(int i=0;i<G.length;i++)
{out[1][i]=G[i];}
return out;
}
public static double  gaussrandom(double xort,double r)
{ int iset=0;
 double gset=0;
 double fac,rsq,v1,v2;
 double aret=0;
 if(iset==0)
 {
  do{
                v1=2.0*Math.random()-1.0;
                v2=2.0*Math.random()-1.0;
                rsq=v1*v1+v2*v2;
    }while(rsq>=1.0 || rsq==0.0);
  fac=Math.sqrt(-2.0*Math.log(rsq)/rsq);
  gset=v1*fac;
  iset=1;
  aret=v2*fac;
 }
 else
 {
 iset=0;
 aret=gset;
 }
```

```
 return aret/8.0*r+xort;
 }

public String toString()
{double[][] out=solution();
 String s="Kuantum davranışlı sürü Gauss dağılımlı Optimum değer = "+out[0][0]+"\n";
 for(int i=0;i<out[0].length;i++)
{s+="x["+i+"] = "+out[1][i]+"\n";}
return s;
}

void println()
{
System.out.println(toString());
}
}
```

```
---------- Capture Output ----------
> "c:\java\bin\javaw.exe" kuantum_davranısli_suru_test
Kuantum davranışlı sürü Gauss dağılımlı Optimum değer = 0.006723133690120342
x[0] = 1.1355542657257396E-5
x[1] = 1.2898496408460182E-4
x[2] = -1.32234918987853E-4
x[3] = -2.1531729195798331E-4
x[4] = -2.452734884887547E-4
x[5] = 0.0014236862473236654
x[6] = 0.004194087163460161
x[7] = 1.789058402924936E-4
x[8] = 2.854769126213447E-4
x[9] = -0.002896189468225505


> Terminated with exit code 0.
```

Xi vd. (2008) fonksiyon değerlerinin (fitness value) algoritmadaki etkinliğini arrtırmak için ortalama en iyi parçacıkların(mean best solution) herbirine ağırlık katsayısı eklemiştir. Böylece populasyandaki her parçacık fonksiyon değerine göre sıralanmış ardından bu parçacıkların sıralanma derecelerine göre ağırlık katsayısı eklenmiştir.

$$m^t = (m_1^t, m_2^t, ..., m_n^t)\left(\frac{1}{N}\sum_{i=1}^{N} a_{i,1} p_{i,1}^t, \frac{1}{N}\sum_{i=1}^{N} a_{i,2} p_{i,2}^t, ..., \frac{1}{N}\sum_{i=1}^{N} a_{i,D} p_{i,1}^t\right)$$

a_i ağırlık katsayısı olarak tanımlanmakta ve 1.5 ‘tan başlayarak 0.5’e kadar doğrusal olarak azalmaktadır.

Program 5.30.4 Ağırlık katsayılı ortalama en iyi pozisyonlu kuantum davranışlı parçacık sürüsü optimizasyonu programı

```
//////////////////////////////////////////////////////////
// Programlayan = Oğuz Emrah Turgut  //
//////////////////////////////////////////////////////////
abstract class f_xj
{
abstract double func(double x[]);
}

public class agirlikli_ortalamali_kuantum_davranisli_suru
{
double c1;
double c2;
double wmax;
double wmin;
double alfa0;
double alfa1;
double betamax;
double betamin;
```

```
int maxiter;
double V[][];
double X[][];
double P[][];
double FP[];
double pp[][];
double G[];
double FG;
double L[][];
double C[];
double Cdepo[];
double Lower[];
double Upper[];
double fitness[];
double BEST[];
int N;
int D;
double fi;
f_xj ff;
 public agirlikli_ortalamali_kuantum_davranisli_suru(f_xj iff,double iLower[],double iUpper[],int imaxiter,int iN)
{

Upper=iUpper;
Lower=iLower;
D=Upper.length;
N=iN;
maxiter=imaxiter;
X=new double[N][D];
V=new double[N][D];
P=new double[N][D];
pp=new double[N][D];
G=new double[D];
L=new double[N][D];
C=new double[D];
Cdepo=new double[N];
fitness=new double[N];
FP=new double[N];
BEST=new double[maxiter];
ff=iff;
c1=2.0;
c2=2.0;
wmax=0.9;
wmin=0.4;
alfa1=1.0;
alfa0=0.5;
betamax=1.5;
betamin=0.5;

}

double[] getminval_index(double[] a)
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double minval=a[0];
for(int i=0;i<a.length;i++)
{if(a[i]<minval){minval=a[i];}}
for(int i=0;i<a.length;i++)
{if(b[i]==minval){m=i;break;}};
double[] dep=new double[2];
dep[0]=minval;
dep[1]=m;
return dep;
}
```

```
public void initialize()
{
     for(int i=0;i<N;i++)
    {
         for(int j=0;j<D;j++)
         {X[i][j]=Lower[j]+(Upper[j]-Lower[j])*Math.random();P[i][j]=X[i][j];}
         fitness[i]=ff.func(X[i]);
         FP[i]=fitness[i];
     }
     double[] dep=getminval_index(FP);
     int bestindex=(int)dep[1];
     for(int i=0;i<D;i++)
    {G[i]=P[bestindex][i];}
     FG=FP[bestindex];
 }

 double mean(double xx[])
 {
   int nn=xx.length;
   double sum=0.0;
   for (int i=0;i<nn;i++)
   {sum+=xx[i];}
    return sum/(double)nn;
 }

double[][] simplebounds(double s[][])
{
   for(int i=0;i<N;i++)
      {for(int j=0;j<D;j++)
              {if(s[i][j]<Lower[j])
                {s[i][j]=Lower[j];}
                if(s[i][j]>Upper[j])
              {s[i][j]=Upper[j];}}}
 return s;
}


double alfa(int tt)
{return alfa0+(((double)maxiter-(double)tt)*((alfa1-alfa0)/(double)maxiter);}

double w(int tt)
{return wmax-(((double)tt)*(wmax-wmin)/((double)maxiter));}

double beta(int tt)
{return (-1.0*(double)tt/(double)maxiter)+1.5;}

 double[][] solution()
 {
  initialize();
  int iter=0;

      while (iter<maxiter)
     {
              for(int j=0;j<D;j++)
          {
             for(int i=0;i<N;i++)
                {Cdepo[i]=beta(iter)*P[i][j];}
                C[j]=mean(Cdepo);
              }


                  for(int i=0;i<N;i++)
                 {
                        for (int j=0;j<D;j++)
```

```
                    {
                    fi=(Math.random()*c1)/((Math.random()*c1)+(Math.random()*c2));
                      pp[i][j]=fi*P[i][j]+(1-fi)*G[j];
                      if(Math.random()>0.5)
                      {X[i][j]=pp[i][j]+alfa(iter)*Math.abs(C[j]-X[i][j])*Math.log10(1.0/Math.random());}
                      else
                      {X[i][j]=pp[i][j]-alfa(iter)*Math.abs(C[j]-X[i][j])*Math.log10(1.0/Math.random());}
                      }
                  }

              X=simplebounds(X);

              for(int i=0;i<N;i++)
                {
                fitness[i]=ff.func(X[i]);
                    if(fitness[i]<FP[i])
                      {
                       for(int j=0;j<D;j++)
                      {P[i][j]=X[i][j];}
                       FP[i]=fitness[i];
                       }
                     if(FP[i]<FG)
                     {
                       for(int j=0;j<D;j++)
                       {G[j]=P[i][j];}
                        FG=FP[i];
                        }
                  }
                 iter++;
            }
        double out[][]=new double[2][G.length];
        out[0][0]=FG;out[0][1]=G.length;
        for(int i=0;i<G.length;i++)
        {out[1][i]=G[i];}
            return out;
    }
public String toString()
{double[][] out=solution();
 String s="Kuantum davranışlı sürü Optimum değer = "+out[0][0]+"\n";
 for(int i=0;i<out[0].length;i++)
{s+="x["+i+"] = "+out[1][i]+"\n";}
return s;
}

void println()
{
System.out.println(toString());
}
}
```

```
---------- Capture Output ----------
> "c:\java\bin\javaw.exe" kuantum_davranısli_suru_test
Ağırlık ortalamalı kuantum davranışlı sürü Optimum değer = 7.993605777301127E-15
x[0] = 1.02448359928845E-15
x[1] = -1.857088233918284E-15
x[2] = -9.128259758900347E-16
x[3] = -3.166063948967982E-15
x[4] = -4.926031723480324E-16
x[5] = -3.437485041725361 3E-15
x[6] = 3.388616398753063E-15
x[7] = -3.662022740752291E-15
x[8] = 2.122002135825235E-15
x[9] = -6.58937007118982E-16

> Terminated with exit code 0.
```

Coelho (2008) kuantum davranışlı parçacık sürüsü optimizasyonu algoritmasına mutasyon operatörü eklemiş ve stokastik etkinliği kaos teorisi ile arttırmaya çalışmıştır. Kaos, başlangıç koşullarına bağlı olarak sonsuz periyodik hareket oluşturan ve doğrusal olmayan bir sistem karekteristiği türüdür (Liu *vd.*, 2006). Zaslavski kaotik haritası (Zaslavskii, 1978) QPSO algoritmasına çeşitlendirme prosesinin etkinliğini arttırmak için aşağıdaki denklemlerde tanımlandığı gibi dahil edilmiştir.

$rand() > p_m^t \; ise$

$$x_i^{t+1} = p + \beta \left| C_i^t - x_i^t \right| \ln(1/u) \;\; k \geq 0.5 ise$$

$$x_i^{t+1} = p - \beta \left| C_i^t - x_i^t \right| \ln(1/u) \;\; k < 0.5 ise$$

$rand() \leq p_m^t \; ise$

$$x_i^{t+1} = p + \beta \left| C_i^t - x_i^t \right| \ln(1/\left|\phi_{i,j}\right|) \;\; k \geq 0.5 ise$$

$$x_i^{t+1} = p - \beta \left| C_i^t - x_i^t \right| \ln(1/\left|\phi_{i,j}\right|) \;\; k < 0.5 ise$$

Bu denklemde p_m^t, rand() ve k 0.0 ile 1.0 arasında uniform dağılımlı tesadüfi oluşturulan sayılardır. Bu algoritma için Yang *vd.* (2013) tarafından önerilen

$$p_m^t = 0.4\left(1 - \frac{t}{T}\right) \qquad \text{(x11)}$$

eşitliği kullanılmıştır. $\phi_{i,j}$ Zaslavskii kaotik haritası sonucunda oluşturan tesadüfi bir sayıdır ve aşağıda tanımlanan denklemler ile ifade edilmektedir.

$$\phi_{i,j}^{t+1} = \left\lfloor \phi_{i,j}^t + v(1 + \mu\varphi_{i,j}^t) + \varepsilon v\mu\cos(2\pi\phi_{i,j}^t) \right\rfloor \pmod 1$$

$$\phi_{i,j}^{t+1} = e^{-r}(\phi_{i,j}^t + \varepsilon\cos(2\pi\phi_{i,j}^t)$$

$$\mu = \frac{1 - e^{-r}}{r} \qquad \text{(x14)}$$

Önerilen algoritma için $\varepsilon = 10.0$, $v = 0.6$ ve $r = 0.2$ değerleri uygulanmıştır.

Program 5.30.5 Kuantum davranışlı sürü parçacık algoritması, Kaotik Zaslavskii haritası kullanıyor

```
 import java.io.*;
// Kuantum davranışlı sürü parçacık algoritması
// Kaotik Zaslavskii haritası kullanıyor
//////////////////////////////////////////////////////////
// Programlayan = Oğuz Emrah Turgut  //
//////////////////////////////////////////////////////////

abstract class f_xj
{
abstract double func(double x[]);
}

public class kuantum_davranısli_suru2
{
double c1;
double c2;
double wmax;
double wmin;
double alfa0;
double alfa1;
int maxiter;
double V[][];
double X[][];
double P[][];
double FP[];
double pp[][];
double PHI[][];
double yPHI[][];
double G[];
```

```
double FG;
double L[][];
double C[];
double Cdepo[];
double Lower[];
double Upper[];
double fitness[];
double BEST[];
int N;
int D;
double fi;
f_xj ff;
double eps;
double v;
double r;
double mu;
public kuantum_davranısli_suru2(f_xj iff,double iLower[],double iUpper[],int imaxiter,int iN)
{

Upper=iUpper;
Lower=iLower;
D=Upper.length;
N=iN;
maxiter=imaxiter;
X=new double[N][D];
V=new double[N][D];
P=new double[N][D];
pp=new double[N][D];
PHI=new double[N][D];
yPHI=new double[N][D];
G=new double[D];
L=new double[N][D];
C=new double[D];
Cdepo=new double[N];
fitness=new double[N];
FP=new double[N];
BEST=new double[maxiter];
ff=iff;
c1=2.0;
c2=2.0;
wmax=0.9;
wmin=0.4;
alfa1=1.0;
alfa0=0.5;
eps=10.0;
v=0.6;
r=2.0;
mu=(1.0-Math.exp(-r))/r;
}

double[] getminval_index(double[] a)
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double minval=a[0];
for(int i=0;i<a.length;i++)
{if(a[i]<minval){minval=a[i];}}
for(int i=0;i<a.length;i++)
{if(b[i]==minval){m=i;break;}};
double[] dep=new double[2];
dep[0]=minval;
dep[1]=m;
return dep;
}

public void initialize()
{
  for(int i=0;i<N;i++)
  {
      for(int j=0;j<D;j++)
     {X[i][j]=Lower[j]+(Upper[j]-Lower[j])*Math.random();P[i][j]=X[i][j];}
     fitness[i]=ff.func(X[i]);
     FP[i]=fitness[i];
   }
```

```
 double[] dep=getminval_index(FP);
 int bestindex=(int)dep[1];
 for(int i=0;i<D;i++)
{G[i]=P[bestindex][i];}
 FG=FP[bestindex];

 // Chaotic Zaslavskii haritası
 for(int ii=0;ii<N;ii++)
  {for(int jj=0;jj<D;jj++)
      {PHI[ii][jj]=Math.random();
  yPHI[ii][jj]=Math.random();}}
}

double mean(double xx[])
{
 int nn=xx.length;
double sum=0.0;
for (int i=0;i<nn;i++)
{sum+=xx[i];}
return sum/(double)nn;
}

double[][] simplebounds(double s[][])
{
   for(int i=0;i<N;i++)
     {for(int j=0;j<D;j++)
       {if(s[i][j]<Lower[j])
          {s[i][j]=Lower[j];}
        if(s[i][j]>Upper[j])
    {s[i][j]=Upper[j];}}}
 return s;
}


double alfa(int tt)
{return alfa0+((double)maxiter-(double)tt)*((alfa1-alfa0)/(double)maxiter);}

 double w(int tt)
 {return wmax-(((double)tt)*(wmax-wmin)/((double)maxiter));}

 double Pm(int tt)
 {return 0.4*(1.0-(((double)tt)/((double)maxiter)));}

  double[][] solution()
  {
   initialize();
   int iter=0;

   while (iter<maxiter)
   {
       for(int j=0;j<D;j++)
       {
            for(int i=0;i<N;i++)
           {Cdepo[i]=P[i][j];}
           C[j]=mean(Cdepo);
        }

        for(int i=0;i<N;i++)
        {
             for (int j=0;j<D;j++)
             {
              fi=(Math.random()*c1)/((Math.random()*c1)+(Math.random()*c2));
              pp[i][j]=fi*P[i][j]+(1-fi)*G[j];
              if(Math.random()>=Pm(iter))
              {
                  if(Math.random()>=0.5)
                 {X[i][j]=pp[i][j]+alfa(iter)*Math.abs(C[j]-X[i][j])*Math.log10(1.0/Math.random());}
                  else
                 {X[i][j]=pp[i][j]-alfa(iter)*Math.abs(C[j]-X[i][j])*Math.log10(1.0/Math.random());}
              }
              else
              {
                  if(Math.random()>=0.5)
                 {X[i][j]=pp[i][j]+alfa(iter)*Math.abs(C[j]-X[i][j])*Math.log10(1.0/PHI[i][j]);}
                  else
                 {X[i][j]=pp[i][j]-alfa(iter)*Math.abs(C[j]-X[i][j])*Math.log10(1.0/PHI[i][j]);}
```

```
                }
                }
               }

               for(int ii=0;ii<N;ii++)
                  {for(int jj=0;jj<D;jj++)
              {
 PHI[ii][jj]=(PHI[ii][jj]+(v*(1.0+mu*yPHI[ii][jj]))+(eps*v*mu*Math.cos(2.0*Math.PI*PHI[ii][jj])))-
 Math.floor((PHI[ii][jj]+(v*(1.0+mu*yPHI[ii][jj]))+(eps*v*mu*Math.cos(2.0*Math.PI*PHI[ii][jj]))));
          yPHI[ii][jj]=Math.exp(-r)*(yPHI[ii][jj]+eps*Math.exp(2.0*Math.PI*PHI[ii][jj]));
           }}
              X=simplebounds(X);

  for(int i=0;i<N;i++)
 {
   fitness[i]=ff.func(X[i]);
     if(fitness[i]<FP[i])
    {
      for(int j=0;j<D;j++)
     {P[i][j]=X[i][j];}
      FP[i]=fitness[i];
    }
    if(FP[i]<FG)
   {
     for(int j=0;j<D;j++)
    {G[j]=P[i][j];}
          FG=FP[i];
  }
}

BEST[iter]=FG;
iter++;
}
/*
double[] plott=new double[maxiter];
for(int i=0;i<maxiter;i++)
{plott[i]=i;}
Plot pp=new Plot(plott,BEST);
pp.plot();
*/
double out[][]=new double[2][G.length];
out[0][0]=FG;out[0][1]=G.length;
for(int i=0;i<G.length;i++)
{out[1][i]=G[i];}
return out;
}

public String toString()
{double[][] out=solution();
 String s="Kuantum davranışlı sürü Kaotik Zaslavskii haritası kullanımıyla Optimum değer = "+out[0][0]+"\n";
 for(int i=0;i<out[0].length;i++)
{s+="x["+i+"] = "+out[1][i]+"\n";}
return s;
}

void println()
{
System.out.println(toString());
}

}
```

```
---------- Capture Output ----------
> "c:\java\bin\javaw.exe" kuantum_davranisli_suru_test
Kuantum davranışlı sürü Kaotik Zaslavskii haritası kullanımıyla Optimum değer = 0.010904463046288448
x[0] = -8.58376541412341E-4
x[1] = -0.002343018637130466
x[2] = -1.0659359109675935E-6
x[3] = 0.002296393157572715
x[4] = -1.2300721169649759E-6
x[5] = -5.1831482732331895E-6
x[6] = -0.0029499576368196088
```

```
x[7] = -0.004604839934563724
x[8] = -0.005144089845667997
x[9] = -0.0025470560062806316


> Terminated with exit code 0.
```

Değişik kaotik haritalar bu algoritma içine dahil edilerek stokastik arama sürecinin etkinliği arttırılmaya çalışılabilir. Bu haritaların formülleri aşğıda listelenmiştir. Program kodu ayrıca verilmeyecektir.

Kaotik Chebyshev haritası

Chebyshev (Tavazoei ve Haeri, 2007) tarafından önerilen iteratif kaotik harita

$$x_{k+1} = \cos(k \cos^{-1}(x_k))$$

ile gösterilmektedir.

Kaotik Circle haritası

Hilborn (2004) tarafından önerilen kaotik harita

$$x_{k+1} = x_k + 0.2 - (0.5 - 2\pi)\sin(2\pi x_k) \bmod(1)$$

ile gösterilmiştir.

Kaotik İteratif harita

Önerilen kaotik dizi (May,1976)

$$x_{k+1} = \left| \sin\left(\frac{rand()\pi}{x_k} \right) \right|$$

ile ifade edilmiştir. Bu denkleminde mutlak değer ifadesi normalizasyon amacıyla kullanılmıştır, *rand()* 0.0 ile 1.0 arasında uniform dağılımlı tesadüfi bir sayıdır.

Kaotik Liebovitch haritası

Libovitch ve Toth (Tavazoei ve Haeri, 2007) tarafından önerilen kaotik harita

$$x_{k+1} = \begin{cases} \alpha x_k & 0 < x_k \le P_1 \\ \dfrac{P_2 - x_k}{P_2 - P_1} & P_1 < x_k \le P_2 \\ 1 - \beta(1 - x_k) & P_2 < x_k \le 1 \end{cases}$$

$$\alpha = \frac{P_2}{P_1}(1 - (P_2 - P_1))$$

$$\beta = \frac{1_2}{P_2 - 1}((P_2 - 1) - P_1(P_2 - P_1))$$

denklemleri ile uygulanmıştır.

Kaotik logistic harita

Kaotik davranışlar gösteren biyolojik bir populasyonun lineer olmayan davraışlarını içereen bu harita Li vd. (2011) tarafından

$$x_{k+1} = \alpha x_k (1 - x_k)$$ denklemi ile

ifade edilir. $x_0 \notin (1,1)$ olmak üzere ve $x_0 \notin \{10.0, 0.25, 0.5, 0.75, 1.0\}$ şartlarının sağlanmasıyla kaotik dizi oluşturulmaktadır. Deneysel veriler sonucunda $\alpha = 4.0$ önerilmiştir.

Kaotik Tent haritası

Önerilen algoritma (Ott, 2002) kaotik logistic haritasının bir benzeri olarak tasarlanmıştır.

$$x_{k+1} = \begin{cases} \dfrac{x_k}{0.7} & x_k < 0.7 \\ 3.33(1 - x_k) & x_k \ge 0.7 \end{cases}$$

Kaotik Piecewise harita

P, 0.0 ile 0.5 arasında değişen kontrol parametresi olmak üzere önerilen algoritma (Tomida, 2008)

$$x_{k+1}=\begin{cases}\dfrac{x_k}{P} & 0\le x_k<P\\ \dfrac{x_k-P}{0.5-P} & P\le x_k<0.5\\ \dfrac{1-P-x_k}{0.5-P} & 0.5\le x_k<1-P\\ \dfrac{1-x_k}{P} & 1-P\le x_k<1\end{cases}$$ denklemiyle gösterilmiştir.

5.31 Global tesadüfi metodlar : Diferansiyel arama algoritması

Diferansiyel arama (Civicioglu., 2012) (Differential search - DS) canlıların besin ihtiyacını karşılamak için uyguladıkları göç hareketini *Brown tipi tesadüfi adımlar* (Vito vd., 2011) yöntemiyle matematiksel olarak ifade eden bir optimizasyon algoritmasıdır. Diferansiyel arama algoritmasında göç halindeki yapay organizma aday çözüm vektörünü temsil etmektedir. Göç süresi sonunda süper organizma problemin minimum değerine ulaşmayı amaçlamaktadır. Besin arayışı sürecini hedefleyen bu göç esnasında süper organizma kendisi için konaklama yerleri (stopover site) belirler. Belirlenen konaklama bölgelerinde süperorganizma bir müddet vakit geçirir ardından süreç optimum noktaya ulaşıncaya kadar devam eder.

DS algoritmasında, yapay organizma $YapayOrganizma=[X_j],\ j=1,2,3,..,N$ ile ifade edilmektdir. Süper organizma ise $SuperOrganizma=[YapayOrganizma_j],\ j=1,2,3,..,N$ ile tanımlanmaktadır. Algoritma içinde konaklama bölgelerinin tayin edilmesi *Brown tipi tesadüfi adımlar* ile ifade edilmektedir. Yapay organizmadan alınmış tesadüfi bireyler $donör=SuperOrganizma_{tesadüfi\_karma(i)}$ metoduyla donörlere doğru yönelerek duraklama merkezlerine hareket ederler. Scale faktörü konaklama bölgelerinin beşirlenmesinde önemli bir rol oynar. Tesadüfi gama sayıları kullanılarak scale faktörü belirlenir. Konaklama (Stopover site) pozisyonu

$Stopoversite=SuperOrganizma+skalafaktörü*(Donör-SuperOrganizma)$

denklemi ile elde edilir. Bu işlem gerçekleştikten sonra , yeni konaklama pozisyonu ile süperorganizmanın pozisyonu karşılaştırılır. Eğer konaklama pozisyonu süperorganizmanın pozisyonundan daha verimliyse süperorganizma vektörü konaklama vektörü ile yerdeğiştirir. Bu süreç optimum noktaya erişilinceye kadar devam eder.

Program 5.31.1 Diferansiyel arama algoritması

```
////////////////////////////////////////////////////////////////////////////////
//Referans ------- Transforming geocentric cartesian coordinates//////////
//to geodetic coordinates by using differential search algorithm //////////
////////////////////////////////////////////////////////////////////////////////
/// Pınar Civicioğlu ////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Programlayan -- Oğuz Emrah Turgut ////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

import java.util.Random;

abstract class f_xj
{
abstract double func(double x[]);
}

public class diferansiyel_arama
{
int maxiter;
int N;
int D;
double Globalminimum;
```

```
double[] Globalminimizer;
double[] Lower;
double[] Upper;
double Scale;
double[][] r;
double[][] Superorganism;
double[][] Stopoversite;
double[] yStopoversite;
double[] ySuperorganism;
double[][] donor;
double[] y;
double p1;
double p2;
double fitnessbest;
f_xj ff;


public diferansiyel_arama(f_xj iff,int iN,double[] iLower,double[] iUpper,int imaxiter)
{
maxiter=imaxiter;
ff=iff;
N=iN;
Lower=iLower;
Upper=iUpper;
D=Lower.length;
Superorganism=new double[N][D];
Stopoversite=new double[N][D];
donor=new double[N][D];
r=new double[N][D];
y=new double[N];
Globalminimizer=new double[D];
yStopoversite=new double[N];
ySuperorganism=new double[N];
Globalminimum=1e10;
}

void initialize()
{
  for(int i=0;i<N;i++)
  {for(int j=0;j<D;j++)
    {Superorganism[i][j]=Lower[j]+((Upper[j]-Lower[j])*Math.random());}}
  for(int i=0;i<N;i++)
  {ySuperorganism[i]=ff.func(Superorganism[i]);}

}

double[] getminval_index(double[] a)
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double minval=a[0];
for(int i=0;i<a.length;i++)
{if(a[i]<minval){minval=a[i];}}
for(int i=0;i<a.length;i++)
{if(b[i]==minval){m=i;break;}};
double[] dep=new double[2];
dep[0]=minval;
dep[1]=m;
return dep;
}

// Random permutation of matrix
double[][] randperm(double a[][])
```

```
{
double y[][]=new double[a.length][a[0].length];
double[] sw=new double[a[0].length];
for(int i=0;i<a.length;i++)
{y[i]=a[i];}
int m=a.length;
int rp=0;
for(int i=0;i<m-1;i++)
{rp=(int)((m-i)*Math.random()+i);sw=y[i];
y[i]=y[rp];y[rp]=sw;}
return y;
}

int randi(int up)
{return (int)Math.floor(up*Math.random());}

int[] randi(int up,int count)
{
  int[] aa=new int[count];
  for(int i=0;i<count;i++)
  {aa[i]=(int)Math.floor(up*Math.random());}
 return aa;
}

double[][] solution()
{
initialize();
int dd[];
int mindex;
int iter=0;

  while(iter<maxiter)
  {
   donor=randperm(Superorganism);

            p1=0.3*Math.random();
            p2=0.3*Math.random();
               if(Math.random()<Math.random())
               {
                           if(Math.random()<p1)
                           { // Random-mutation I strategy
                                       for(int i=0;i<N;i++)
                                       {for(int j=0;j<D;j++)
                                                   {if(Math.random()<Math.random())
                                                               {r[i][j]=1.0;}}}
                           }
                                       else
                                       { // Diferential mutation strategy
                                         for(int i=0;i<N;i++)
                                         {r[i][randi(D)]=1.0;}
                                       }
                     }
               else
               {
                           // Random-mutation II strategy
                           for(int i=0;i<N;i++)
                           {dd=randi(D,(int)(p2*D));
                                       for(int j=0;j<dd.length;j++)
                                       {r[i][dd[j]]=1.0;}}
              }

                     Random rndm=new Random();
               Scale=4.0*(rndm.nextGaussian());// brownian walk
               // bio-interaction
 for(int i=0;i<N;i++)
```

```
{for(int j=0;j<D;j++)
{Stopoversite[i][j]=Superorganism[i][j]+((Scale*r[i][j])*(donor[i][j]-Superorganism[i][j]));}}

            // Boundary control;
               for(int i=0;i<N;i++)
               {for(int j=0;j<D;j++)
                        {if((Stopoversite[i][j]<Lower[j])||(Stopoversite[i][j]>Upper[j]))
                          {Stopoversite[i][j]=Lower[j]+(Math.random()*(Upper[j]-Lower[j]));}}}

                        // Selection II
                        for(int i=0;i<N;i++)
                        {yStopoversite[i]=ff.func(Stopoversite[i]);}

                        for(int i=0;i<N;i++)
                        {
                         if(yStopoversite[i]<ySuperorganism[i])
                         {ySuperorganism[i]=yStopoversite[i];
                           for(int j=0;j<D;j++)
                           {Superorganism[i][j]=Stopoversite[i][j];}}
                        }

                        double x1[]=getminval_index(ySuperorganism);
                        fitnessbest=x1[0];
                        mindex=(int)x1[1];

                        if (fitnessbest<Globalminimum)
               {
                            Globalminimum=fitnessbest;
                   for(int j=0;j<D;j++)
                   {Globalminimizer[j]=Superorganism[mindex][j];}
               }
                                    iter++;
  }

          double out[][]=new double[2][Globalminimizer.length];
              out[0][0]=Globalminimum;out[0][1]=Globalminimizer.length;
              for(int i=0;i<Globalminimizer.length;i++)
              {out[1][i]=Globalminimizer[i];}
              return out;

}

public String toString()
{double[][] out=solution();
String s="Optimum değer = "+out[0][0]+"\n";
for(int i=0;i<out[0][1];i++)
{s+="x["+i+"] = "+out[1][i]+"\n";}
return s;
}

public void print()
{System.out.println(toString());}

}
```

Program 5.31.2 Diferansiyel arama test programı

```
class f8 extends f_xj// Colville  f(x)=0.0  @x=(1,1,1,1)   -10.0<=x[i]<=10.0 i=0,1,2,3
{
double func(double x[])
{
double first=0.0;
first=(100.0*(x[0]-x[1]*x[1])*(x[0]-x[1]*x[1]))+((1.0-x[0])*(1.0-x[0]))+(90.0*(x[3]-x[2]*x[2])*(x[3]-x[2]*x[2]))+((1.0-x[2])*(1.0-
x[2]))+(10.1*((x[1]-1.0)*(x[1]-1.0)+(x[3]-1.0)*(x[3]-1.0)))+(19.8*(x[1]-1.0)*(x[3]-1.0));
return first;
```

```
}}

class f10 extends f_xj// Himmelblau f(x)=0.0  @x=(3.0,2.0),(-2.8051,3.1313),(-3.7793,-3.2831),(3.5844,-//1.8481)   -6.0<=x[i]<=6.0 i=0,1
{
double func(double x[])
{
double first=0.0;
first=(((x[0]*x[0]+x[1]-11.0)*(x[0]*x[0]+x[1]-11.0))+(x[0]+x[1]*x[1]-7.0)*(x[0]+x[1]*x[1]-7.0));
return first;
}}


class f53 extends f_xj //Hansen function     f(x)=-176.54  @x(-1.30,-1.42)
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double s1=0.0;double s2=0.0;
for(int i=0;i<=4;i++)
{s1+=((double)i+1.0)*Math.cos((double)i*x[0]+(double)i+1.0);}
for(int j=0;j<=4;j++)
{s2+=((double)j+1.0)*Math.cos(((double)j+2.0)*x[1]+(double)j+1.0);}
return s1*s2;
}}

class f104 extends f_xj  // Quintic  -10.0<x<10.0
{
        double func(double x[])
        {
        int n=x.length;
        double t=0.0;
        for(int i=0;i<n;i++)
        {t+=Math.abs(Math.pow(x[i],5.0)-3.0*Math.pow(x[i],4.0)+4.0*Math.pow(x[i],3.0)+2.0*x[i]*x[i]-10.0*x[i]-4.0);}
        return  t;
        }
}

class f57 extends f_xj //Zettl function     f(x1, x2) = f(-0.02990, 0.0) = -0.003791.  -30<=x[i]<=30
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double s1=Math.pow((x[0]*x[0]+x[1]*x[1]-2.0*x[0]),2.0)+0.25*x[0];
return s1;
}}
public class diferansiyel_arama_test
{
   public static void main(String args[])
   {
     double[] Lower={-10.0,-10.0,-10.0,-10.0};// f8 collvile
     double[] Upper={10.0,10.0,10.0,10.0};

     //double[] Lower={-6.0,-6.0};// f10 bohac
     //double[] Upper={6.0,6.0};

     //double[] Lower={-5.0,-5.0};// f53
     //double[] Upper={5.0,5.0};

     //double[] Lower={-10.0,-10.0};// f104
     //double[] Upper={10.0,10.0};

     //double[] Lower={-30.0,-30.0};// f57
     //double[] Upper={30.0,30.0};
         int maxiter=1000;
         int N=40;
```

```
        f8 ff=new f8();
        diferansiyel_arama ds=new diferansiyel_arama(ff,N,Lower,Upper,maxiter);
        ds.print();
  }
}
```

Zettl test fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" differential_search_test
Optimized value = -0.0037912372204688986
x[0] = -0.029895984994587257
x[1] = 1.8027918762197613E-10
> Terminated with exit code 0.
```

Quintic test fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" differential_search_test
Optimized value = 0.0
x[0] = -0.40262794118612383
x[1] = -0.40262794118612383
> Terminated with exit code 0.
```

Hansen test fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" differential_search_test
Optimized value = -176.54179313674572
x[0] = -1.3067077035192765
x[1] = 4.858056879332532
```

Himmelblau test fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" differential_search_test
Optimized value = 0.0
x[0] = 3.0
x[1] = 2.0
```

Colville test fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" differential_search_test
Optimized value = 0.0
x[0] = 1.0
x[1] = 1.0
x[2] = 1.0
x[3] = 1.0
```

5.32 Global tesadüfi metodlar : Hibrit Ahenk arama + Yapay Arı Kolonisi algoritması optimizasyonu

Ahenk arama ve Yapay arı kolonisi algoritmasının temelleri daha önceki bölümlerde ele alınmıştı. Bu bölümde Wu vd. (2012) tarafından önerilen Ahenk arama ve Yapay arı kolonisi algoritmasının hibritleştirilmesiyle oluşturulmuş Hibrit algoritma üzerinde durulmuştur. Ahenk arama algoritmasının dezavantajlarından biri lokal minimum noktalara çabuk takılması olarak gösterilebilir. Bunun sebebi ise Ahenk Hafızasının etkili olarak oluşturulamamasındandır. Ahenk Hafızası elit çözümlerin oluşturduğu bit matristir ve bu matrisin oluşturulma stratejisinin değiştirilmesiyle optimum çözüme daha çabuk ve doğru şekilde yakınsanabilir. Önerilen algoritmada Yapay arı kolonisi algoritması ahenk hafızasının optimizasyonu için kullanılmıştır. Yeni oluşturulan ahenk hafızası matrisi ise Pan vd. (2010) tarafından

önerilen adaptif ahenk arama yöntemiyle oluşturulmaktadır. Ayrıca Yapay arı kolonisi algoritmasının çeşitlendirme etkinliğinin arttırılması için parçacık sürü optimizasyonu (Kennedy et. al., 2001) yöntemi bu algoritma içinde uygulanmıştır. Bu uygulamayla algoritma içindeki sanal arılar konumlarını

$$v_{ij} = \omega x_{ij} + c_1\varphi_{ij}\left(x_j^{best} - x_{ij}\right) + c_2\varphi_{ij}\left(x_{kj} - x_{ij}\right) \quad \text{(x1)}$$

denklemiyle güncellerler. Tablo 5.11XX 'de önerilen algoritmanın adımları gösterilmiştir.

Tablo 5.32.1 Hibrit Ahenk arama + Yapay Arı kolonisi optimizasyonu algoritması

A – Başlangıç değerlerinin tanımlanması : Maxiter, AHB, "limit", bg , AHHO, PAO
B – AH'nin tanımlanması
For i=1 to AHB do {
For j=1 to D do
{ $x_{ij} = lb_j + rand1() * (ub_j - lb_j);$ }
f(x_i)=değerlendir(i);} cy=1;
C – AH'nin , Yapay arı kolonisi algoritması ile doğaçlanması
C-1 Görevli arıların besin kaynağı aramaya başlanması
For i=1 to AHB do {
$v_i = x_i;$
$v_{ij} = \omega x_{ij} + c_1\varphi_{ij}\left(x_j^{best} - x_{ij}\right) + c_2\varphi_{ij}\left(x_{kj} - x_{ij}\right)$
If ($f(v_i) > f(x_i)$)
deneme[i]++; $x_i = v_i;$
Else
deneme[i]++;}
C-2 For i=1 to AHB do // Her balarısı için olasılıklı seçilmenin hesaplanması
{p[i]=olasılık(i);}
C-3 For i=1 to AHB do // Seyirci arıların besin arama görevi
{index=Rulet seçilim();
//Seyirci arılar görevli arılara dönüşmektedir
C-1 'i tekrar et
}
C-4 For i=1 to AHB do
{ If (deneme[i]>limit)
$x_i = lb + rand2() * (ub - lb);$
}
D – Yeni bir ahenk doğaçlama işlemi oluşturma
For j =1 to D do
{
If(rand3()<AHHO)
{ $x_{yeni,j} = x_{indeks,j} + rand3() * bg;$
If(rand4()<PAO)
$x_{yeni,j} = x_{eniyi,j};$}
Else
$x_{yeni,j} = lb_j + rand5() * (ub_j - lb_j);$
}
}
E – If $f(x_{yeni}) > f(x_w)$ ise AH'yi $x_w = x_{yeni}$ şeklinde tanımlama
F – AH içindeki en iyi çözümü kayda alma; cy++;

G – If cy < Maxiter , C'ye dön, Else iterasyonu bitir;

AH = Ahenk Hafızası
PAO = Perde Ayarlama Oranı
AHB = Ahenk Hafızası Büyüklüğü
AHHO = Ahenk Hafızası Hatırlatma Oranı

Program 5.32.1 Hibrit Ahenk arama + Yapay Arı kolonisi optimizasyonu

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/// hibrit arama = Yapay arı kolonisi + Hahenk arama ///////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//hybrit harmony search and artificial bee colony algorithm for global optimization problems ////////
//Bin Wu., Cunhua Qian, Weihong Ni, Shuhai Fan ////////////////////////////////////////////////////////////
// Nanjing University of Technology,Department of Industrial Engineering ////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/// Programlayan = Oğuz Emrah Turgut ///////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////

import java.io.*;

abstract class f_xj
{abstract double func(double x[]);}

public class hibrit_arama
{
int D;
int N;// Colony size(employed + onlooker bee)
int FS;// Number of food source
int limit;//Limited food source number
int Cyclemax;//Maximum number of Cycles
double Lower[];//Lower bound
double Upper[];//Upper bound
double HMCR; //Harmony consideration rate
double PAR; //Pitch adjustment rate
double PARmax;
double PARmin;
double minval;
int mindex;
double maxval;
int maxdex;
double Xnew[];//new vector for harmony arama
double best[];
double worst[];
double fitness[];
int iter;
double prob[];// probability vector for roulette selection
int trial[];
double X[][];
double V[][];
int randj;
int randindex;
int L;
double c1;
double c2;
double w;
f_xj ff;

 public hibrit_arama(f_xj iff,int iN,int ilimit,int iCyclemax,double iLower[],double[] iUpper,double iHMCR)
{
ff=iff;
N=iN;
FS=N/2;
Lower=iLower;
```

```
Upper=iUpper;
D=Lower.length;
limit=ilimit;
Cyclemax=iCyclemax;
HMCR=iHMCR;
PARmax=2.0;
PARmin=0.0;
X=new double[FS][D];
V=new double[FS][D];
best=new double[D];
worst=new double[D];
Xnew=new double[D];
trial=new int[FS];
prob=new double[FS];
fitness=new double[FS];
//w=iw;
//c1=ic1;
//c2=ic2;
}

 void initialize()
{
for(int i=0;i<FS;i++)
{for(int j=0;j<D;j++)
 {X[i][j]=Lower[j]+(Math.random()*(Upper[j]-Lower[j]));}}

for(int i=0;i<FS;i++)
{fitness[i]=ff.func(X[i]);}
double[] d1=getminval_index(fitness);
minval=d1[0];
mindex=(int)d1[1];
for(int i=0;i<D;i++)
{best[i]=X[mindex][i];}

double d2[]=getmaxval_index(fitness);
maxval=d2[0];
maxdex=(int)d2[1];
for(int i=0;i<D;i++)
{worst[i]=X[maxdex][i];}
}

double[][] bounds(double x[][])
{
 int n=x.length;
for(int i=0;i<FS;i++)
 {for(int j=0;j<D;j++)
   {if((x[i][j]<Lower[j])||(x[i][j]>Upper[j]))
      {x[i][j]=Lower[j]+(Math.random()*(Upper[j]-Lower[j]));}}}
return x;
}

double[] roulette_selection(double x[])
{
 int n=x.length;double s1=0.0;
for(int i=0;i<n;i++)
{s1+=x[i];}
double[] out=new double[FS];
for(int i=0;i<FS;i++)
{out[i]=x[i]/s1;}
 return out;
}

double[][] solution()
{ initialize();
 iter=1;
while(iter<Cyclemax)
```

```
{
  for(int i=0;i<FS;i++)
  {randj=(int)Math.floor(D*Math.random());
    randindex=(int)Math.floor(FS*Math.random());
    while(randindex==i)
    {randindex=(int)Math.floor(FS*Math.random());}

     for(int j=0;j<D;j++)
     {V[i][j]=X[i][j];}

     V[i][randj]=(w*X[i][randj])+c2*(((Math.random()-0.5)*2.0)*(X[randindex][randj]-X[i][randj]))+(c1*((Math.random()-
0.5)*2.0)*(best[randj]-X[i][randj]));
     V=bounds(V);

      if(ff.func(V[i])<ff.func(X[i]))
     {
      trial[i]=0;
     for(int j=0;j<D;j++)
     {X[i][j]=V[i][j];}
      }
      else
     {trial[i]++;}
     }

for(int i=0;i<FS;i++)
{fitness[i]=ff.func(X[i]);}
prob=roulette_selection(fitness);

int tt=0;
int ii=0;
while(tt<FS)
{
 if(Math.random()<prob[ii])
 {
   tt++;
   randj=(int)Math.floor(D*Math.random());
   randindex=(int)Math.floor(FS*Math.random());
   while(randindex==ii)
  {randindex=(int)Math.floor(FS*Math.random());}
   for(int j=0;j<D;j++)
   {V[ii][j]=X[ii][j];}
    V[ii][randj]=(w*X[ii][randj])+c2*(((Math.random()-0.5)*2.0)*(X[randindex][randj]-X[ii][randj]))+(c1*((Math.random()-
0.5)*2.0)*(best[randj]-X[ii][randj]));
    if(ff.func(V[ii])>ff.func(X[ii]))
    {
     trial[ii]=0;
     for(int j=0;j<D;j++)
     {X[ii][j]=V[ii][j];}
    }
    else
   {trial[ii]++;}
  }
    ii++;
  if(ii==(FS))
   {ii=0;}
}

for(int i=0;i<FS;i++)
{
 if(trial[i]>limit)
 {
   for(int j=0;j<D;j++)
{X[i][j]=Lower[j]+(Math.random()*(Upper[j]-Lower[j]));}}}

for(int j=0;j<D;j++)
{
```

```
   if(Math.random()<HMCR)
   {
     L=(int)(((double)FS)*Math.random());
     PAR=PARmax-((PARmax-PARmin)*((double)iter)/((double)Cyclemax));
     Xnew[j]=X[L][j]+Math.random()*((Lower[j]-Upper[j])/100.0);
     if(Math.random()<PAR)
    {Xnew[j]=best[j];}
  }
  else
 {Xnew[j]=Lower[j]+(Math.random()*(Upper[j]-Lower[j]));}
}

 if(ff.func(Xnew)>ff.func(worst))
{for(int j=0;j<D;j++)
 {worst[j]=Xnew[j];}}

for(int i=0;i<FS;i++)
{fitness[i]=ff.func(X[i]);}

double[] dd4=getminval_index(fitness);
minval=dd4[0];
mindex=(int)dd4[1];
for(int i=0;i<D;i++)
{best[i]=X[mindex][i];}

double[] dd5=getmaxval_index(fitness);
maxval=dd5[0];
maxdex=(int)dd5[1];
for(int i=0;i<D;i++)
{X[maxdex][i]=worst[i];}
 iter++;
}

double[][] dp=new double[2][D];
dp[0][0]=minval;
for(int i=0;i<D;i++)
{dp[1][i]=best[i];}
 return dp;
}

void toStringnew()
{
double out[][]=solution();
System.out.println("Optimized value = "+out[0][0]);
for(int i=0;i<D;i++)
{System.out.println("x["+i+"] = "+out[1][i]);}
}

 double[] getmaxval_index(double a[])
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double maxval=a[0];
for(int j=0;j<a.length;j++)
{if(a[j]>maxval){maxval=a[j];}}
for(int i=0;i<b.length;i++)
{if(b[i]==maxval){m=i;break;}}
double dep2[]=new double[2];
dep2[0]=maxval;
dep2[1]=m;
return dep2;
}

double[] getminval_index(double[] a)
```

```
{
double m=0.0;
double b[]=new double[a.length];
for(int i=0;i<a.length;i++)
{b[i]=a[i];}
double minval=a[0];
for(int i=0;i<a.length;i++)
{if(a[i]<minval){minval=a[i];}}
for(int i=0;i<a.length;i++)
{if(b[i]==minval){m=i;break;}};
double[] dep=new double[2];
dep[0]=minval;
dep[1]=m;
 return dep;
}

}
```

Program 5.32.2 Hibrit Ahenk arama + Yapay Arı kolonisi optimizasyonu test programı

```
class f41 extends f_xj //Penalized  function   f(x)=0   @x(-1.0,-1.0)   -5.0<x[i]<5.0
{
public double u(double x,double a,double k,double m)
{
   double c=0.0;

     if(x>a){c=k*Math.pow(x-a,m);return c;}
                if((x>=-a)||(x<=a)){c=0;return c;}
                if(x<-a){c=k*Math.pow(-x-a,m);return c;}

   return c;
}

public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;
double[] y=new double[n];
for(int i=0;i<n;i++)
{y[i]=(0.25*(x[i]+1.0))+1.0;}
s1=Math.pow(Math.sin(3.1415*y[0]),2.0)*10.0;

for(int i=0;i<n-1;i++)
{s1+=Math.pow((y[i]-1.0),2.0)*(10.0*Math.pow(Math.sin(3.1415*y[i+1]),2.0)+1.0);}
s1+=Math.pow((y[n-1]-1.0),2.0);
s1=s1*3.1415/(double)n;
double s2=0.0;
for(int i=0;i<n;i++)
{s2+=u(x[i],10.0,100.0,4.0);}
return s1+s2;
}
}

class f102 extends f_xj  // Giunta   -1.0<x<1.0
{
  double func(double x[])
   {
    int n=x.length;
    double t=0.0;
    for(int i=0;i<n;i++)
{t+=(Math.sin(1.0-(x[i]*16.0/15.0))*Math.sin(1.0-(x[i]*16.0/15.0)))-(0.02*Math.sin(4.0-(x[i]*64.0/15.0)))- (Math.sin(1.0-
(x[i]*16.0/15.0)));
  }
    return  t+0.6;
    }
}
```

```
class f103 extends f_xj  // İnfinity  -1.0<x<1.0
{
		double func(double x[])
		{
		int n=x.length;
		double t=0.0;
		for(int i=0;i<n;i++)
		{t+=Math.pow(x[i],6.0)*(Math.sin(1.0/x[i])+2.0);}
		return  t;
		}
}

class f45 extends f_xj //Exp2 function   f(x)=0   @x(1.0,10.0)   0.0<x[i]<20.0
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;
for(int i=0;i<10;i++)
{s1+=Math.pow((Math.exp(-(double)i*x[0]/10.0)-(5.0*Math.exp(-(double)i*x[1]/10.0))-Math.exp(-(double)i/10.0)+5.0*Math.exp(-
(double)i)),2.0);}
return s1;
}
}

class f51 extends f_xj //Leon function     f(x)=0.0  @x(1.0,1.0)
{
public double func(double x[])
{
//çözümü istenen fonksiyon

double s1=100.0*(x[1]-x[0]*x[0]*x[0])*(x[1]-x[0]*x[0]*x[0])+(x[0]-1.0)*(x[0]-1.0);
return s1;
}
}


public class  hibrit_arama_test
{

     public static void main(String args[])
    {
		int N=21;
		int limit=2000;
		int Cyclemax=50000;


       double[] Lower={0.0,0.0};// exp2 f45
       double[] Upper={20.0,20.0};

       //double[] Lower={-1.2,-1.2};//  f95 leon
       //double[] Upper={1.2,1.2};

      //double[] Lower={-30.0,-30.0};//  f57 Zettl
     //double[] Upper={30.0,30.0};

     //double[] Lower={-1.0,-1.0};//  f102 Giunta
     //double[] Upper={1.0,1.0};

      //double[] Lower={-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0};//  f103 Infinity
     // double[] Upper={1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0};

      //double[] Lower={-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,};//  f40 Step
      //double[] Upper={5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0};
```

```
      //double[] Lower={-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,-5.0,};//  f41 Penalized
     //double[] Upper={5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0};

      double HMCR=0.99;
                f45 ff=new f45();
                hibrit_arama hs=new hibrit_arama(ff,N,limit,Cyclemax,Lower,Upper,HMCR);
                hs.toStringnew();

              }

}
```

10 boyutlu Penalized1 fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" hybrid_search_test
Optimized value = 2.7052601579082393E-10
x[0] = -0.9998832097116027
x[1] = -0.9999999999895121
x[2] = -1.00000000000082
x[3] = -1.0000000000009017
x[4] = -0.9999999999997512
x[5] = -1.0000000000003755
x[6] = -1.0000000000004476
x[7] = -0.999999999999499
x[8] = -0.9999999999993767
x[9] = -1.0000000000008908
> Terminated with exit code 0.
```

2 boyutlu Giunta fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" hybrid_search_test
Optimized value = 1.051589167740698E-19
x[0] = 3.0000000000581326
x[1] = 1.9999999999658042
> Terminated with exit code 0.
```

10 boyutlu Infinity fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" hybrid_search_test
Optimized value = 2.000025369942223E-297
x[0] = -8.204999704765738E-55
x[1] = -1.06220666119658E-54
x[2] = -4.396908585138148E-55
x[3] = -4.966788650452286E-55
x[4] = 3.0032515706337345E-56
x[5] = -4.9823747944436764E-55
x[6] = -3.5493502571066204E-50
x[7] = 1.0034313647674954E-54
x[8] = -2.3479466144951277E-55
x[9] = -7.05654735871882E-53
x[10] = -2.0096086873071583E-55
> Terminated with exit code 0.
```

2 boyutlu Exp2 fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" hybrid_search_test
Optimized value = 3.918816046997309E-33
x[0] = 1.0
x[1] = 10.0
> Terminated with exit code 0.
```

2 boyutlu Leon fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" hybrid_search_test
Optimized value = 2.2157556996451415E-8
x[0] = 0.9998514575035851
x[1] = 0.9997019743480161
> Terminated with exit code 0.
```

2 boyutlu Zettl fonksiyonu için optimum değer

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" hybrid_search_test
Optimized value = -0.003791237220468898
x[0] = -0.029895984919806294
x[1] = 5.629613744326016E-10
> Terminated with exit code 0.
```

5.33 Sınırlamalı lineer olmayan optimizasyon

Bu noktada biraz daha genel bir problem olan sınırlamalı optimizasyon problemini irdeleyelim. Sınırlamalı optimizasyon problemi genel olarak

Minimum f(x)

Sınır şartları : p(x)=0

q(x) $\geq$ 0 olarak tanımlanabilir.

Genelde bu tür sınır şartlarını denkleme ekliyerek yeni bir denklem oluştururuz. Birinci denklem p(x)=0 için

$P(x) = P(x_0, x_1, ..) = \sum_{k=0}^{r} p_k^2(x)$ olarak tanımlansın

İkinci denklem q(x) $\geq$ 0 için

$$Q(x) = Q(x_0, x_1, ..) = \sum_{k=0}^{r} \min^2\{0, q_k(x)\}$$

burada min fonksiyonu 0 ile $q_k(x)$ in minimum olanını seçer.

Minimumunu bulmamız gereken fonksiyon

$F(x) = f(x) + \mu[P(x) + Q(x)]$ şeklini alır. Denklemdeki μ ceza katsayısı ismini alır.

İkinci denklem q(x) $\geq$ 0 için Q(x) denklemimizi

$Q(x) = Q(x_0, x_1, ..) = \sum_{k=0}^{r} \min^2\{0, [q_k(x) - \varepsilon]\}$ şeklinde değiştirerek de kullanabiliriz. Burada ε >0 çok küçük bir sayıdır. Bu küçük değişiklikle optimal değerin tam sınıra gelmesi durumunda orijinal fonksiyonun çözümü reddetmesini engelleyici bir etki yaratmış oluruz. Örnek problemde Fletcher-Reeves metodu kullanılmıştır, doğal olarak sınırlandırılmış optimizasyon hesaplarında gördüğümüz tüm optimizasyon metodlarından yararlanabiliriz.

Program 5.33.1 Fletcher-Reeves optimizasyon metodu kullanılarak Sınırlandırılmış optimizasyon

```
İmport java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
//SINIRLAMALI OPTİMİZASYON
class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
double mu=100.0;
ff=(x[0]-1.0)*(x[0]-1.0)+(x[1]-2.0)*(x[1]-2.0)+(x[2]-3.0)*(x[2]-3.0)+mu*(P(x)+Q(x));
return ff;
}
public double P(double x[])
{// p(x)=0 sınırlama durumu
int n=x.length;
 return (x[0]+x[1]-2.0)*(x[0]+x[1]-2.0);
}
public double Q(double x[])
{// p(x)=0 sınırlama durumu
```

```
double eps=1.0e-6;
int n=x.length;
double ax=x[1]-x[2]-3.0-eps;
double bx=0;
if(ax<0) bx=ax*ax;
 return bx;
}
}

public class OPO12H
{
public static double VT_V(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*left[i];
  }
return tot;
}

public static double[][] V_VT(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
for(int i=0;i<n;i++)
  {
  for(int j=0;j<n;j++)
    {
                aa[i][j]=left[i]*left[j];
                }
  }
return aa;
}

public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}

public static double turev(f_xj f,double x[],int denklem_ref,int x_ref)
{
//
// verilen fonksiyonun ikinci türevlerinin matrisi
// fonksiyon f in denklem_ref sayılı fonksiyonunun(sayılar 0 dan başlar)
// x[x_ref] değişkenine göre türevi (sayılar 0 dan başlar)
// df_denklem_ref(x)/d_x_ref
// bu metod newtond metodu içinde kullanılmak içindir.
double h0=0.256808;
int i,m;
int n=7;
double f1[];
f1=new double[x.length];
double f2[];
f2=new double[x.length];
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
```

```
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
x2[i]=x[i];
}
//turev of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i]*1.0094847;
f1=df(f,x1);
f2=df(f,x2);
T[i][0]=( f1[denklem_ref] - f2[denklem_ref])/(2.0*h[i]);
x1[x_ref]=x[x_ref];
x2[x_ref]=x[x_ref];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}
public static double[] df(f_xj f,double xxi[])
{
//Gredyen vektörü
// f(x0,x1,x2,...xn) founksiyonunun türev vektörü
// [df/dx0, df/dx1,df/dx2,....,df/dxn]
// df/dxj  j=0...x.length
//
// bu fonksiyon bilgisayar tarafından hesaplanmaktadır
// kullanıcı tarafından tanımlanan f(x) fonksiyonunun
// sayısal olarak alınan türevidir.

//
double a[]=new double[xxi.length];
for(int i=0;i<xxi.length;i++)
   {
   a[i]=turev(f,xxi,i);
   }
return a;
}
public static double turev(f_xj f,double xxi[],int x_ref)
{
```

```
// df/dxj
double h0=0.256808;
int i,m;
int n=7;
double x1[];
x1=new double[xxi.length];
double x2[];
x2=new double[xxi.length];
for(i=0;i<xxi.length;i++)
{
x1[i]=xxi[i];
x2[i]=xxi[i];
}
//turev of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
//h[i]=h0*Math.pow(r,i);
h0/=2.0;
h[i]=h0;
}
//first turev (difference formula)
for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i];
T[i][0]=(f.func(x1)-f.func(x2))/(2.0*h[i]);
x1[x_ref]=xxi[x_ref];
x2[x_ref]=xxi[x_ref];
}

for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] -
  h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i] - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}
public static double[] turev(f_xj f_deriv,double x[])
{
// df/dxj  j=0...x.length
// This method calculates turev of a function with more than one independent variable.
// Accuracy of method can be adjusted by changing variables h0 and n
// function input should be in the form given in abstract class
// f_xj,j=0...x.length = df/dx(x_ref)
double a[]=new double[x.length];
for(int x_ref=0;x_ref<x.length;x_ref++)
   {
   a[x_ref]=turev(f_deriv,x,x_ref);
   }
return a;
```

```
}

public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}


public static double linmin(f_xj f,double p[],double xi[])
{
        double tol=2.0e-4;
   int n=p.length;
   double pcom[],xicom[];
        int j;
        double xx,xmin,bx,ax;
        f1dim f1=new f1dim(f,p,xi);
        ax=0.0;
        xx=1.0;
        bx=2.0;
        double aa[]=mnbrak(f1,ax,xx,bx);
        ax=aa[0];
        xx=aa[1];
        bx=aa[2];
        xmin=brent(f1,ax,xx,bx,tol);
        return xmin;
}

public static double[][] linminiter(f_xj f,double p[],double xi[])
{   // yeni iterasyon noktasını verir
   int n=p.length;
   double xmin=linmin(f,p,xi);
   double aa[][]=new double[2][n];
        for (int j=0;j<n;j++)
        {   xi[j] *= xmin;
                p[j] += xi[j];
                aa[0][j]=p[j];
                aa[1][j]=xi[j];
        }
   return aa;
}
public  static double[] brentf(f_x f,double ax,double bx,double cx,double tol)
{
// ax,bx,cx üç adet tahmin noktası (bxax ile cx arasında yer almalıdır)
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double CGOLD=(3.0-Math.sqrt(5))/2.0; //altın oran
double ZEPS=1.0e-10;
double xmin;
double aa[]=new double[2];
// SHFT(double a,double b,double c,d) (a)=(b);(b)=(c);(c)=(d);
        int iter;
        double a,b,d,etemp,fu,fv,fw,fx,p,q,r,tol1,tol2,u,v,w,x,xm;
        double e=0.0;
        d=0.0;
        a=((ax < cx) ? ax : cx);
        b=((ax > cx) ? ax : cx);
        x=w=v=bx;
        fw=fv=fx=f.func(x);
        for (iter=1;iter<=ITMAX;iter++) {
                xm=0.5*(a+b);
                tol2=2.0*(tol1=tol*Math.abs(x)+ZEPS);
                if (Math.abs(x-xm) <= (tol2-0.5*(b-a))) {
                        xmin=x;
            aa[0]=xmin;
            aa[1]=fx;
            return aa;
                }
```

```
                        if (Math.abs(e) > tol1) {
                                r=(x-w)*(fx-fv);
                                q=(x-v)*(fx-fw);
                                p=(x-v)*q-(x-w)*r;
                                q=2.0*(q-r);
                                if (q > 0.0) p = -p;
                                q=Math.abs(q);
                                etemp=e;
                                e=d;
                                if (Math.abs(p) >= Math.abs(0.5*q*etemp) || p <= q*(a-x) || p >= q*(b-x))
                                        d=CGOLD*(e=(x >= xm ? a-x : b-x));
                                else {
                                        d=p/q;
                                        u=x+d;
                                        if (u-a < tol2 || b-u < tol2)
                                                d=SIGN(tol1,xm-x);
                                }
                        } else {
                                d=CGOLD*(e=(x >= xm ? a-x : b-x));
                        }
                        u=(Math.abs(d) >= tol1 ? x+d : x+SIGN(tol1,d));
                        fu=f.func(u);
                        if (fu <= fx) {
                                if (u >= x) a=x; else b=x;
                                {v=w;w=x;x=u;}
                                {fv=fw;fw=fx;fx=fu;}
                        } else {
                                if (u < x) a=u; else b=u;
                                if (fu <= fw || w == x) {
                                        v=w;
                                        w=u;
                                        fv=fw;
                                        fw=fu;
                                } else if (fu <= fv || v == x || v == w) {
                                        v=u;
                                        fv=fu;
                                }
                        }
                }
                System.out.println("BRENT metodunda maksimum iterasyon sayısı aşıldı");
                xmin=x;
                //minumum değer a[0] ile fonksiyon değeri a[1] ile geridöndürülmektedir
                aa[0]=xmin;
                aa[1]=fx;
                return aa;
}
public  static double[] mnbrak(f_x f,double ax,double dx)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçilen uzaklık
return mnbrak(f,ax,(ax-dx),(ax+dx));
}

public  static double[] mnbrak(f_x f,double ax)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçiln uzaklık
return mnbrak(f,ax,(ax-0.1),(ax+0.1));
}

public  static double[] mnbrak(f_x f,double ax,double bx,double cx)
{
// ax,bx,cx üç adet tahmin noktası verildiğinde içinde minimum bulunan ax,bx,cx üç adetnoktayı bize arama
// sonucu bulur
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
```

```
int ITMAX=100;
double xmin;
double GOLD=(Math.sqrt(5.0)+1.0)/2.0;;
double GLIMIT=100.0;
double TINY=1.0e-20;
double fa,fb,fc;
fa=0;
fb=0;
fc=0;
        double ulim,u,r,q,fu,dum;
        double aa[]=new double[3];
        fa=f.func(ax);
        fb=f.func(bx);
        if (fb > fa) {
                //SHFT(dum,*ax,*bx,dum)
                {dum=ax;ax=bx;bx=dum;}
                //SHFT(dum,*fb,*fa,dum)
                {dum=fb;fb=fa;fa=dum;}
        }
        cx=(bx)+GOLD*(bx-ax);
        fc=f.func(cx);
                aa[0]=ax;
                aa[1]=bx;
                aa[2]=cx;
        while (fb > fc) {
                r=(bx-ax)*(fb-fc);
                q=(bx-cx)*(fb-fa);
                u=(bx)-((bx-cx)*q-(bx-ax)*r)/
                        (2.0*SIGN(MAX(Math.abs(q-r),TINY),q-r));
                ulim=(bx)+GLIMIT*(cx-bx);
                if ((bx-u)*(u-cx) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                ax=bx;
                                bx=u;
                                fa=fb;
                                fb=fu;
                                return aa;
                        } else if (fu > fb) {
                                cx=u;
                                fc=fu;
                                return aa;
                        }
                        u=(cx)+GOLD*(cx-bx);
                        fu=f.func(u);
                } else if ((cx-u)*(u-ulim) > 0.0) {
                        fu=f.func(u);
                        if (fu < fc) {
                                //SHFT(bx,cx,u,cx+GOLD*(cx-bx))
                                {bx=cx;cx=u;u=cx+GOLD*(cx-bx);}
                                //SHFT(fb,fc,fu,f.func(u))
                                {fb=fc;fc=fu;fu=f.func(u);}
                        }
                } else if ((u-ulim)*(ulim-cx) >= 0.0) {
                        u=ulim;
                        fu=f.func(u);
                } else {
                        u=cx+GOLD*(cx-bx);
                        fu=f.func(u);
                }
                //SHFT(ax,bx,cx,u)
                {ax=bx;bx=cx;cx=u;}
                //SHFT(fa,fb,fc,fu)
                {fa=fb;fb=fc;fc=fu;}
                aa[0]=ax;
                aa[1]=bx;
```

```
                        aa[2]=cx;
                }
                        return aa;
}
public  static double brent(f_x f,double ax,double bx,double cx,double tol)
{
 double aa[]=brentf(f,ax,bx,cx,tol);
 return aa[0];
}
// 4. fletcher Reeves metodu
//_____________________________________________________________________
public static double[] fletcher_reeves( f_xj f,double x[])
{
int i,j,ii,jj,k;
int nmax=500;
double gamma=1.0;
double tolerance=1.0e-4;
int n=x.length;
double alpha;
double g[]=new double[n];
double ge[]=new double[n];
double d[]=new double[n];
double beta;
double alpha_d;
double Q[]=new double[n];
int nn=15;
i=0;
j=0;k=0;
while( j++ < nmax && gamma > tolerance )
{
   g=turev(f,x);
   gamma=norm(g);
   if(k<1) {for(i=0;i<n;i++){d[i]=-g[i];};ge=g;}
   else
   {beta=VT_V(g)/VT_V(ge);
         for(i=0;i<n;i++){Q[i]=beta*d[i];};
   }
   for(i=0;i<n;i++){d[i]=-g[i]+Q[i];}
   alpha=linmin(f,x,d);
   for(i=0;i<n;i++){x[i]+=alpha*d[i];}
j++;k++;
if(k>=nn) k=0;
}
if(j >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
                public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }

                public static void main (String args[])
    {
                String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
```

```
    double [] x0=verigir(s);
    f1 f_x=new f1();
    double [] r1= fletcher_reeves(f_x,x0);
    s=" çözüm seti  : \n"+Matrix.toStringT(r1);
    String s2="Fletcher-Reeves çok değişkenli fonksiyon optimizasyonu:";
     System.out.println(s2+s);      }

}
```

$f(x)=(x_0-1)^2+(x_2-2)^2+(x_3-3)^2$ fonksiyonunun sınır şartı olmadan optimizasyonu, y=1.232595164407831E-32

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO12H
Fletcher-Reeves çok değişkenli fonksiyon optimizasyonu: optimizasyon değeri . 0.0
 çözüm seti  :
       1.000000000000000
       2.000000000000000
       3.000000000000000

> Terminated with exit code 0.
```

$f(x)=(x_0-1)^2+(x_2-2)^2+(x_3-3)^2$ fonksiyonunun $p(x)=x_1+x_2-2=0$ sınır şartıyla optimizasyonu,

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO12H
Fletcher-Reeves çok değişkenli fonksiyon optimizasyonu: optimizasyon değeri . 0.3333333333333332
 çözüm seti  :
       0.666666666666668
       1.666666666666666
       3.000000000000000

> Terminated with exit code 0.
```

$f(x)=(x_0-1)^2+(x_2-2)^2+(x_3-3)^2$ fonksiyonunun $q(x)=x_2-x_3-3 \geq 0$ sınır şartıyla optimizasyonu

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO12H
Fletcher-Reeves çok değişkenli fonksiyon optimizasyonu: optimizasyon değeri . 5.333336000000333
 çözüm seti  :
       1.000000000000000
       3.333333666666666
       1.666666333333334

> Terminated with exit code 0.
```

$f(x)=(x_0-1)^2+(x_2-2)^2+(x_3-3)^2$ fonksiyonunun $p(x)=x_1+x_2-2=0$ ve
$q(x)=x_2-x_3-3 \geq 0$ sınır şartıyla optimizasyonu, y=13.87126018223184

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO12H
Fletcher-Reeves çok değişkenli fonksiyon optimizasyonu: optimizasyon değeri . 7.375003250000374
 çözüm seti  :
       0.124999874999977
       2.750000249999014
       1.374999625000838

> Terminated with exit code 0.
```

Şimdi de aynı problemi genetik algoritmalara uyarlanmış bir versiyon olarak verelim.

Program 5.33.2 Genetik algoritma kullanarak optimizasyon metodu kullanılarak Sınırlandırılmış optimizasyon

```
import java.io.*;
class fa extends f_xj
{public double func(double x[])
{
//çözümü istenen fonksiyon
```

```
double ff;
double mu1=0.0;
double mu2=0.0;
ff=(x[0]-1.0)*(x[0]-1.0)+(x[1]-2.0)*(x[1]-2.0)+(x[2]-3.0)*(x[2]-3.0)+mu1*P(x)+mu2*Q(x);
return 400-ff;
}
public double P(double x[])
{// p(x)=0 sınırlama durumu
int n=x.length;
 return (x[0]+x[1]-2.0)*(x[0]+x[1]-2.0);
}
public double Q(double x[])
{// p(x)=0 sınırlama durumu
double eps=1.0e-6;
int n=x.length;
double ax=x[1]-x[2]-3.0-eps;
double bx=0;
if(ax<0) bx=ax*ax;
 return bx;
}
}

class OPO17H
{
   public static void main(String args[]) throws IOException
   {
   int N=18;
            int POPSIZE=200;
   int MAXGENS=200;
   int NVARS=3;
   double PXOVER=0.3;
   double PMUTATION=0.02;
   fa fx=new fa();
   Genetic1 xx=new Genetic1(POPSIZE,MAXGENS,NVARS,N,PXOVER,PMUTATION);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=0.0;high[0]=10.0;
   low[1]=0.0;high[1]=10.0;
   low[2]=0.0;high[2]=10.0;
   xx.setPopulation(low,high,fx);
   System.out.println("best=\n"+Matrix.toStringT(xx.calculate(fx,false)));
   }
}
```

$f(x)=(x_0-1)^2+(x_2-2)^2+(x_3-3)^2$ fonksiyonunun sınır şartı olmadan genetik algoritmalar kullanılarak optimizasyonu,

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO17H
best=
       1.003612532091263
       2.003181469655875
       3.008701357655936
     399.999901114237700
> Terminated with exit code 0.
```

$f(x)=(x_0-1)^2+(x_2-2)^2+(x_3-3)^2$ fonksiyonunun $p(x)=x_1+x_2-2=0$ sınır şartıyla genetik algoritmalar kullanılarak optimizasyonu,

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO17H
best=
       0.722659006725337
       1.562429666250863
       2.968723177807532
     399.649360385406300
```

```
> Terminated with exit code 0.
```

$f(x)=(x_0-1)^2+(x_2-2)^2+(x_3-3)^2$ fonksiyonunun $q(x)=x_2-x_3-3 \geq 0$ sınır şartıyla genetik algoritmalar kullanılarak optimizasyonu,

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO17H
best=
      1.015628874316690
      3.333447774687861
      1.666876475816634
    394.666419671884100
> Terminated with exit code 0.
```

$f(x)=(x_0-1)^2+(x_2-2)^2+(x_3-3)^2$ fonksiyonunun $p(x)=x_1+x_2-2=0$ ve
$q(x)=x_2-x_3-3 \geq 0$ sınır şartıyla genetik algoritmalar kullanılarak optimizasyonu

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO17H
best=
      0.249901771170697
      2.499971389661368
      1.250042915507948
    392.499967856004000
> Terminated with exit code 0.
```

5.34 Gezgin satıcı problemi

Gezgin Satıcı problemi (Travelling Salesman Problem-TSP) bir çok yönleri ile özel olan bir optimizasyon problemidir. Herşeyden önce günümüzde oldukça yoğun kullanılan bir problem olması ilginçliğini arttırır. Problem göreceli olarak basittir. Bir satıcı n şehre uğrayarak mallarını satacaktır. Şehirlerin koordinatları bellidir. Satıcı en kısa yolu giderek bu turu nasıl yapailir. Problemi iki temel alt versiyona ayırabiliriz. Birinci versiyonda başlangıç ve bitiş noktaları belirlenmiş olabilir. Örneğin 2 şehir arasında bir telefon (veya internet) bağlantısı kurulacaksa bu versiyon geçerlidir. İkinci versiyonda başlangıç ve bitiş noktaları verilen listeden en kısa yolu oluşturacak şekilde seçilebilir. Bu tür problemlere genel olarak tamsayı(kombinatörel) optimizasyon problemleri adı da verilir.
Formülle tanım yapmak istersek

$$Mesafe=\sum_{i=1}^{n}\left[(x_i-x_{i-1})^2+(y_i-y_{i-1})^2\right]^{1/2}$$ burada x_i ve y_i i'nci şehrin koordinatlarıdır. Aynı formülü sabit başlangıç vebitiş koordinatları için yazarsak:

$$Mesafe=\left[(x_0-x_b)^2+(y_0-y_b)^2\right]^{1/2}+\sum_{i=1}^{n}\left[(x_i-x_{i-1})^2+(y_i-y_{i-1})^2\right]^{1/2}+$$

$$\left[(x_{bitiş}-x_n)^2+(y_{bitiş}-y_n)^2\right]^{1/2}$$

Olur. Buradaki (x_b, y_b) seyahatin başlama koordinatları, ($x_{bitiş}$, $y_{bitiş}$) seyahatin bitiş noktalarıdır. Gezgin satıcı (Travelling Salesman Problem- kısaca TSP) probleminin normal versiyonunda aynı şehrin sadece bir kere ziyaret edilmesi zorunluğu vardır, ancak bazı gerçek problemlerde bu zorunluk mevcut değildir. Bu problemin çözümünde her şehir için geri kalan şehirlerin uzaklıklarını hesap ederek en yakı olanını seçmek kullanılabilecek bir yaklaşım gibi görülsede şehir sayısı 20 lerin üzerine çıktığında olasılıklar çok büyüdüğünden kullanılabilir bi metod oluşturmaz. Şimdi değişik Gezgin satıcı problem çözümlerine göz atalım. İlk önce Genetik algoritmalara göz atacağız.

5.34.1 Gezgin satıcı problemi genetik algoritma çözümü

Genetik algoritmalar Gezgin satıcı (TSP) probleminin çözümünde oldukça yararlı görülmektedir. Ancak bu problem için kendine özgü şartlar altında geliştirilmelidir. Burada vereceğimiz örneklerde temel geni geliştirmek için javadaki ArrayList yapısını kullanarak verilen şehirlerin sıralanma sırasını oluşturur. Bunun için önce seyehat edilecek şehirlerin koordiatları liste olarak verilir. Örneğin alttaki sehirler.txt dosyasında olduğu gibi

```
100.0     0.0
100.0     0.0
95.10565162951535         30.901699437494738
80.90169943749474         58.778525229247315
58.778525229247315        80.90169943749474
30.901699437494745        95.10565162951535
6.123233995736766E-15 100.0
-30.901699437494734       95.10565162951536
-58.7785252292473         80.90169943749474
-80.90169943749473        58.77852522924732
-95.10565162951535        30.901699437494752
-100.0    1.2246467991473532E-14
-95.10565162951538        -30.90169943749469
-80.90169943749474        -58.7785252292473
-58.77852522924732        -80.90169943749473
-30.901699437494756       -95.10565162951535
-1.8369701987210297E-14           -100.0
30.901699437494724        -95.10565162951536
58.77852522924729         -80.90169943749476
80.90169943749473         -58.778525229247336
95.10565162951535         -30.901699437494763
```

Buradaki ilk satır başlama noktası koordinatlarını, ikinci satır ise son varılacak şehrin koordinatlarını vermektedir. Sonraki koordinatlar ziyaret edilebilecek şehirlerin koordinatlarıdır. Dosyada ikinci sıradan sonraki şehir koordinatları aynı zamanda sıralamayı vermektedir. Örneğin dosyadaki üçüncü sıradaki şehrin numarası 0 dördüncü satırdaki şehrin numarası 1, beşinci satırdaki şehrin numarası 2 dir. Eğer ArayList değişkenindeki listede

2	7	3	8	1	4	6	5

gibi bir liste mevcutsa ziyaret **Başlangıç şehri-2-7-3-8-1-4-6-5-bitiş şehri** şeklinde olacaktır. Listenin programdaki düzenleme şekli aynı şehrin 2 kere tekrarlamasını engellemektedir. Programımız satici_gen ve satici_genetik olarak 2 sınıftan oluşmaktadır. 1. Sınıf yukarda verdiğimiz en yapısını tanımlar, bu genin çoğalması(anne ve baba bireylerden çocuk bireyinin gen yapısını-ArayList dizgisini oluşturmak) ve mutasyon oluştuğunda oluşacak yapıları oluşturma görevini üstlenmiştir. Her bireyin sadece 1 geni vardır. İkinci programımız satici_genetik verilen nüfus için yeni bireyleri oluşturmaka, gerekirse olasılıklara göre mutasyona izin vermekte ve istenilen kadar yeni nesil üretmektedir. Her nesilde en iyi çözüm yeni nesle değişmeksizin aktarılmaktadır. Seyahati daha iyi izleyebilmek için satici_gen programına plot() metodu da eklenmiştir. GA nın istatiksel karekteristiğinden dolayı sonucun en iyi sonucayaklaşacağını umabiliriz, ancak her zaman en iyi sonucu yakalayamıyacağımızı hatta her zaman aynı sonucu yakalayamıyacağımızı hatırlatmakta yarar görüyoruz. Altaki ilk listede satici_gen.java programı, İkinci listede satici_genetik.java programı verilmiştir.

Program 5.34.1 satici_gen.java programı gezgin satıcı GA algoritması için gen yapısını oluşturur.

```
import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
//çıkış ve varış noktaları belirlenmiş tur
```

```
public class satici_gen
{
public int n;
public double xcikilan,ycikilan;
public double xvarilan,yvarilan;
public double x[],y[];
public double Fitness;    // uyum değeri
public double RFitness;   // relatif uyum
public double CFitness;   // kümülatif uyum
public boolean Breeder;

ArrayList<String> v;
String dizi;

public satici_gen(double xi[],double yi[])
{n=xi.length-2;
v=new ArrayList<String>(n);
x=new double[n];
y=new double[n];
xcikilan=xi[0];
ycikilan=yi[0];
xvarilan=xi[1];
yvarilan=yi[1];
Breeder=false;
for(int i=0;i<n;i++) {x[i]=xi[i+2];y[i]=yi[i+2];v.add(""+i);}
Collections.shuffle(v);
//swap();
mesafe();
}

public void mutate()
{
int n1=(int)(n*Math.random());
int n2=(int)(n*Math.random());
//System.out.println("n="+n+"n1="+n1+"n2="+n2);
if(n1!=n2) Collections.swap(v, n1, n2);
}


public void swap()
{int pos[]=new int[n];
 int key[]=new int[n];
 int p,k;
 int n1=0;
for(int i=0;i<n;i++)
{ p=Collections.binarySearch(v,""+i);
  if(p==i) {pos[i]=p;n1++;}
}
for(int i=0;i<n1;i+=2)
{Collections.swap(v,pos[i],pos[i+1]);}
}
```

```
public satici_gen(satici_gen anne)
{n=anne.n;
 v=kopye(anne.v);
 xcikilan=anne.xcikilan;
 ycikilan=anne.ycikilan;
 xvarilan=anne.xvarilan;
 yvarilan=anne.yvarilan;
 x=new double[n];
 y=new double[n];
 for(int i=0;i<n;i++) {x[i]=anne.x[i];y[i]=anne.y[i];}
 mesafe();
}

public satici_gen(int a[],double xi[],double yi[])
{
ArrayList<String> anne=add(a);
v=kopye(anne);
Collection torba=v;
n=xi.length-2;
xcikilan=xi[0];
ycikilan=yi[0];
xvarilan=xi[1];
yvarilan=yi[1];
x=new double[n];
y=new double[n];
for(int i=0;i<n;i++) {x[i]=xi[i+2];y[i]=yi[i+2];}
mesafe();
}

public satici_gen(String dosya)
{double a[][]=Text.readDouble(dosya);
n=a[0].length-2;
v=new ArrayList<String>(n);
x=new double[n];
y=new double[n];
xcikilan=a[0][0];
ycikilan=a[1][0];
xvarilan=a[0][1];
yvarilan=a[1][1];
for(int i=0;i<n;i++) {x[i]=a[0][i+2];y[i]=a[1][i+2];v.add(""+i);}
Collections.shuffle(v);
mesafe();
}
        public satici_gen(satici_gen anne, satici_gen baba)
   {
   n=anne.n;
   xcikilan=anne.xcikilan;
   ycikilan=anne.ycikilan;
   xvarilan=anne.xvarilan;
   yvarilan=anne.yvarilan;
   x=new double[n];
   y=new double[n];
```

```
    for(int i=0;i<n;i++)
    {x[i]=anne.x[i];y[i]=anne.y[i];}
    ArrayList<String> cocuk=s_cocuk(anne.v,baba.v);
            v=kopye(cocuk);
    mesafe();
            }

public satici_gen(ArrayList<String> anne,double xi[],double yi[])
{
v=kopye(anne);
Collection torba=v;
n=xi.length-2;
xcikilan=xi[0];
ycikilan=yi[0];
xvarilan=xi[1];
yvarilan=yi[1];
x=new double[n];
y=new double[n];
for(int i=0;i<n;i++) {x[i]=xi[i+2];y[i]=yi[i+2];}
mesafe();
}

public void plot()
{
double xi[]=new double[n+2];
double yi[]=new double[n+2];
    int a=0;
    int b=0;
            String s1="";
            String s2="";
    Iterator i=v.iterator();
    int j=0;
    double mesafe;
    dizi="0";
    s1=(String)i.next();
    a=Integer.parseInt(s1);
    mesafe=Math.sqrt((x[a]-xcikilan)*(x[a]-xcikilan)+(y[a]-ycikilan)*(y[a]-ycikilan));
    xi[j]=xcikilan;
    yi[j]=ycikilan;
    j++;
    xi[j]=x[a];
    yi[j]=y[a];
    j++;
    while(i.hasNext())
    {
    s2=(String)i.next();
    b=Integer.parseInt(s2);

    mesafe+=Math.sqrt((x[a]-x[b])*(x[a]-x[b])+(y[a]-y[b])*(y[a]-y[b]));
    xi[j]=x[b];
    yi[j]=y[b];
    j++;
```

```
    a=b;
    }
    xi[j]=xvarilan;
    yi[j]=yvarilan;
    mesafe+=Math.sqrt((x[b]-xvarilan)*(x[b]-xvarilan)+(y[b]-yvarilan)*(y[b]-yvarilan));
    Fitness=1.0/mesafe;
    Plot pp=new Plot(xi,yi);
    double wx[]=new double[2];
    double wy[]=new double[2];
    wx[0]=xi[0];
    wy[0]=yi[0];
    wx[1]=xi[n+1];
    wy[1]=yi[n+1];
    pp.addData(wx,wy);
    pp.addData(xi,yi);
    pp.setPlotType(1,32);
    pp.setPlotType(2,22);
    pp.setPlabel("Genetik satıcı problemi mesafe = "+mesafe);
    pp.setXlabel("x, km");
    pp.setYlabel("y, km");
    pp.plot();
}

public double mesafe()
{
    int a=0;
    int b=0;
            String s1="";
            String s2="";
    Iterator i=v.iterator();
    int j=0;
    double mesafe;
    dizi="0";
    s1=(String)i.next();
    a=Integer.parseInt(s1);
    mesafe=Math.sqrt((x[a]-xcikilan)*(x[a]-xcikilan)+(y[a]-ycikilan)*(y[a]-ycikilan));
    while(i.hasNext())
    {
    s2=(String)i.next();
    b=Integer.parseInt(s2);
    mesafe+=Math.sqrt((x[a]-x[b])*(x[a]-x[b])+(y[a]-y[b])*(y[a]-y[b]));
    a=b;
    }
    mesafe+=Math.sqrt((x[b]-xvarilan)*(x[b]-xvarilan)+(y[b]-yvarilan)*(y[b]-yvarilan));
    Fitness=1.0/mesafe;
    return mesafe;
}

public void setFitness()
{Fitness=1.0/mesafe();}

public double getFitness()
```

```
{return Fitness;}

public void setRFitness(double RFitnessi)
{RFitness=RFitnessi;}

public double getRFitness()
{return RFitness;}

public void setCFitness(double CFitnessi)
{CFitness=CFitnessi;}

public double getCFitness()
{return CFitness;}
 public static String print(Collection c)
 {
  String s="";
  Iterator i=c.iterator();
  while(i.hasNext()) {s+=i.next()+"\n";}
  return s;
 }

 public String dizi()
 {return print(v);}

 public ArrayList<String>kopye(ArrayList<String>list1)
  {
     ArrayList<String> bilesimVectoru = new ArrayList<String>();
     String s1,s2;
     Iterator n1=list1.iterator();
     while(n1.hasNext())
     {
        s1=(String)n1.next();
        bilesimVectoru.add(s1);
     }
     return bilesimVectoru;
  }

  public static ArrayList<String> add(int a[])
  {
     ArrayList<String> bilesimVectoru = new ArrayList<String>();
     String s1;
     for(int i=0;i<a.length;i++)
     {
        s1=""+a[i];
        if(!bilesimVectoru.contains(s1))
           bilesimVectoru.add(s1);
     }
     return bilesimVectoru;
  }

  public ArrayList<String> bilesim(ArrayList<String>list1, int a)
  {
```

```
    ArrayList<String> bilesimVectoru = new ArrayList<String>();
    String s1,s2;
    Iterator n1=list1.iterator();
    String b=new String(""+a);
    while(n1.hasNext())
    {
       s1=(String)n1.next();
       bilesimVectoru.add(s1);
    }
    s2=(String)b;
       if(!bilesimVectoru.contains(s2))
          bilesimVectoru.add(s2);
    return bilesimVectoru;
}

public ArrayList<String> bilesim(ArrayList<String> list1, ArrayList<String> list2)
{
    ArrayList<String> bilesimVectoru = new ArrayList<String>();
    String s1,s2;
    Iterator n1=list1.iterator();
    Iterator n2=list2.iterator();
    while(n1.hasNext())
    {
       s1=(String)n1.next();
       bilesimVectoru.add(s1);
    }
    while(n2.hasNext())
    {
       s2=(String)n2.next();
       if(!bilesimVectoru.contains(s2))
          bilesimVectoru.add(s2);
    }
    return bilesimVectoru;
}

public  ArrayList<String> altküme(ArrayList<String> list1, int nn)
{
    ArrayList<String> bilesimVectoru = new ArrayList<String>();
    String s1,s2;
    Iterator n1=list1.iterator();
    for(int i=0;i<nn;i++)
    {
       s1=(String)n1.next();
       bilesimVectoru.add(s1);
    }
    return bilesimVectoru;
}

public  ArrayList<String> kalanküme(ArrayList<String> list1, int nn)
{
    ArrayList<String> bilesimVectoru = new ArrayList<String>();
    String s1,s2;
```

```
    Iterator n1=list1.iterator();

    for(int i=0;i<nn;i++)
    { s1=(String)n1.next();}

    while(n1.hasNext())
    {
       s1=(String)n1.next();
       bilesimVectoru.add(s1);
    }
    return bilesimVectoru;
}

public  ArrayList<String> s_cocuk(ArrayList<String> anne, ArrayList<String> baba)
{int nn=(int)(n*Math.random())+1;
//System.out.println("nn="+nn);
          ArrayList<String> anne1=altküme(anne,nn);
          ArrayList<String> anne2=kalanküme(anne,nn);
          ArrayList<String> baba1=altküme(baba,nn);
          ArrayList<String> baba2=kalanküme(baba,nn);
          ArrayList<String> cocuk=bilesim(anne1,baba1);
          ArrayList<String> cocuk1=bilesim(cocuk,anne2);
          ArrayList<String> cocuk2=bilesim(cocuk1,baba2);
          return cocuk2;
         }

        public String ciktiS()
{
    Iterator enm = v.iterator();
    String s="  ("+xcikilan+","+ycikilan+")\n";
    String s1="";
    int a=0;
    while(enm.hasNext())
    {  s1=(String)enm.next();
      a=Integer.parseInt(s1);
              s+=s1+" ("+x[a]+","+y[a]+") \n";
           }
           s+="  ("+xvarilan+","+yvarilan+") \n";
return s;
}

        public String[] ciktiSA()
{
    Iterator enm = v.iterator();
    String s[]=new String[n];
    int i=0;
    while(enm.hasNext())
    {  s[i]= (String)enm.next()+" ";i++;}
return s;
}

public String toString()
```

```
   {
          String s=ciktiS()+" mesafe = "+mesafe();
   return s;
   }
}
```

Program 5.34.2 satici_genetik.java programı gezgin satıcı GA algoritmasını çözer. Başlangıç ve bitiş noktaları sabitlenmiştir.

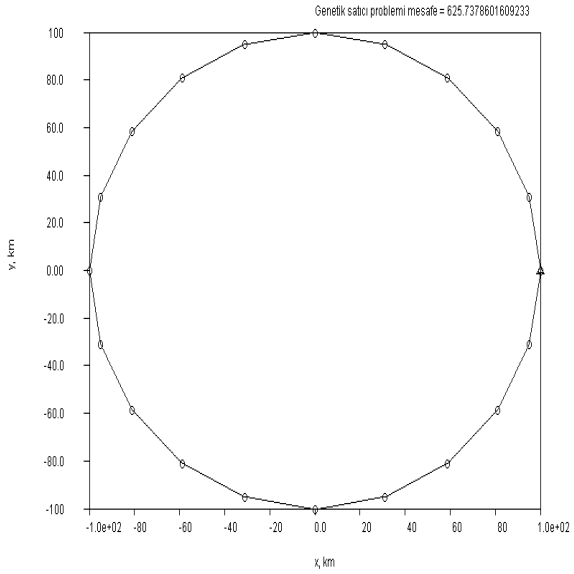
```
import java.io.*;
import java.text.*;
import java.util.Locale;

class satici_genetik
{
public satici_gen G[];    // string of variables makes a genotype1
public boolean Breeder;   //
public int    POPSIZE;          // population size
int    MAXGENS;          // maximum number of generations;
double PMUTATION;        // proportion of mutated variables
double PXOVER;           // probability of crossover;
int    Generation;       // Current generation number
int    NBreeder;   //   Number of survivors in a generation
int    Best; //The best genotype1 in the population
double toplameniyi=0;
double nesileniyi=0;
double xi[],yi[]; //şehir adresleri

 public satici_genetik(String name,int POPSİZEi,int MAXGENSi,double PXOVERi,double PMUTATIONi )
 {double a[][]=Text.readDoubleT(name);
  double xi[]=a[0];
  double yi[]=a[1];
  int best;
  POPSIZE=POPSİZEi;
  PXOVER=PXOVERi;
  MAXGENS=MAXGENSi;
  PMUTATION=PMUTATIONi;
  G=new satici_gen[POPSIZE+1];
  double totalfitness=0;
  for(int i=0;i<(POPSIZE+1);i++)
  {G[i]=new satici_gen(xi,yi);G[i].setFitness();totalfitness+=G[i].getFitness();}
  double xR,xC=0;
  for(int i=0;i<(POPSIZE+1);i++)
  {xR=G[i].getRFitness()/totalfitness;xC+=xR;G[i].setRFitness(xR);G[i].setCFitness(xC);}
 }

 public void setPopulation()
 {
  double totalfitness=0;
  for(int i=0;i<POPSIZE;i++)
  {G[i]=new satici_gen(xi,yi);}
   evaluate();
 }
```

```
 public void evaluate()
 {
  nesileniyi=0;
  double totalfitness=0;
  for(int i=0;i<POPSIZE+1;i++)
  {G[i].setFitness();
   double mesafe=1.0/G[i].getFitness();
   if(mesafe>nesileniyi) nesileniyi=mesafe;
   if(mesafe>toplameniyi) toplameniyi=mesafe;
   totalfitness+=G[i].getFitness();
  }
  double xR=0,xC=0,xRP=G[POPSIZE].getFitness()/totalfitness;
  for(int i=0;i<POPSIZE;i++)
  {xR=G[i].getFitness()/totalfitness;xC+=xR;G[i].setRFitness(xR);G[i].setCFitness(xC);
            if(xR > xRP)
               {
                        Best=i;
                        G[POPSIZE]=new satici_gen(G[i]);
                        xRP=G[POPSIZE].getFitness()/totalfitness;
       }
 }
}
 public satici_gen[] copyPopulation()
 {
            satici_gen Po[]=new satici_gen[POPSIZE+1];
            for(int i=0;i<(POPSIZE+1);i++)
              { Po[i]=new satici_gen(G[i]);
                Po[i].Fitness=G[i].Fitness;
                        Po[i].RFitness=G[i].RFitness;
                Po[i].CFitness=G[i].CFitness;
              }
            return Po;
 }

            public void select()
            {
            double r,r1,r2,gf0,gf1;
            evaluate();
            //anne baba aday adaylarını seç
            satici_gen Po[]=new satici_gen[POPSIZE+1];
            NBreeder=0;
            int i,j;
            for(i=0;i<(POPSIZE+1);i++)
   {   r=Math.random();
       gf0=0;
       for(j=0;j<POPSIZE;j++)
       {
       gf1=G[j].getCFitness();
       if((r>=gf0) && (r<=gf1))
       {Po[NBreeder]=new satici_gen(G[j]);
       //System.out.println("j= "+j+"gf0="+gf0+"gf1="+gf1+"r="+r+"NBreeder="+NBreeder);
```

```
      NBreeder++;break;}
      gf0=gf1;
      }
  }
  // seçilen anne, baba aday adaylarının yeni çocuk yapmalarına olanak ver
   int breed=(int)(POPSIZE*PXOVER);
   for(i=0;i<breed;i++)
          {
                  int a=(int)(Math.random()*NBreeder);
       int b=(int)(Math.random()*NBreeder);
       G[i]=new satici_gen(Po[a],Po[b]);
       //System.out.println("a="+a+"b="+b+"cocuk="+G[i].toString());
       r2=Math.random();
       //mutasyon olasılığı : PMUTATION
            if(r2 < PMUTATION) G[i].mutate();
            //System.out.println(NBreeder+" "+G[POPSIZE].mesafe());
          }
          //listenin gerisini seçilen anne baba adaylarıyla doldur
            for(i=breed;i<POPSIZE;i++)
            {  int a=(int)(Math.random()*NBreeder);
        G[i]=new satici_gen(Po[a]);
            }
  }

  public void nesiller()
  {
         for(int i=0;i<MAXGENS;i++) {select();}
         }

  public satici_gen secilmis_nesil()
  {
  nesiller();
  G[POPSIZE].plot();
  return G[POPSIZE];
         }


 public static void main(String arg[])
 {
  satici_genetik sg=new satici_genetik("sehirler.txt",1000,1000,1.0,0.05 );
  System.out.println("seçilmiş nesil : "+sg.secilmis_nesil().toString());
 }

}
```

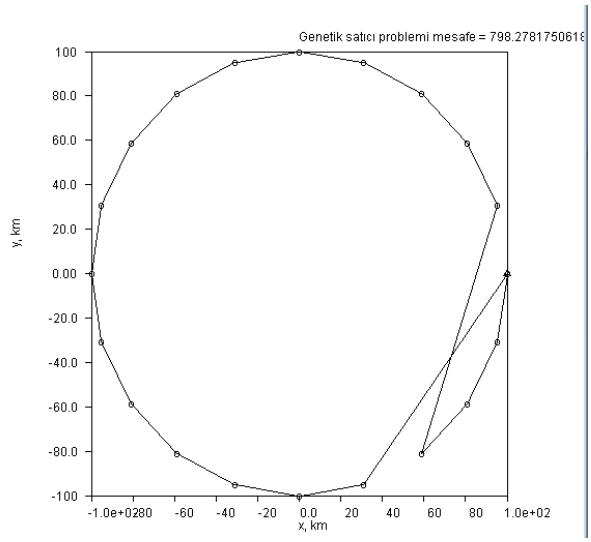
İlk sonucumuz tam doğru bir sonuç vermektedir, çünkü test verimiz bir dairenin etrafındaki aynı şehirde başlayan vebiten bir turu gösteriyordu.

Gezgin satıcı problemi GA satici_genetik çözüm 1 grafik çıktı sabit başlangıç ve bitiş şehirleri verilmiş.



(bolum5_042.jpg,Resimaltı **:** Gezgin satıcı problemi GA satici_genetik çözüm 1 grafik çıktı sabit başlangıç ve bitiş şehirleri verilmiş.)

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" satici_genetik
seçilmiş nesil :   (100.0,0.0)
0 (95.10565162951535,30.901699437494738)
1 (80.90169943749474,58.778525229247315)
2 (58.778525229247315,80.90169943749474)
3 (30.901699437494745,95.10565162951535)
4 (6.123233995736766E-15,100.0)
5 (-30.901699437494734,95.10565162951536)
6 (-58.7785252292473,80.90169943749474)
7 (-80.90169943749473,58.77852522924732)
8 (-95.10565162951535,30.901699437494752)
9 (-100.0,1.2246467991473532E-14)
10 (-95.10565162951538,-30.90169943749469)
11 (-80.90169943749474,-58.7785252292473)
12 (-58.77852522924732,-80.90169943749473)
13 (-30.901699437494756,-95.10565162951535)
14 (-1.8369701987210297E-14,-100.0)
15 (30.901699437494724,-95.10565162951536)
16 (58.77852522924729,-80.90169943749476)
17 (80.90169943749473,-58.778525229247336)
18 (95.10565162951535,-30.901699437494763)
  (100.0,0.0)
 mesafe = 625.7378601609233
```



(bolum5_043.jpg,Resimaltı **:** Gezgin satıcı problemi GA satici_genetik çözüm 2 grafik çıktı sabit başlangıç ve bitiş şehirleri)

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" satici_genetik
seçilmiş nesil :   (100.0,0.0)
18 (95.10565162951535,-30.901699437494763)
17 (80.90169943749473,-58.778525229247336)
16 (58.77852522924729,-80.90169943749476)
0 (95.10565162951535,30.901699437494738)
1 (80.90169943749474,58.778525229247315)
2 (58.778525229247315,80.90169943749474)
3 (30.901699437494745,95.10565162951535)
4 (6.123233995736766E-15,100.0)
5 (-30.901699437494734,95.10565162951536)
6 (-58.7785252292473,80.90169943749474)
7 (-80.90169943749473,58.77852522924732)
8 (-95.10565162951535,30.901699437494752)
9 (-100.0,1.2246467991473532E-14)
10 (-95.10565162951538,-30.90169943749469)
11 (-80.90169943749474,-58.7785252292473)
12 (-58.77852522924732,-80.90169943749473)
13 (-30.901699437494756,-95.10565162951535)
14 (-1.8369701987210297E-14,-100.0)
15 (30.901699437494724,-95.10565162951536)
  (100.0,0.0)
 mesafe = 798.2781750618202
```

Eğer gezgin satıcı probleminde başlangıç ve bitiş noktaları sabitlenmemişse, yani tura herhangi bir şehirden başlanabiliyorsa ve tur herhangi bir şehirde bitebiliyorsa programlama kodu bir üsttekine benzer ancak biraz değişiktir. İlk ve son şehir koordinatları hem okunan listeden hem de program kodundan kaldırılarak hesaplamalar gerçekleştirilir. Program kodları satici_gen1 ve satici_genetik1 altta verilmiştir.

Program 5.34.3 satici_gen1.java programı gezgin satıcı GA algoritması için gen yapısını oluşturur. Başlangıç ve bitiş noktaları sabitlenmemiştir.

```
import java.util.*;
import java.awt.*;
```

```
import javax.swing.*;
import java.awt.event.*;
//çıkış ve varış noktaları belirlenmemiş tur
public class satici_gen1
{
public int n;
public double x[],y[];
public double Fitness;    // uyum değeri
public double RFitness;   // relatif uyum
public double CFitness;   // kümülatif uyum
public boolean Breeder;

ArrayList<String> v;
String dizi;

public satici_gen1(double xi[],double yi[])
{n=xi.length;
v=new ArrayList<String>(n);
x=new double[n];
y=new double[n];
Breeder=false;
for(int i=0;i<n;i++) {x[i]=xi[i];y[i]=yi[i];v.add(""+i);}
Collections.shuffle(v);
mesafe();
}

public void mutate()
{
int n1=(int)(n*Math.random());
int n2=(n1+1)%n;
//int n2=(int)(n*Math.random());
Collections.swap(v, n1, n2);
}

public void swap()
{int pos[]=new int[n];
 int key[]=new int[n];
 int p,k;
 int n1=0;
for(int i=0;i<n;i++)
{ p=Collections.binarySearch(v,""+i);
  if(p==i) {pos[i]=p;n1++;}
}
for(int i=0;i<n1;i+=2)
{Collections.swap(v,pos[i],pos[i+1]);}
}

public satici_gen1(satici_gen1 anne)
{n=anne.n;
 v=kopye(anne.v);
 x=new double[n];
 y=new double[n];
```

```
 for(int i=0;i<n;i++) {x[i]=anne.x[i];y[i]=anne.y[i];}
 mesafe();
}

public satici_gen1(int a[],double xi[],double yi[])
{
ArrayList<String> anne=add(a);
v=kopye(anne);
Collection torba=v;
n=xi.length;
x=new double[n];
y=new double[n];
for(int i=0;i<n;i++) {x[i]=xi[i];y[i]=yi[i];}
mesafe();
}

public satici_gen1(String dosya)
{double a[][]=Text.readDouble(dosya);
n=a[0].length;
v=new ArrayList<String>(n);
x=new double[n];
y=new double[n];
for(int i=0;i<n;i++) {x[i]=a[0][i];y[i]=a[1][i];v.add(""+i);}
Collections.shuffle(v);
mesafe();
}
        public satici_gen1(satici_gen1 anne, satici_gen1 baba)
   {
   n=anne.n;
   x=new double[n];
   y=new double[n];
   for(int i=0;i<n;i++)
   {x[i]=anne.x[i];y[i]=anne.y[i];}
   ArrayList<String> cocuk=s_cocuk(anne.v,baba.v);
        v=kopye(cocuk);
   mesafe();
        }

public satici_gen1(ArrayList<String> anne,double xi[],double yi[])
{
v=kopye(anne);
Collection torba=v;
n=xi.length;
x=new double[n];
y=new double[n];
for(int i=0;i<n;i++) {x[i]=xi[i];y[i]=yi[i];}
mesafe();
}

public void plot()
{
double xi[]=new double[n];
```

```
double yi[]=new double[n];
   int a=0;
   int b=0;
         String s1="";
         String s2="";
   Iterator i=v.iterator();
   int j=0;
   double mesafe=0;
   dizi="0";
   s1=(String)i.next();
   a=Integer.parseInt(s1);
   xi[j]=x[a];
   yi[j]=y[a];
   j++;
   while(i.hasNext())
   {
   s2=(String)i.next();
   b=Integer.parseInt(s2);
   mesafe+=Math.sqrt((x[a]-x[b])*(x[a]-x[b])+(y[a]-y[b])*(y[a]-y[b]));
   xi[j]=x[b];
   yi[j]=y[b];
   j++;
   a=b;
   }
   Plot pp=new Plot(xi,yi);
   double wx[]=new double[2];
   double wy[]=new double[2];
   wx[0]=xi[0];
   wy[0]=yi[0];
   wx[1]=xi[n-1];
   wy[1]=yi[n-1];
   pp.addData(wx,wy);
   pp.addData(xi,yi);
   pp.setPlotType(1,32);
   pp.setPlotType(2,22);
   pp.setPlabel("Genetik satıcı problemi mesafe = "+mesafe);
   pp.setXlabel("x, km");
   pp.setYlabel("y, km");
   pp.plot();
}

public double mesafe()
{
   int a=0;
   int b=0;
         String s1="";
         String s2="";
   Iterator i=v.iterator();
   int j=0;
   double mesafe;
   dizi="0";
   s1=(String)i.next();
```

```
   a=Integer.parseInt(s1);
   mesafe=0.0;
   while(i.hasNext())
   {
   s2=(String)i.next();
   b=Integer.parseInt(s2);
   mesafe+=Math.sqrt((x[a]-x[b])*(x[a]-x[b])+(y[a]-y[b])*(y[a]-y[b]));
   a=b;
   }
   Fitness=1.0/mesafe;
   return mesafe;
}

public void setFitness()
{Fitness=1.0/mesafe();}

public double getFitness()
{return Fitness;}

public void setRFitness(double RFitnessi)
{RFitness=RFitnessi;}

public double getRFitness()
{return RFitness;}

public void setCFitness(double CFitnessi)
{CFitness=CFitnessi;}

public double getCFitness()
{return CFitness;}
 public static String print(Collection c)
 {
   String s="";
   Iterator i=c.iterator();
   while(i.hasNext()) {s+=i.next()+"\n";}
   return s;
 }

 public String dizi()
 {return print(v);}

 public ArrayList<String>kopye(ArrayList<String>list1)
   {
      ArrayList<String> bilesimVectoru = new ArrayList<String>();
      String s1,s2;
      Iterator n1=list1.iterator();
      while(n1.hasNext())
      {
         s1=(String)n1.next();
         bilesimVectoru.add(s1);
      }
      return bilesimVectoru;
```

```
}

public static ArrayList<String> add(int a[])
{
   ArrayList<String> bilesimVectoru = new ArrayList<String>();
   String s1;
   for(int i=0;i<a.length;i++)
   {
      s1=""+a[i];
      if(!bilesimVectoru.contains(s1))
         bilesimVectoru.add(s1);
   }
   return bilesimVectoru;
}

public ArrayList<String> bilesim(ArrayList<String>list1, int a)
{
   ArrayList<String> bilesimVectoru = new ArrayList<String>();
   String s1,s2;
   Iterator n1=list1.iterator();
   String b=new String(""+a);
   while(n1.hasNext())
   {
      s1=(String)n1.next();
      bilesimVectoru.add(s1);
   }
   s2=(String)b;
      if(!bilesimVectoru.contains(s2))
         bilesimVectoru.add(s2);
   return bilesimVectoru;
}

public ArrayList<String> bilesim(ArrayList<String> list1, ArrayList<String> list2)
{
   ArrayList<String> bilesimVectoru = new ArrayList<String>();
   String s1,s2;
   Iterator n1=list1.iterator();
   Iterator n2=list2.iterator();
   while(n1.hasNext())
   {
      s1=(String)n1.next();
      bilesimVectoru.add(s1);
   }
   while(n2.hasNext())
   {
      s2=(String)n2.next();
      if(!bilesimVectoru.contains(s2))
         bilesimVectoru.add(s2);
   }
   return bilesimVectoru;
}
```

```
public  ArrayList<String> altküme(ArrayList<String> list1, int nn)
{
   ArrayList<String> bilesimVectoru = new ArrayList<String>();
   String s1,s2;
   Iterator n1=list1.iterator();
   for(int i=0;i<nn;i++)
   {
      s1=(String)n1.next();
      bilesimVectoru.add(s1);
   }
   return bilesimVectoru;
}

public  ArrayList<String> kalanküme(ArrayList<String> list1, int nn)
{
   ArrayList<String> bilesimVectoru = new ArrayList<String>();
   String s1,s2;
   Iterator n1=list1.iterator();

   for(int i=0;i<nn;i++)
   { s1=(String)n1.next();}

   while(n1.hasNext())
   {
      s1=(String)n1.next();
      bilesimVectoru.add(s1);
   }
   return bilesimVectoru;
}

public  ArrayList<String> s_cocuk(ArrayList<String> anne, ArrayList<String> baba)
{int nn=(int)(n*Math.random())+1;
//System.out.println("nn="+nn);
          ArrayList<String> anne1=altküme(anne,nn);
          ArrayList<String> anne2=kalanküme(anne,nn);
          ArrayList<String> baba1=altküme(baba,nn);
          ArrayList<String> baba2=kalanküme(baba,nn);
          ArrayList<String> cocuk=bilesim(anne1,baba1);
          ArrayList<String> cocuk1=bilesim(cocuk,anne2);
          ArrayList<String> cocuk2=bilesim(cocuk1,baba2);
          return cocuk2;
         }

        public String ciktiS()
{
   Iterator enm = v.iterator();
   String s="";
   int a=0;
   String s1="";
   while(enm.hasNext())
   {  s1=(String)enm.next();
     a=Integer.parseInt(s1);
```

```
                s+=s1+" ("+x[a]+","+y[a]+") \n";
              }
   return s;
   }

          public String[] ciktiSA()
   {
      Iterator enm = v.iterator();
      String s[]=new String[n];
      int i=0;
      while(enm.hasNext())
      {  s[i]= (String)enm.next()+" ";i++;}
   return s;
   }

   public String toString()
   {
          String s=ciktiS()+" mesafe = "+mesafe();
   return s;
   }
}
```

Program 5.34.4 satici_genetik.java programı gezgin satıcı GA algoritmasını çözer. Başlangıç ve bitiş noktaları sabitlenmemiştir.

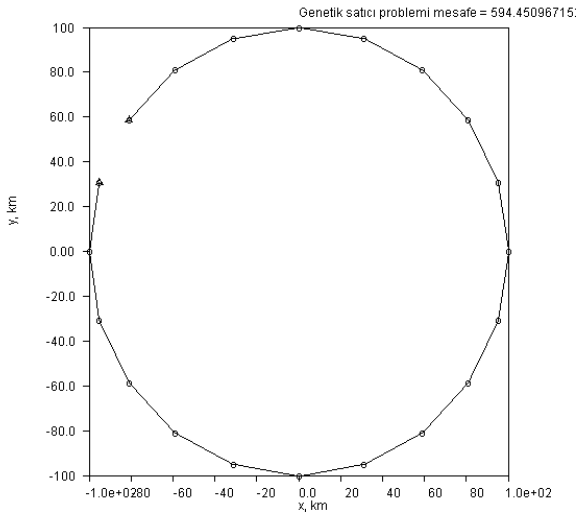
```
import java.io.*;
import java.text.*;
import java.util.Locale;
//
class satici_genetik1
{
public satici_gen1 G[];    // string of variables makes a genotype1
public boolean Breeder;   //
public int    POPSIZE;          // population size
int    MAXGENS;          // maximum number of generations;
double PMUTATION;        // proportion of mutated variables
double PXOVER;           // probability of crossover;
int    Generation;       // Current generation number
int    NBreeder;   //   Number of survivors in a generation
int    Best; //The best genotype1 in the population
double toplameniyi=0;
double nesileniyi=0;
double xi[],yi[]; //şehir adresleri

  public satici_genetik1(String name,int POPSİZEi,int MAXGENSi,double PXOVERi,double PMUTATIONi )
  {double a[][]=Text.readDoubleT(name);
   double xi[]=a[0];
   double yi[]=a[1];
   int best;
   POPSIZE=POPSİZEi;
   PXOVER=PXOVERi;
   MAXGENS=MAXGENSi;
   PMUTATION=PMUTATIONi;
```

```
  G=new satici_gen1[POPSIZE+1];
  double totalfitness=0;
  for(int i=0;i<(POPSIZE+1);i++)
  {G[i]=new satici_gen1(xi,yi);G[i].setFitness();totalfitness+=G[i].getFitness();}
  double xR,xC=0;
  for(int i=0;i<(POPSIZE+1);i++)
  {xR=G[i].getRFitness()/totalfitness;xC+=xR;G[i].setRFitness(xR);G[i].setCFitness(xC);}
 }

 public void setPopulation()
 {
  double totalfitness=0;
  for(int i=0;i<POPSIZE;i++)
  {G[i]=new satici_gen1(xi,yi);}
   evaluate();
 }

 public void evaluate()
 {
  nesileniyi=0;
  double totalfitness=0;
  for(int i=0;i<POPSIZE+1;i++)
  {G[i].setFitness();
   double mesafe=1.0/G[i].getFitness();
   if(mesafe>nesileniyi) nesileniyi=mesafe;
   if(mesafe>toplameniyi) toplameniyi=mesafe;
   totalfitness+=G[i].getFitness();
  }
  double xR=0,xC=0,xRP=G[POPSIZE].getFitness()/totalfitness;
  for(int i=0;i<POPSIZE;i++)
  {xR=G[i].getFitness()/totalfitness;xC+=xR;G[i].setRFitness(xR);G[i].setCFitness(xC);
            if(xR > xRP)
               {
                        Best=i;
                        G[POPSIZE]=new satici_gen1(G[i]);
                        xRP=G[POPSIZE].getFitness()/totalfitness;
       }
 }
}
 public satici_gen1[] copyPopulation()
 {
           satici_gen1 Po[]=new satici_gen1[POPSIZE+1];
           for(int i=0;i<(POPSIZE+1);i++)
             { Po[i]=new satici_gen1(G[i]);
               Po[i].Fitness=G[i].Fitness;
                       Po[i].RFitness=G[i].RFitness;
               Po[i].CFitness=G[i].CFitness;
             }
           return Po;
 }

           public void select()
```

```
        {
        double r,r1,r2,gf0,gf1;
        evaluate();
        //anne baba aday adaylarını seç
        satici_gen1 Po[]=new satici_gen1[POPSIZE+1];
        NBreeder=0;
        int i,j;
        for(i=0;i<(POPSIZE+1);i++)
{   r=Math.random();
   gf0=0;
   for(j=0;j<POPSIZE;j++)
   {
   gf1=G[j].getCFitness();
   if((r>=gf0) && (r<=gf1))
   {Po[NBreeder]=new satici_gen1(G[j]);
   //System.out.println("j= "+j+"gf0="+gf0+"gf1="+gf1+"r="+r+"NBreeder="+NBreeder);
   NBreeder++;break;}
   gf0=gf1;
   }
}
// seçilen anne, baba aday adaylarının yeni çocuk yapmalarına olanak ver
  int breed=(int)(POPSIZE*PXOVER);
  for(i=0;i<breed;i++)
          {
                  int a=(int)(Math.random()*NBreeder);
      int b=(int)(Math.random()*NBreeder);
      G[i]=new satici_gen1(Po[a],Po[b]);
      //System.out.println("a="+a+"b="+b+"cocuk="+G[i].toString());
      r2=Math.random();
      //mutasyon olasılığı : PMUTATION
            if(r2 < PMUTATION) G[i].mutate();
            //System.out.println(NBreeder+" "+G[POPSIZE].mesafe());
          }
         //listenin gerisini seçilen anne baba adaylarıyla doldur
            for(i=breed;i<POPSIZE;i++)
            {  int a=(int)(Math.random()*NBreeder);
       G[i]=new satici_gen1(Po[a]);
            }
}

public void nesiller()
{
        for(int i=0;i<MAXGENS;i++) {select();}
        }

public satici_gen1 secilmis_nesil()
{
nesiller();
G[POPSIZE].plot();
return G[POPSIZE];
        }
```

```
  public static void main(String arg[])
  {
    satici_genetik1 sg=new satici_genetik1("sehirler1.txt",1000,1000,1.0,0.03 );
    //sg.nesiller();
    //System.out.println("en iyi mesafe = "+sg.toplameniyi);
    System.out.println("seçilmiş nesil : "+sg.secilmis_nesil().toString());
  }

}
```

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" satici_genetik1
seçilmiş nesil : 9 (-95.10565162951535,30.901699437494752)
10 (-100.0,1.2246467991473532E-14)
11 (-95.10565162951538,-30.90169943749469)
12 (-80.90169943749474,-58.7785252292473)
13 (-58.77852522924732,-80.90169943749473)
14 (-30.901699437494756,-95.10565162951535)
15 (-1.8369701987210297E-14,-100.0)
16 (30.901699437494724,-95.10565162951536)
17 (58.77852522924729,-80.90169943749476)
18 (80.90169943749473,-58.778525229247336)
19 (95.10565162951535,-30.901699437494763)
0 (100.0,0.0)
1 (95.10565162951535,30.901699437494738)
2 (80.90169943749474,58.778525229247315)
3 (58.778525229247315,80.90169943749474)
4 (30.901699437494745,95.10565162951535)
5 (6.123233995736766E-15,100.0)
6 (-30.901699437494734,95.10565162951536)
7 (-58.7785252292473,80.90169943749474)
8 (-80.90169943749473,58.77852522924732)
 mesafe = 594.4509671528772
```



(bolum5_044.jpg,Resimaltı **:** Gezgin satıcı problemi GA satici_genetik çözüm 2 grafik çıktı t başlangıç ve bitiş şehirleri tesadüfi seçiliyor)

Gezgin satıcı problemi GA satici_genetik çözüm 2 grafik çıktı sabit başlangıç ve bitiş şehirleri

5.34.2 Gezgin satıcı problemi monte carlo algoritması çözümü

Monte karlo algoritmaları birçok problemde alternatif olarak önem taşır. Ancak Gezgin satıcı problemi için çok iyi bir alternatif oluşturmaz. Burada bu yöntem özellikle geçerli çözüme ulaşamadığımız göstermek amacıyla verilmiştir. Temel olarak genetik algoritmada kullandığımız satici_gen sınıfından yararlanılmıştır. Programımızda 100 milyon tesadüfi yol oluşturulmuş ve bu yollardan en kısası seçilmiştir.

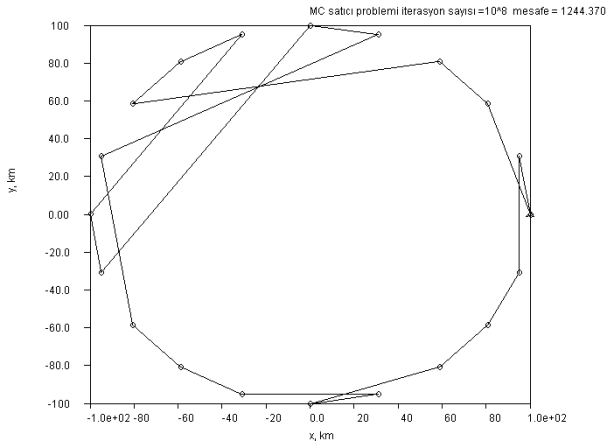
Program 5.34.5 satici_montecarlo.java programı gezgin satıcı montecarlo algoritmasını çözer. Başlangıç ve bitiş noktaları sabitlenmiştir.

```
import java.io.*;
import java.text.*;
import java.util.Locale;

class satici_montecarlo
{
public satici_gen G;    // string of variables makes a genotype1
public int    POPSIZE;          // population size
double xi[],yi[]; //şehir adresleri

  public satici_montecarlo(String name,int POPSİZEi)
  {double a[][]=Text.readDoubleT(name);
   double xi[]=a[0];
   double yi[]=a[1];
   POPSIZE=POPSİZEi;
   satici_gen G1=new satici_gen(xi,yi);
   double toplameniyi=1e20;
   double mesafe;
   int counter;
   for(int i=0;i<POPSIZE;i++)
   {G1=new satici_gen(xi,yi);
   mesafe=G1.mesafe();
   counter=i%1000000;
   if(counter==0) System.out.println("i="+i);
   if(mesafe < toplameniyi) {toplameniyi=mesafe;G=new
satici_gen(G1);System.out.println("i="+i+"\n"+G.toString());}
    }}

  public static void main(String arg[])
  {
    satici_montecarlo sg=new satici_montecarlo("sehirler.txt",100000000);
    //System.out.println("seçilmiş nesil : "+sg.G.toString());
    sg.G.plot();
  }

}
```

```
---------- Capture Output ----------
```

```
> "D:\java\bin\javaw.exe" satici_montecarlo
i=0
  (100.0,0.0)
5 (-30.901699437494734,95.10565162951536)
0 (95.10565162951535,30.901699437494738)
18 (95.10565162951535,-30.901699437494763)
17 (80.90169943749473,-58.778525229247336)
4 (6.123233995736766E-15,100.0)
2 (58.778525229247315,80.90169943749474)
11 (-80.90169943749474,-58.7785252292473)
14 (-1.8369701987210297E-14,-100.0)
1 (80.90169943749474,58.778525229247315)
3 (30.901699437494745,95.10565162951535)
12 (-58.77852522924732,-80.90169943749473)
9 (-100.0,1.2246467991473532E-14)
16 (58.77852522924729,-80.90169943749476)
13 (-30.901699437494756,-95.10565162951535)
15 (30.901699437494724,-95.10565162951536)
8 (-95.10565162951535,30.901699437494752)
6 (-58.7785252292473,80.90169943749474)
7 (-80.90169943749473,58.77852522924732)
10 (-95.10565162951538,-30.90169943749469)
  (100.0,0.0)
 mesafe = 2343.4261590027527
i=2788
  (100.0,0.0)
0 (95.10565162951535,30.901699437494738)
6 (-58.7785252292473,80.90169943749474)
7 (-80.90169943749473,58.77852522924732)
5 (-30.901699437494734,95.10565162951536)
4 (6.123233995736766E-15,100.0)
8 (-95.10565162951535,30.901699437494752)
15 (30.901699437494724,-95.10565162951536)
11 (-80.90169943749474,-58.7785252292473)
12 (-58.77852522924732,-80.90169943749473)
16 (58.77852522924729,-80.90169943749476)
18 (95.10565162951535,-30.901699437494763)
14 (-1.8369701987210297E-14,-100.0)
10 (-95.10565162951538,-30.90169943749469)
13 (-30.901699437494756,-95.10565162951535)
9 (-100.0,1.2246467991473532E-14)
17 (80.90169943749473,-58.778525229247336)
3 (30.901699437494745,95.10565162951535)
2 (58.778525229247315,80.90169943749474)
1 (80.90169943749474,58.778525229247315)
  (100.0,0.0)
 mesafe = 1861.2913632188056
i=8810541
  (100.0,0.0)
1 (80.90169943749474,58.778525229247315)
18 (95.10565162951535,-30.901699437494763)
16 (58.77852522924729,-80.90169943749476)
```

4 (6.123233995736766E-15,100.0)
5 (-30.901699437494734,95.10565162951536)
7 (-80.90169943749473,58.77852522924732)
8 (-95.10565162951535,30.901699437494752)
10 (-95.10565162951538,-30.90169943749469)
11 (-80.90169943749474,-58.7785252292473)
9 (-100.0,1.2246467991473532E-14)
6 (-58.7785252292473,80.90169943749474)
3 (30.901699437494745,95.10565162951535)
2 (58.778525229247315,80.90169943749474)
0 (95.10565162951535,30.901699437494738)
13 (-30.901699437494756,-95.10565162951535)
12 (-58.77852522924732,-80.90169943749473)
14 (-1.8369701987210297E-14,-100.0)
15 (30.901699437494724,-95.10565162951536)
17 (80.90169943749473,-58.778525229247336)
 (100.0,0.0)
 mesafe = 1384.7588558636148
i=51556228
 (100.0,0.0)
0 (95.10565162951535,30.901699437494738)
18 (95.10565162951535,-30.901699437494763)
17 (80.90169943749473,-58.778525229247336)
16 (58.77852522924729,-80.90169943749476)
14 (-1.8369701987210297E-14,-100.0)
15 (30.901699437494724,-95.10565162951536)
13 (-30.901699437494756,-95.10565162951535)
12 (-58.77852522924732,-80.90169943749473)
11 (-80.90169943749474,-58.7785252292473)
8 (-95.10565162951535,30.901699437494752)
3 (30.901699437494745,95.10565162951535)
4 (6.123233995736766E-15,100.0)
10 (-95.10565162951538,-30.90169943749469)
9 (-100.0,1.2246467991473532E-14)
5 (-30.901699437494734,95.10565162951536)
6 (-58.7785252292473,80.90169943749474)
7 (-80.90169943749473,58.77852522924732)
2 (58.778525229247315,80.90169943749474)
1 (80.90169943749474,58.778525229247315)
 (100.0,0.0)
 mesafe = 1244.370680344478



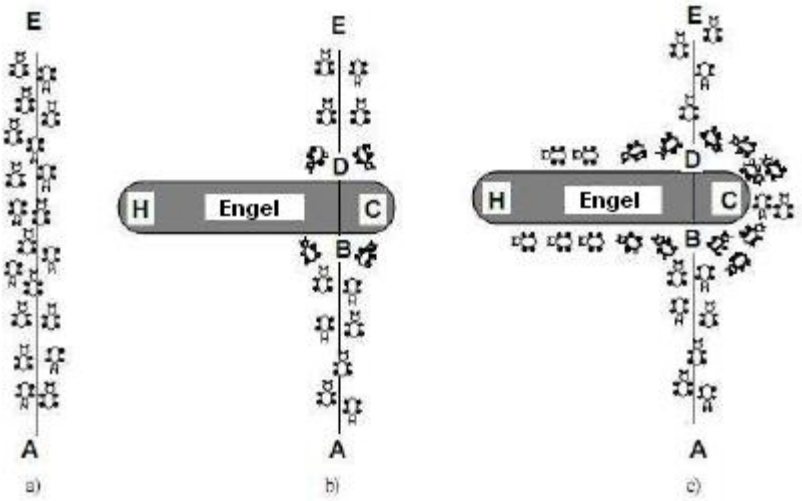
(bolum5_045.jpg,Resimaltı **:**Monte-Carlo çözümü 1000000 iterasyon)

5.34.3 Gezgin satıcı problemi, karınca algoritması çözümü

Satıcı problemi olarak bilinen n nokta arasındaki en kısa yolu (tüm n noktaya uğramak şartı ile) bulma problemi günlük hayatta sıkça karşımıza çıkabilir. Günümüzdeki en yaygın kullanım alanları, telefon internet gibi elektronik bağlantıları en kısa yol üzerinden yapma eylemidir. Burada bu problemin özel bir çözümü üzerinde duracağız. Karınca simulasyonu.

Karınca simulasyon yöntemi, yakın zamanda geliştirilmiş olan (90 ların sonları); bir tür stokastik (tesadüfi sayı sistemleri kullanan) kombinasyonal optimizasyon yöntemidir. Özellikle; TSP – Travelling Salesman Program – , MST – Minimum Spanning Tree – , iş programı yapma problemleri gibi konularda uygulama alanı bulmuştur. Temel özellikleri; pozitif geribesleme *(positive feedback)*, dağıtılmış hesaplama *(distributed computing)*, ve üretken açgözlü sezgiselliğin *(constructive greedy heuristic)* kullanımıdır.
Pozitif geribesleme, uygun sonuçların çabuk bulunmasında; dağıtılmış hesaplama, erken (kusurlu) yakınsamanın engellenmesinde; açgözlü sezgisellik ise, arama işleminin ilk kademelerinde kabul edilebilir sonuçlar bulunmasında etkilidir.

Bu simulasyonun temeli gerçek dünyadan, karıncaların yiyeceğe giden yolu bulma prensibinden ortaya çıkmıştır. Karıncalar yuvalarından yiyecek buldukları yere giderken feremon denen koku salan bir kimyasal madde yayarlar. Diğer karıncalar bu kokuyu algılayabilir. Kokunun varlığı ve şiddeti, diğer karıncaların yol tercihinde etkili olur. Ancak kimyasal madde zamanla uçtuğundan koku zaman içinde yok olur. Bu şekilde karıncaların takipettiği bir kokuizi oluşur. Bu şekilde yenigelen karıncalar yiyecek kaynağına daha kolaylıkla bu koku yolunu izleyerek ulaşabilirler. Her geçen karınca da yeni feromen bırakacağından çok kullanılan yolun kokuyoğunluğu artarken kullanılmayan yollarda olan feromen de uçacağından koku yok olur. Eğer bir yol daha kısa ise o yol üzerinde biriken feromen miktarı da karıncalar yuvaya daha hızlı dönebildikleri de hesaba katılırsa daha yoğun olacaktır.



(bolum5_046.jpg,Resimaltı **:** Karıncaların bir engel karşısındaki davranışı -yol seçimi)

Şimdi bu prensibi bilgisayar algoritmasına nasıl çevirebileceğimizi irdeleyelim : Karıncaların 1 ve 2 gibi iki değişik yoldan yiyecek kaynağına ulaştıklarını varsayalım. Bu durumda t zamanında (m+1) inci karıncanın birinci yolu seçme olasılığı :

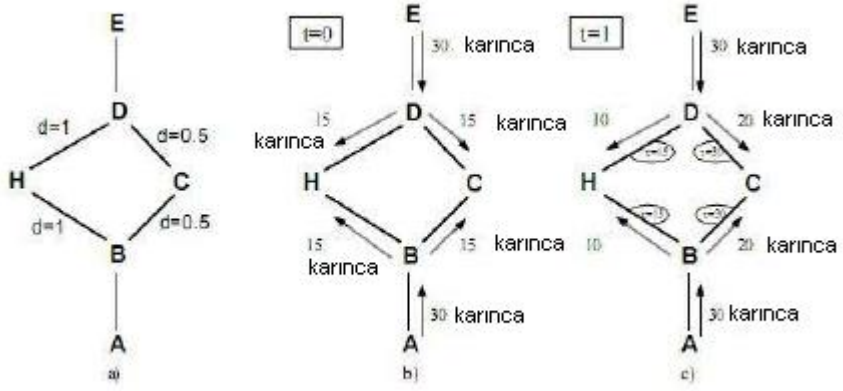
$$p_{1(m+1)} = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h}$$

olarak yazılabilir. Buradaki $m_1$ve m_2 birinci yolu kullanan ve ikinci yolu kullanan karınca sayısıdır.k ve h sabitlerdir. (m+1) inci karıncanın ikinci yolu seçme olasılığı da :

$P_{2(m+1)}= 1 - P_{1(m+1}$

Şeklinde yazılabilir. Bilgisayar simulasyonunda kullanılan *ajan karıncalar*, gerçek hayattaki karıncalara göre belirli farklılıklar gösterirler. Bunlar;

1. Ajan karıncaların hafızası vardır. Örneğin; *gezgin satıcı probleminde* daha önce gittikleri yerleri bir Tabu listesinde saklarlar, hatırlarlar.
2. Tamamiyle kör değillerdir. Örneğin; seçtikleri yolu, uzunluğuna göre de irdelerler.
3. Zamanın ayrık olduğu bir çevrede yaşarlar.



(bolum5_047.jpg,Resimaltı **:** Karıncaların iki yol problemlerindeki yol seçiminin zaman içindeki dağılımı)

- t = 0 anında; E'den A'ya doğru, A'dan da E'ye doğru 30'ar karınca yola çıkmaktadır.
- Henüz yollarda feramon birikmesi olmadığı için, karıncalar eşit olasılıkta B ve D şehirlerinden H ve C şehirlerine doğru yönleceklerdir. Bu durumda 15'er li iki grup oluşacaktır.
- B – C – D yolu daha kısa olduğu için, burada daha fazla feramon birikecektir.

- t = 1 anında; B – C – D yolunda daha fazla feramon biriktiği için, karıncalar ağırlıklı olarak bu güzergahı tercih edecektir.

Karınca Kolonilerinin Gezgin Satıcı Programine Uygulaması

Bilindiği üzere; gezgin satıcı problemi, verilen n tane şehirden oluşan kümedeki her şehire bir kez uğrayarak oluşturulan en kısa yolun bulunmasını kapsar.

Tanımlar :

Şehirlerin bulunduğu uzayı, Öklid uzayı olarak kabul edersek;

n : Şehir sayısı
d_{ij} : i ile j numaraları birbirine birleştiren Öklid mesafesi
$b_i(t)$: t anındaki, i şehirdeki karınca sayısı
m : Toplam karınca sayısı

$$m = \sum_{i=1}^{n} b_i(t)$$

$\tau_{ij}(t)$: t anında; ij yolundaki izin yoğunluğu

ρ : Buharlaşma oranı; sonsuz feramon birikmesini ve yerel minimalara takılmayı engellemek için1 den küçük olmalıdır.

$\Delta\tau_{ij}^{k}$: ij yolunda, k'ıncı karınca tarafından bırakılmış izin, yol uzunluğuna oranı

$\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$ Eğer k ıncı karınca bu yolu kullandıysa

$\Delta\tau_{ij}^{k} = 0$ Eğer k ıncı karınca bu yolu kullanmadıysa

L_k *;* k ıncı karıncanın toplam katettiği yol
Q; bırakılan **sabit** feramon miktarı

$Tabu_k$: k ıncı karıncanın gittiği şehirlerin listesi
η_{ij} : Görünürlük

$$\eta_{ij} = \frac{1}{d_{ij}}$$

$p_{ij}^{k}(t)$: i inci şehirden j inci şehire geçiş olasılığı

Eğer j şehri tabu listesinde yoksa;

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}{\sum [\tau_{ik}(t)]^{\alpha} \times [\eta_{ik}]^{\beta}}$$

α ve β; görünürlük ile feramon miktarı arasındaki önemi kontrol eden değişkenlerdir

Eğer j şehri tabu listesinde var ise;

$$p_{ij}^{k}(t) = 0$$

Algoritma :

1. Başlangıç :
   a. t = 0 — t : zaman sayacı
   b. NC = 0 — NC : çevrim sayacı

c. Her yol için; başlangıç feramon değerlerini belirler;
d. m tane karıncayı n tane düğüme yerleştir

2. s = 0 s : tabu listesi sayacı
   k = 1 den m ye kadar;
   k ıncı karınca için $tabu_k(s)$ listesinde ilk şehirleri yerleştir.
3. Liste dolana kadar tekrarla;
   a. s = s + 1
   b. k = 1 den m ye kadar;
      i. j inci gidilecek şehri; $p_{ij}^{k}(t)$ olasılığına bağlı olarak seç
      ii. k ıncı karıncayı j inci şehre götür
      iii. j inci şehri tabu listesine ekle
4. k = 1 den m ye kadar
   a. k ıncı karıncayı $tabu_k(n)$ den $tabu_k(1)$ e götür
   b. k ıncı karınca tarafından katedilen mesafeyi hesapla; L_k
   c. bulunan en kısa mesafeyi güncelle
   d. her ij yolu için;
      i. k = 1 den m ye kadar

      Eğer k ıncı karınca bu yolu kullandıysa $\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$

      Eğer k ıncı karınca bu yolu kullanmadıysa $\Delta\tau_{ij}^{k} = 0$

      $$\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^{k}$$

5. Her ij yolu için; $\tau_{ij}(t+n)$ değerini hesapla;
   a. $\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij}$
   b. t = t + n
   c. NC = NC + 1
   d. Her ij yolu için $\Delta\tau_{ij} = 0$
6.
   a. Eğer $NC < NC_{maks}$; bütün tabu listelerini temizle ve 2. maddeye git;
   b. Değilse; en kısa mesafeyi yaz ve dur.
      (tıkanma davranışı / stagnation behaviour)

Literatürde; farklı karınca kolonileri algoritmaları mevcuttur. Bazıları sadece feramon bırakılış şekline alternatif getirirken bazıları koloni şeklini değiştirmektedir.

Bunlardan bazıları :

1. Karınca Döngüsü *(Ant Cycle)* : Bu bir önceki sayfada açıklanmış algoritmadır.

2. Karınca Yoğunluğu *(Ant density)* : Bu metodaki farklılık; feramonlar güncellenirken, karıncanın toplam katettiği yola bakılmaz.

   Eğer k ıncı karınca bu yolu kullandıysa $\Delta\tau_{ij}^{k} = Q$

   Eğer k ıncı karınca bu yolu kullanmadıysa $\Delta\tau_{ij}^{k} = 0$

3. Karınca Sayı *(Ant Quantity)* : Bu metodda ise; feramonlar güncellenirken, karıncanın toplam katettiği yol yerine, ij şehirleri arasındaki mesafe kullanılır.

Eğer k ıncı karınca bu yolu kullandıysa $\Delta\tau_{ij}^{k} = \frac{Q}{d_{ij}}$

Eğer k ıncı karınca bu yolu kullanmadıysa $\Delta\tau_{ij}^{k} = 0$

Bu iki yöntem arasındaki fark, birinde karıncanın bıraktığı feramon miktarındaki artış, şehirler arasındaki mesafeyle ters orantılı iken; diğerinde mesafeden bağımsızdır.

4. Ardışık Karınca Kolonisi *(Sequential Ant Colony Optimization)* : Bu metodda; sadece tek karınca bütün düğümleri, bir olasılıksal geçiş kuralına göre gezer[2].

5. Elitist Karınca Kolonisi *(Elitist Strategy)* : Bu yöntem genetik algoritmalarındaki, elitist stratejiye benzer. En iyi yola, o yola doğru gidişi arttırmak açısından e*Q/L miktarında feramon eklenir. Burada; e, en iyi karınca sayısı; L ise en kısa tur mesafesidir.

Satıcı problemi için karınca optimizasyonu Program 5.29-1 de verilmiştir. Örnek test veri girdisi olarak bir çemberin üzerine dizilmiş noktalar aldık. Girdi x_i,y_i koordinat verileri dosya sehirler.txt de verilmiştir. Sonuçta şehirlerin girdi vektörü sırasına göre sıralaması verilmektedir.

Program 5.34.6 Satıcı problemi çözüm yöntemi olarak karınca optimizasyonu

```
import java.io.*;
import java.math.*;

class harita { double[][] sehirler;
            public double mesafe(double x1,double x2,double y1,double y2)
            { double mesafe;
              mesafe=Math.pow((Math.pow((x2-x1),2)+Math.pow((y2-y1),2)),0.5);
              return mesafe;
            }
            public harita(sehiroku x) {
            sehirler=new double[x.sehirsayisi][x.sehirsayisi+3];
            for(int i=0;i<x.sehirsayisi;i++) {
            sehirler[i][0]=i; //?ehrin ID si
            sehirler[i][1]=x.sehirler[i][0]; //?ehrin x koordinat?
            sehirler[i][2]=x.sehirler[i][1]; //?ehrin y koordinat?
            for(int k=3;k<(x.sehirsayisi+3);k++)
            {sehirler[i][k]=mesafe(x.sehirler[i][0],x.sehirler[k-3][0],x.sehirler[i][1],x.sehirler[k-3][1]);}}
                                }
          }

class sehiroku
{   String dosyaismi=new String(" ");
    double[][] sehirler;
    int sehirsayisi;
    public sehiroku(String x)
    {   dosyaismi=x;
        sehirler=Text.readDouble(x);
        sehirsayisi=sehirler.length;
    }
}

public class ant{
              double minmesafe;
```

```
    public static void quicksort(double[][] x,int y,int z){
    double[] mid=new double[3];
    double[] temp=new double[3];
    int k,j;
    k=y;
    j=z;
    mid[0]=x[(y+z)/2][0];
    mid[1]=x[(y+z)/2][1];
    mid[2]=x[(y+z)/2][2];
    do {while(x[k][2] < mid[2] & k < z) {k++;}
        while(x[j][2] > mid[2] & j > y) {j--;}
       if(k<=j){
          temp[2]=x[k][2];
          temp[1]=x[k][1];
          temp[0]=x[k][0];
          x[k][2]=x[j][2];
          x[k][1]=x[j][1];
          x[k][0]=x[j][0];
          x[j][2]=temp[2];
          x[j][1]=temp[1];
          x[j][0]=temp[0];
          k++;
          j--;
       }

    }while(k<=j);
    if(y<j) {
       quicksort(x,y,j);
    }
    if(k<z) {quicksort(x,k,z);}
    }

  public double readMesafe()
  {return minmesafe;}

        public  int[] ant_optimization(String dosya,double vaporizarion,double feramoni,int nkarinca )
        {   double tolerans=1.0e-5;
    double dx=0.1;
    //vaporization 0-1
    double buharlasmaorani=vaporizarion; //feramonun buharlasma oranı
    //feramoni=5 (örneğin)
    double feramon=feramoni; //feramon
    double beta=3; //
    double alfa=1; //
    double baslangicferamonu=1; //t=0 anindaki yoldaki feramon miktar?
    //int sakkarinca=10; //iterasyon aras? korunacak maksimum kar?nca say?s?
    //int sakis; //saklanacak karınca işareti
    sehiroku sehir=new sehiroku(dosya); //şehirlerin okunması
    harita bolge=new harita(sehir); //haritanın hazırlanması
    double[][] feramondurumu=new double[sehir.sehirsayisi][sehir.sehirsayisi]; //yolların feramon durumunun
kaydedileceği
```

```
double[][] deltaferamon=new double[sehir.sehirsayisi][sehir.sehirsayisi];
//int karincasayisi=1000; //koloni büyüklüğü
int karincasayisi=nkarinca;
int it=0;
int maxit=10; //maksimum iterasyon sayisi
int ilksehir;
int yolsayisi;
double toplamolasilik=0;
double[][] olasiliklar=new double[sehir.sehirsayisi][sehir.sehirsayisi];
double minolasilik=0;
double maxolasilik=0;
double tolasilik=0;
double mesafe;
minmesafe=0;
double minmesafeeski=0.1;
dx=Math.abs(minmesafe-minmesafeeski);
int minid=0;
int[] minguzergah=new int[sehir.sehirsayisi];
karincaorj[] karincaordusu=new karincaorj[karincasayisi];
//karincaorj[] sakkar=new karincaorj[sakkarinca];
yolsayisi=0;
//yolsayisinin hesaplanması
while (it<maxit  &&  dx>tolerans){
 minmesafeeski=minmesafe;
 yolsayisi=0;
for(int i=1;i<sehir.sehirsayisi-1;i++){
   yolsayisi=yolsayisi+sehir.sehirsayisi-i;
}
yolsayisi=yolsayisi+1;
//System.out.println(yolsayisi);
double[][] olasilikpasta=new double[yolsayisi][3];

//t=0 anındaki feramonun yollara konulması
if(it==0){
for(int i=0;i<(sehir.sehirsayisi-1);i++) {
   for(int j=i+1;j<(sehir.sehirsayisi);j++) {
      if(i==j){
         feramondurumu[i][j]=0;
      }else{
         feramondurumu[i][j]=baslangicferamonu;
      }
            }
 }
}
 //olasilik pastasinin hesaplanmasi
   //toplam olasiligin hesaplanmasi
for(int i=0;i<sehir.sehirsayisi-1;i++){
   for(int j=i+1;j<(sehir.sehirsayisi);j++){
      if(i!=j){
         toplamolasilik=Math.pow((feramondurumu[i][j]),alfa)*Math.pow(bolge.sehirler[i][j+3],-beta);
                  }
   }
```

```
        }
        //toplamolasilik=toplamolasilik/2;
           //pastanin hesaplanması
        for(int i=0;i<sehir.sehirsayisi-1;i++){
           for(int j=i+1;j<(sehir.sehirsayisi);j++){
              if(i!=j){
                 olasiliklar[i][j]=(Math.pow((feramondurumu[i][j]),alfa)*Math.pow(bolge.sehirler[i][j+3],-
beta))/toplamolasilik;
                 if(bolge.sehirler[i][j+3]==0){
                 }
                 if(i==0 & j==1){
                    maxolasilik=olasiliklar[i][j];
                    minolasilik=olasiliklar[i][j];
                    olasilikpasta[0][2]=minolasilik;
                    olasilikpasta[0][0]=i;
                    olasilikpasta[0][1]=j;
                    olasilikpasta[yolsayisi-1][2]=maxolasilik;
                    olasilikpasta[yolsayisi-1][0]=i;
                    olasilikpasta[yolsayisi-1][1]=j;
                 } else {
                    if(maxolasilik<olasiliklar[i][j]) {
                       maxolasilik=olasiliklar[i][j];
                       olasilikpasta[yolsayisi-1][2]=maxolasilik;
                       olasilikpasta[yolsayisi-1][0]=i;
                       olasilikpasta[yolsayisi-1][1]=j;
                    } else {
                    if(minolasilik>olasiliklar[i][j]){
                       minolasilik=olasiliklar[i][j];
                       olasilikpasta[0][0]=i;
                       olasilikpasta[0][1]=j;
                    }
                    }
                 }
                           }
              else {
                 olasiliklar[i][j]=0;
              }
           }
        }
        for(int i=1;i<yolsayisi-1;i++){
           olasilikpasta[i][2]=minolasilik;
        }
           //pastanin siralanmasi
        int l=0;
         for(int i=0;i<sehir.sehirsayisi-1;i++){
           for(int j=i+1;j<(sehir.sehirsayisi);j++){
             if(i!=j){
              olasilikpasta[l][2]=olasiliklar[i][j];
              olasilikpasta[l][0]=i;
              olasilikpasta[l][1]=j;
             l++;
              }
```

```
        }
      }
      //olasiliklari siralama
      quicksort(olasilikpasta,0,yolsayisi-1);
      //t=0 anında ilk karincaların yerleştirilmesi
      for(int i=0;i<karincasayisi;i++){
         double x;
         karincaordusu[i]=new karincaorj(i,sehir.sehirsayisi,feramon);
         x=Math.random()*sehir.sehirsayisi;
         ilksehir=(int) x;
         karincaordusu[i].gyol[0]=ilksehir;
      }
      //Karincaların yürümesi
      for(int i=1;i<sehir.sehirsayisi;i++){
         for(int j=0;j<karincasayisi;j++){
            karincaordusu[j].yuru(bolge,olasilikpasta,i,yolsayisi);
         }
      }
      minmesafe=karincaordusu[0].kmesafe;
      minguzergah=karincaordusu[0].gyol;
      for(int j=0;j<karincasayisi;j++){
         if(karincaordusu[j].kmesafe<minmesafe){
            minmesafe=karincaordusu[j].kmesafe;
            minguzergah=karincaordusu[j].gyol;
         }
      }

      for(int j=0;j<karincasayisi;j++){
         karincaordusu[j].ferguncel(deltaferamon);
      }
       for(int i=0;i<(sehir.sehirsayisi-1);i++) {
         for(int j=i+1;j<(sehir.sehirsayisi);j++) {
            if(i==j){
               feramondurumu[i][j]=0;
            }else{
               feramondurumu[i][j]=feramondurumu[i][j]*buharlasmaorani+deltaferamon[i][j];
            }
                  }
      }
         it++;
         dx=Math.abs(minmesafe-minmesafeeski);
      }
      return minguzergah;
      }
}
```

Program 5.34.7 Satıcı problemi çözüm yöntemi olarak karınca optimizasyonu, test programı

```
public class anttest
{
       public static void main(String args[])
      {
      double buharlasmaorani=0.90; //feramonun buharlasma oran?
```

```
        double feramon=5; //feramon
        int toplamkarinca=1000;
        ant x=new ant();
        int[] tur=x.ant_optimization("sehirler.txt",buharlasmaorani,feramon,toplamkarinca);
        for(int i=0;i<tur.length;i++){System.out.print(tur[i]+" ");}
        System.out.println("\nminimum mesafe = "+x.readMesafe());
        }
}
```

Tablo 5.34.1 Satıcı problemi çözüm yöntemi olarak karınca optimizasyonu, test verisi

100	0
95.10565163	30.90169944
80.90169944	58.77852523
58.77852523	80.90169944
30.90169944	95.10565163
6.12E-15	100
-30.90169944	95.10565163
-58.77852523	80.90169944
-80.90169944	58.77852523
-95.10565163	30.90169944
-100	1.22E-14
-95.10565163	-30.90169944
-80.90169944	-58.77852523
-58.77852523	-80.90169944
-30.90169944	-95.10565163
-1.84E-14	-100
30.90169944	-95.10565163
58.77852523	-80.90169944
80.90169944	-58.77852523
95.10565163	-30.90169944
100	-2.45E-14

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" anttest
2 1 0 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3
minimum mesafe = 594.4540062839565

> Terminated with exit code 0.
```

5.35 Lineer optimizasyon, simpleks metodu

Optimizasyonun son konusu olarak lineer optimizasyona göz atacağız. Nispeten kolay olan lineer optimizasyon metodları, bir çok lineer optimizasyon problemiyle karşılaştığımız için önemlidir. Lineer problemler tanımı icabı sadece sınırlar ile beraber anlam taşırlar sınırsız lineer optimizasyon tanımlı değildir.
Lineer optimizasyon problemini en genel olarak

Maksimum $Z=a_0x_0+ a_1x_1+ a_2x_2+ a_3x_3+\ldots\ldots+ a_nx_n$
Sınır şartları :
$b_{00}x_0+ b_{01}x_0+ b_{02}x_0+ b_{03}x_0+\ldots\ldots+ b_{0n}x_0 <= A_0$
$b_{10}x_0+ b_{11}x_0+ b_{12}x_0+ b_{13}x_0+\ldots\ldots+ b_{1n}x_0 <= A_1$
$b_{20}x_0+ b_{21}x_0+ b_{22}x_0+ b_{23}x_0+\ldots\ldots+ b_{2n}x_0 <= A_2$

.....

$b_{n0}x_0 + b_{n,1}x_0 + b_{n2}x_0 + b_{n3}x_0 + \ldots\ldots + b_{nn}x_0 <= A_n$

şeklinde tanımlıyabiliriz. Sistemde maksimum yerine minimum olduğunda veya <= yerine >= gibi değişik şartların gelmesini işaret değiştirerek çözebiliriz. Bir örnek yazarsak problem :

Maksimum Z=60x+50y
Sınır şartları :
2x+4y<=80
3x+2y<=60

şeklinde verilebilir.

Denklemi simpleks metodu ile çözmek için önce formunda değişiklik yapmak gerekir. denklem

$Z - a_0x_0 - a_1x_1 - a_2x_2 - a_3x_3 - \ldots\ldots - a_nx_n = 0$
Sınır şartları :
$b_{00}x_0 + b_{01}x_1 + b_{02}x_2 + b_{03}x_3 + \ldots\ldots + b_{0n}x_n + W_0 = A_0$
$b_{10}x_0 + b_{11}x_1 + b_{12}x_2 + b_{13}x_3 + \ldots\ldots + b_{1n}x_n + W_1 = A_1$
$b_{20}x_0 + b_{21}x_1 + b_{22}x_2 + b_{23}x_3 + \ldots\ldots + b_{2n}x_n + W_2 = A_2$

.....

$b_{n0}x_0 + b_{n,1}x_1 + b_{n2}x_2 + b_{n3}x_3 + \ldots\ldots + b_{nn}x_n + W_n = A_n$

şekline getirilir. Denklem matris formunda yazılırsa

$$\begin{bmatrix} -a_0 & -a_1 & -a_2 & .... & -a_n & 0 & 0 & ... & 0 \\ b_{00} & b_{01} & b_{02} & .... & b_{0n} & 1 & 0 & ... & 0 \\ b_{10} & b_{11} & b_{12} & .... & b_{1n} & 0 & 1 & ... & 0 \\ ... & ... & ... & .... & ... & ... & ... & ... & ... \\ b_{n0} & b_{n1} & b_{n2} & .... & b_{nn} & 0 & 0 & ... & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ .... \\ x_n \\ W_0 \\ W_1 \\ W_2 \\ .... \\ W_n \end{bmatrix} = \begin{bmatrix} 0 \\ A_0 \\ A_1 \\ .... \\ A_n \end{bmatrix}$$

şeklini alır. Görüldüğü gibi matrisin satır ve sütun sayıları uymadığından direk bir denklem çözme sistemi ie çözülmesi mümkün değildir. Bunun yerine Simpleks metodu simi verilen bir metod kullanacağız. Bu metodun uygulanması temel olarak şu şekilde yapılır :

1. Ana denklemde katsayıları negatif olarak en büyük olan sütun seçilir. Bu sütuna kritik sütun adı verilir.
2. Şart denklemlerinin ikinci tarafı kritik sütundaki değere bölünerek değerleri bulunur. Değeri en küçük olan sütun kritik sütun olarak alınır.
3. kritik sütunla kritik satırın kesim noktasına kritik eleman adı verilir.
4. kritik satır, kritik elemana bölünerek normalize edilir
5. kritik satır dışındaki satırlar
   yeni değer = eski değer – [kritik sütundaki aynı satıra gelen değer]*[kritik satırdaki aynı sütuna gelen değer]/kritik eleman formülü kullanılarak hesaplanır.
6. iteratif olarak 1 den 5 e kadar olan stepler tüm ana denklemdeki tüm – değerler 0 veya positif olana kadar tekrarlanır.

Şimdi örnek problem üzerinde bu işlemleri yineleyelim :
Maksimum Z=60x+50y
Sınır şartları :

2x+4y<=80
3x+2y<=60
şeklinde verilen örnek problem simpleks formuna getirilirse :
$Z-60x-50y+0W_A+0W_B = 0$
$2x+4y + W_A+0W_B= 80$
$3x+2y +0W_A+ W_B=60$
matrix formunda :

$$\begin{bmatrix} -60 & -50 & 0 & 0 & 0 \\ 2 & 4 & 1 & 0 & 80 \\ 3 & 2 & 0 & 1 & 60 \end{bmatrix}$$

şeklinde tanımlanabilir. 1. iterasyonda anahtar sütunun -60 ın yer aldığı sütun olduğu görülür. Anahtar satırı bulmak için t1=80/2=40 t2=60/3=20 karşılaştırılır, 20 daha küçük olduğundan 3 satır anahtar satır olarak seçilir. 3 satır anahtar eleman olarak bulunan 3 değerine bölünerek

$1 \quad \frac{2}{3} \quad 0 \quad \frac{1}{3} \quad 20$ şekline dönüştürülür kritik satır dışındaki satırlarda kritik sütundaki değerler (x katsayıları) elimine edilecek şekilde kritik kolon sabit bir sayıyla çarpılarak işlem yapılacak değerle toplanır. Örneğin 0 ıncı sütundaki x değerini 0 yapmak için 3 satır(kritik sayır) 60 ile çarpılarak o satırla toplanır. Bu durumda değerleri

$0 \quad -10 \quad 0 \quad 20 \quad 1200$ olacaktır.

birinci sütundaki x değerini 0 yapmak için 3 satır(kritik sayır) -2 ile çarpılarak 1. satırla toplanır. Bu durumda değerleri

$0 \quad \frac{8}{3} \quad 1 \quad -\frac{2}{3} \quad 40$

olur bu durumda matris

$$\begin{bmatrix} 0 & -10 & 0 & 20 & 1200 \\ 0 & \frac{8}{3} & 1 & -\frac{2}{3} & 40 \\ 1 & \frac{2}{3} & 0 & \frac{1}{3} & 20 \end{bmatrix}$$

formunu alır. Aynı iteratif işlemi bir daha tekrarlarız. Bu sefer anahtar sütun 1 anahtar satır 1 olacaktır. Anahtar eleman 8/3 tür. Elimine prosesi sonucunda matris

$$\begin{bmatrix} 0 & 0 & \frac{15}{4} & \frac{35}{2} & 1350 \\ 0 & 1 & \frac{3}{8} & -\frac{1}{4} & 15 \\ 1 & 0 & -\frac{1}{4} & \frac{1}{2} & 10 \end{bmatrix}$$

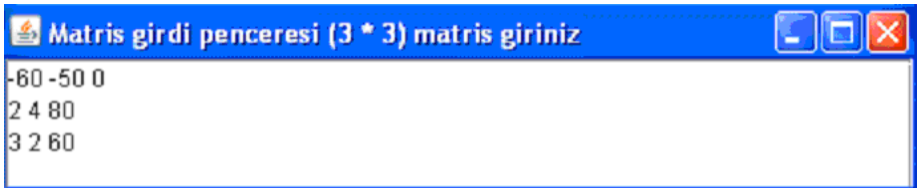
formunu alır. Bu forma geldikten sonra W_A ve W_B değerleri yerine 0 koyulursa x=10 y=15 ve fonksiyon değeri 1350 olarak bulunur. Şimdi aynı örnek problemi bilgisayar ortamında çözersek :

Program 5.35.1 Lineer simpleks optimizasyonu

```
class simplex
{
    public static double[] simplex(double[][] a)
    {
    int n=a.length; //satır
    int m=a[0].length; //sütun
    System.out.println("m="+m+"n="+n);
    int m1=m-1;//x,y lerin sayısı
    int n1=n-1;
    int m2=m1+n1;
```

```
int m3=m2+1;
int keyrow=0;
int keycolumn=0;
double keynumber=0.0;
double ratio=1.0;
double b[][]=new double[n][m3];
double d[]=new double[m];
String s[]=new String[m];
for(int i=0;i<n;i++)
{ b[i][m2]=a[i][m1]; }
for(int i=0;i<n-1;i++)
  {
          for(int j=0;j<m1;j++) {b[i][j]=a[i][j];}
  for(int j=m1;j<m2;j++) {if(i==(j-m1)) b[i+1][j]=1.0;System.out.println("i="+i+"j="+j+"bij="+b[i+1][j]);}
  }
double max;
double min;
System.out.println(Matrix.toString(b));
for(int k=0;k<n1;k++)
{
        max=-9.999e30;
min=9.999e30;
for(int j=0;j<m1;j++)
{if( (b[0][j]<0) && Math.abs(b[0][j])>max) {max=Math.abs(b[0][j]);keycolumn=j;}}
for(int i=1;i<n;i++)
{ ratio=b[i][m2]/b[i][keycolumn];
  if(ratio<min)  {min=ratio;keyrow=i;}
}
keynumber=b[keyrow][keycolumn];
double c[][]=new double[n][m3];
for(int i=0;i<n;i++)
{ for(int j=0;j<m3;j++) {c[i][j]=b[i][j];}}
for(int i=0;i<n;i++)
{ if(i!=keyrow) {for(int j=0;j<m3;j++) {b[i][j]=c[i][j]-c[keyrow][j]*c[i][keycolumn]/keynumber;}}
  else {for(int j=0;j<m3;j++) {b[i][j]=c[i][j]/keynumber;}}
}
System.out.println("keynumber = "+keynumber);
System.out.println("keyrow = "+keyrow+" keycolumn = "+keycolumn);
System.out.println(Matrix.toString(b));
}

d[m1]=b[0][m2];
s[m1]="y=f(x)";
for(int j=0;j<m1;j++)
   { s[j]="x["+j+"]";}
for(int i=1;i<n;i++)
{
   for(int j=0;j<m1;j++)
   {
                if((b[i][j]==1.0)&& (b[0][j]==0.0))
        { d[j]=b[i][m2];}
   }
```

```
    }
    Text.print(d,s,"Simplex algorithm");
    return d;
    }
            public static void main(String arg[])
            {
            double a[][]=Text.readDouble_TextArea();
            double d[]=simplex(a);
            System.out.println(Matrix.toString(d));
            }
}
```



(bolum5_044.jpg,Resimaltı **: Lineer simpleks optimizasyonu**)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" OPO25
     -60.000000000000000     -50.000000000000000       0.000000000000000       0.000000000000000
0.000000000000000
       2.000000000000000       4.000000000000000       1.000000000000000       0.000000000000000
80.000000000000000
       3.000000000000000       2.000000000000000       0.000000000000000       1.000000000000000
60.000000000000000

keynumber = 3.0
keyrow = 2 keycolumn = 0
       0.000000000000000     -10.000000000000000       0.000000000000000      20.000000000000000
1200.000000000000000
       0.000000000000000       2.666666666666667       1.000000000000000      -0.666666666666667
40.000000000000000
       1.000000000000000       0.666666666666667       0.000000000000000       0.333333333333333
20.000000000000000

keynumber = 2.666666666666667
keyrow = 1 keycolumn = 1
       0.000000000000000       0.000000000000000       3.750000000000000      17.500000000000000
1350.000000000000000
       0.000000000000000       1.000000000000000       0.375000000000000      -0.250000000000000
14.999999999999998
       1.000000000000000       0.000000000000000      -0.250000000000000       0.500000000000000
10.000000000000002

     10.000000000000002      14.999999999999998            1350.000000000000000
```

İkinci bir örnek problem :

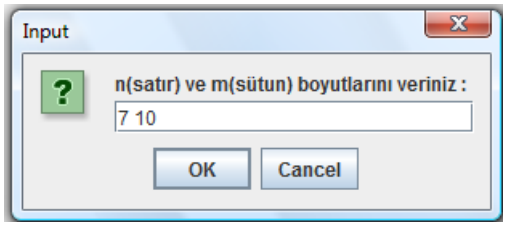
Bir spor malzemesi üreten şirketin tenis raketleri üreten 3 fabrikası bulunmaktadır. Tenis raketlerinin Ağır, orta ve hafif olarak 3 türü bulunmaktadır. Bu üç türün hepsi de 3 fabrikada da üretilebilmektedir. Fabrikaların kapasiteleri günde 550, 650 ve 300 raketle sınırlıdır. Her tür raketin fabrikaya göre kar oranları değişik prosesler kullanıldığından farklıdır. Fabrikalara göre raketlerin kar oranları aşağıdaki tabloda verilmiştir.

Raket tipi	Fabrika 1	Fabrika 1	Fabrika 1
Ağır	10 TL	8 TL	12 TL
Orta	7 TL	9 TL	8 TL
Hafif	8 TL	5 TL	4 TL

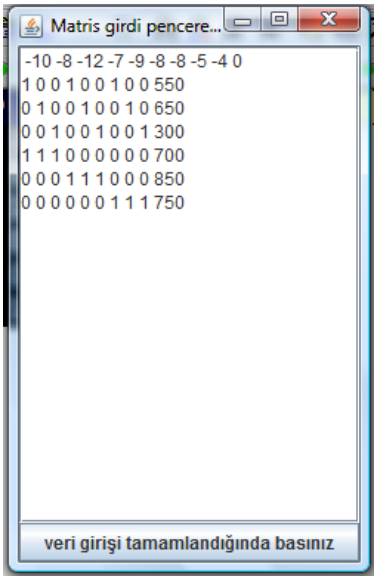
Satış bölümü yıllık satış tahminlerini ağır raketler için 700 birim, orta raketler için 850 birim ve hafif raketler için 750 birim olarak vermiştir.
En fazla kar etmek için(Karı maksimize etmek için) her fabrikada hangi türden kaç raket üreteceğimizi bulunuz.
Denklem seti
Maximum : 10xh1+8xh2+12xh3+7xm1+9xm2+8xm3+8xl1+5xl2+4xl3
Sınır şartları :
xh1+xm1+xl1<=550
xh2+xm2+xl2<=650
xh3+xm3+xl3<=300
xh1+xh2+xh3<=700
xm1+xm2+xm3<=850
xl1+xl2+xl3<=750

olarak verilir. Denklemi simpleks fromunda yazarsak:

z-10xh1-8xh2-12xh3-7xm1-9xm2-8xm3-8xl1-5xl2-4xl3=0
xh1+xm1+xl1+W1=550
xh2+xm2+xl2+W2=650
xh3+xm3+xl3+W3=300
xh1+xh2+xh3+W4=700
xm1+xm2+xm3+W5=850
xl1+xl2+xl3+W6=750
şimdi katsayılar matrisini programımıza girelim :



(bolum5_048.jpg,Resimaltı **:**)



(bolum5_049.jpg,Resimaltı **:**)

Simpleks algoritması çözüm stepleri :

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" OPO25
m=10n=7
   -10    -8   -12    -7    -9    -8    -8    -5    -4     0     0     0     0     0     0     0
     1     0     0     1     0     0     1     0     0     1     0     0     0     0     0   550
     0     1     0     0     1     0     0     1     0     0     1     0     0     0     0   650
     0     0     1     0     0     1     0     0     1     0     0     1     0     0     0   300
     1     1     1     0     0     0     0     0     0     0     0     0     1     0     0   700
     0     0     0     1     1     1     0     0     0     0     0     0     0     1     0   850
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1   750

keynumber = 1.0
keyrow = 3 keycolumn = 2
   -10    -8     0    -7    -9     4    -8    -5     8     0     0    12     0     0     0  3600
     1     0     0     1     0     0     1     0     0     1     0     0     0     0     0   550
     0     1     0     0     1     0     0     1     0     0     1     0     0     0     0   650
     0     0     1     0     0     1     0     0     1     0     0     1     0     0     0   300
     1     1     0     0     0    -1     0     0    -1     0     0    -1     1     0     0   400
     0     0     0     1     1     1     0     0     0     0     0     0     0     1     0   850
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1   750

keynumber = 1.0
keyrow = 4 keycolumn = 0
     0     2     0    -7    -9    -6    -8    -5    -2     0     0     2    10     0     0  7600
     0    -1     0     1     0     1     1     0     1     1     0     1    -1     0     0   150
     0     1     0     0     1     0     0     1     0     0     1     0     0     0     0   650
     0     0     1     0     0     1     0     0     1     0     0     1     0     0     0   300
     1     1     0     0     0    -1     0     0    -1     0     0    -1     1     0     0   400
     0     0     0     1     1     1     0     0     0     0     0     0     0     1     0   850
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1   750

keynumber = 1.0
keyrow = 2 keycolumn = 4
     0    11     0    -7     0    -6    -8     4    -2     0     9     2    10     0     0 13450
     0    -1     0     1     0     1     1     0     1     1     0     1    -1     0     0   150
     0     1     0     0     1     0     0     1     0     0     1     0     0     0     0   650
     0     0     1     0     0     1     0     0     1     0     0     1     0     0     0   300
     1     1     0     0     0    -1     0     0    -1     0     0    -1     1     0     0   400
     0    -1     0     1     0     1     0    -1     0     0    -1     0     0     1     0   200
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1   750

keynumber = 1.0
keyrow = 1 keycolumn = 6
     0     3     0     1     0     2     0     4     6     8     9    10     2     0     0 14650
     0    -1     0     1     0     1     1     0     1     1     0     1    -1     0     0   150
     0     1     0     0     1     0     0     1     0     0     1     0     0     0     0   650
     0     0     1     0     0     1     0     0     1     0     0     1     0     0     0   300
     1     1     0     0     0    -1     0     0    -1     0     0    -1     1     0     0   400
     0    -1     0     1     0     1     0    -1     0     0    -1     0     0     1     0   200
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1   750

keynumber = 1.0
keyrow = 1 keycolumn = 6
     0     3     0     1     0     2     0     4     6     8     9    10     2     0     0 14650
     0    -1     0     1     0     1     1     0     1     1     0     1    -1     0     0   150
     0     1     0     0     1     0     0     1     0     0     1     0     0     0     0   650
     0     0     1     0     0     1     0     0     1     0     0     1     0     0     0   300
     1     1     0     0     0    -1     0     0    -1     0     0    -1     1     0     0   400
     0    -1     0     1     0     1     0    -1     0     0    -1     0     0     1     0   200
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1   750

keynumber = 1.0
keyrow = 1 keycolumn = 6
     0     3     0     1     0     2     0     4     6     8     9    10     2     0     0 14650
```

```
 0   -1   0   1   0   1   1   0   1   1   0   1   -1   0   0   150
 0    1   0   0   1   0   0   1   0   0   1   0    0   0   0   650
 0    0   1   0   0   1   0   0   1   0   0   1    0   0   0   300
 1    1   0   0   0  -1   0   0  -1   0   0  -1    1   0   0   400
 0   -1   0   1   0   1   0  -1   0   0  -1   0    0   1   0   200
 0    0   0   0   0   0   0   0   0   0   0   0    0   0   1   750

400      0      300      0      650      0      150      0      0      14650
```

Sonuçlar : x[0]=400.0, x[1]=0.0, x[2]=300.0, x[3]=0.0, x[4]=650.0, x[5]=0.0, x[6]=150.0, x[7]=0.0, x[8]=0.0
toplam kar=14650 TL
simplex verisi çok büyüdüğünde veriyi dosyadan okumak daha kolay olabilir. Veriyi dosyadan okuyan, hesaplamayı üstteki simpleks programındaki metodla yapan küçük bir ana program altta verilmiştir :

Program 5.35-2 Lineer simpleks optimizasyonu dosyadan okuyan versiyonu

```
import java.io.*;
import javax.swing.*;

class simplex1
{
        public static void main(String arg[])
        {
        String s1="simplex.txt";
        JFileChooser fc=new JFileChooser();
        if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file =
fc.getSelectedFile();s1=file.getName(); }
        double a[][]=Text.readDouble(s1);
        double d[]=simplex.simplex(a);
        System.out.println(Matrix.toString(d));
        }
}
```

5.36 Problemler

PROBLEM 1
Çürük buharla çalışan bir damıtma tankında benzen ve tolüen ayıştırılmaktadır. Sıvı fazda en yüksek tolüen saflığı elde etmek için (karışımdaki tolüen oranını maksimize etmek için)tank hangi sıcaklıkta çalıştırılmalıdır?

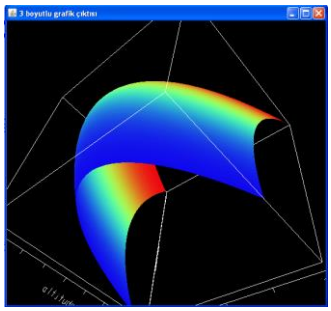
$x_{tolüen}P_{doyma\_tolüen} + x_{benzen}P_{doyma\_benzen} = P = 760$ mmHg

$\log_{10}(P_{doyma\_benzen}) = 6.905 - 1211/(T+221)$
$\log_{10}(P_{doyma\_tolüen}) = 6.953 - 1344/(T+219)$
sıcaklık C, basınç mmHg

PROBLEM 2
Karıştırılan iki adet reaktör tankında A maddesi B ye dönüştürülecektir.Ürün B ve reaksiyon girdisi A bir ayırıcıda saflaştırılmaktadır. Reaksiyona girmemiş A reaktöre geri beslenmektedir. Bir süreç mühendisi sistemin ilk yatırım maliyetinin birinci ve ikinci reaktörler için dönüşüm oranları x_{A1} ve x_{A2} nin fonksiyonu olduğunu bulmuştur. Sistemin toplam maliyetini minimum yapacak şekilde her iki reaktörün de dönüşümünü (x_{A1} ve x_{A2}) bulun.

PROBLEM 3
Bir kanaldaki kirletici derişiminin iki boyutlu dağılımı
$C(x,y)=7.9+0.13x+0.21y-0.05x^2-0.016y^2-0.007xy$

İfadesiyle tanımlanabilir. Fonksiyon ve tepe noktasının -10 <= x <= 10 ve 0 <= y <= 20 sınırları arasında kaldığı bilgisi verildiğine göre derişikliğin tepe noktasının gerçek yerini belirleyiniz.



(bolum5_050.jpg,Resimaltı **:**)

PROBLEM 4

Bir petro-kimya şirketi ticari uçaklar için yeni bir yakıt katkısı geliştirmektedir. Katkı malzemesi X,Y ve Z gibi üç bileşenden oluşmaktadır. En iyi performans için yakıta eklenen katkı en az 6 mL/L olmalıdır. Güvenlik nedeniyle çok yanıcı olan X ve Y bileşenlerinin toplamı 3 mL/L yi geçmemelidir.ayrıca X bileşeninin miktarı daima Y ye eşit veya daha fazla, Z de Ynin yarısından fazla olmalıdır. X,Y ve Z bileşenlerinin maliyetleri sırasıyla 0.15,0.025 ve 0.05 YTL ise, her litre yakıt için en düşük maliyetli katkıyı belirleyin

PROBLEM 5

Büyük bir konserve şirketi konserve içindeki gıdayı maksimuma çıkarırken malzeme maliyetini minimuma indiren yeni bir kutu tasarımı yapmaya çalışmaktadır. 320 cm^3 hacim istenmekte ve estetik kısıtlar çapın 3 ile 10 cm arasında yüksekliğin de 2 ile 10 cm arasında olmasını gerektirmektedir. Kutunun et kalınlığı 0.4 mm olup, yoğunluğu 7800 kg/m^3 ve kutu malzemesinin kilo maliyeti 37 YTL dir. Kutunun üst ve altkapak dahil aynı kalınlıkta aynı malzemeden yapıldığını varsayarak maliyeti minimum yapacak boyutu ve kutu başına maliyeti bulunuz.

PROBLEM 6

3 kademeli hava kompresör sistemine hava 1 bar 300K de girmektedir.her kademeden sonra hava orijinal 300K değerine bir ara soğutucu kullanılarak soğutulmaktadır. Kompresör çıkışı 27 bardır. Kompresör iş girişini minimize eden ara basınç değerlerini hesaplayınız

Her kademe için iş:

$W=C_p(T_i - T_e)=C_pT_i(1-T_e/T_i)=kRT_i/(k-1)*[1-(P_e/P_i)^{(k-1)/k}]$

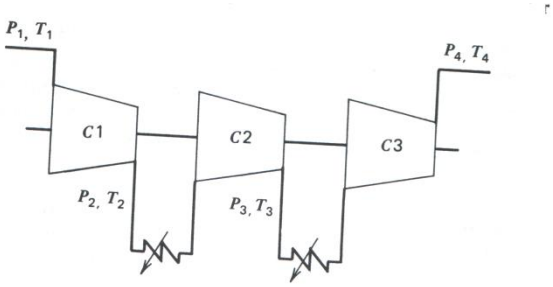
Üç kademe için toplam iş

$W=kRT_i/(k-1)*[1-(P_2/P_1)^{(k-1)/k}]+ kRT_i/(k-1)*[1-(P_3/P_2)^{(k-1)/k}]+ kRT_i/(k-1)*[1-(P_4/P_3)^{(k-1)/k}]$

$W= kRT_i/(k-1)*[3-(P_2/P_1)^{(k-1)/k}-(P_3/P_2)^{(k-1)/k}-(P_4/P_3)^{(k-1)/k}]$

$(P_2/P_1) *(P_3/P_2)*(P_4/P_3)= (P_4/P_1)=27$

$k=1.4$ $C_p=1.005$ KJ/kg K



(bolum5_051.jpg,Resimaltı **:**)

java dilinde çözüm fonksiyonu :

```
class f1 extends f_xj
{
public double func(double x[])
```

```
{
//çözümü istenen fonksiyon
 double ff=1.4*8.3145*300/(1.4-1)*(1-x[0]*(1.4-1)/1.4)+ 1.4*8.3145*300/(1.4-1)*(1-(x[1]/x[0])*(1.4-1)/1.4)
 +1.4*8.3145*300/(1.4-1)*(1-(27/x[1])*(1.4-1)/1.4);
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}}
```

cevap : 3 bar ve 9 bar

PROBLEM 7

Aşağıdaki veriyi kullanarak maksimum noktayı bulunuz. İpucu : interpolasyon polinomları kulanarak bir fonksiyon tanımına gidebilirsiniz.

X	Y
0	2
1	6
2	8
3	12
4	14
5	13
6	11
7	9
8	13
9	14.5
10	7

PROBLEM 8

```
double func(double x[])
 { double ff= 34+x[0]*(Math.sin(3.0*Math.PI*(x[0]-x[1]))+x[1]*Math.sin(20.0*Math.PI*x[1]));
   return ff; //maksimum testi
}
```

Fonksiyonu verilmiştir. Maksimumunu bulunuz.

PROBLEM 9

Antibiyotik üreten bir mayanın özgül çoğalma hızı(g) besin derişikliği (c) nin bir fonksiyonudur.

$g = 2*c/(4+0.8c+c^2+0.2c^3)$

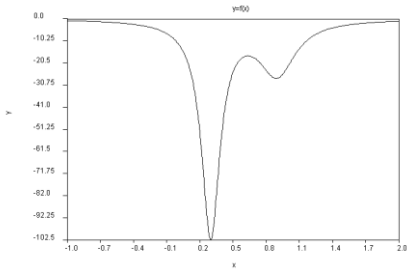
çoğalmanın maksimum olduğu c değerini bulunuz.

Besin sınırlaması nedeniyle düşük derişikliklerde büyüme hızı 0 a düşmektedir. Aynı şekilde yüksek derişikliklerde de besin zehirlenmesi sebebiyle 0 a düşmektedir. bu yüzden c = 10 mg/L den büyük değerleri incelemeye gerek yoktur.
Not: istediğiniz yöntemi kullanabilirsiniz ancak çözüme başlamadan önce fonksiyon değerlendirmesi ile maksimumun olduğu bölgeyi sınırlandırabilirsiniz, bu size işlem kolaylığı sağlayacaktır.

PROBLEM 10

f(x)=-(1.0/((x-0.3)*(x-0.3)+0.01)+1.0/((x-0.9)*(x-0.9)+0.04))

fonksiyonunun minimum değerini bulunuz.



(bolum5_052.jpg,Resimaltı :)

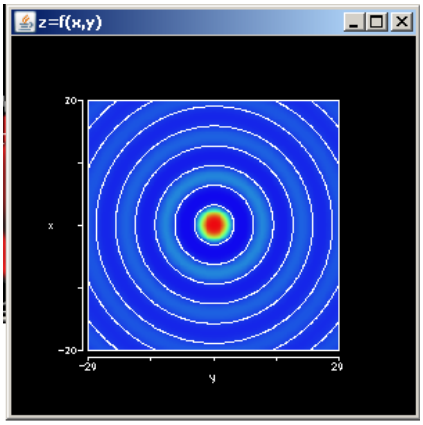
PROBLEM 11

$$z(x,y)=\frac{\sin(\sqrt{x^2+y^2+\varepsilon})}{\sqrt{x^2+y^2+\varepsilon}}$$

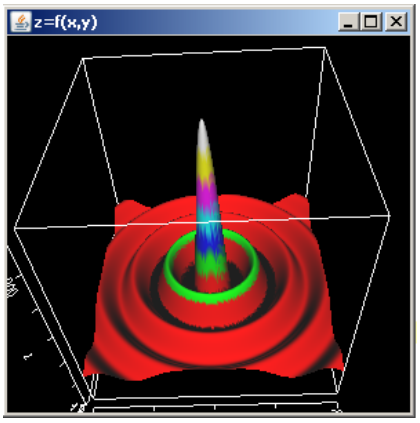
fonksiyonunun $-20\le x\le 20$

ve $-20\le y\le 20$

noktaları arasında ε=1e-2 için P(18,18) noktasından başlayarak maksimum değerini bulunuz



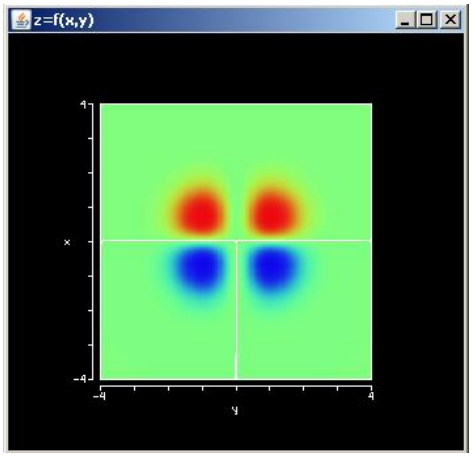
(bolum5_053.jpg,Resimaltı :)



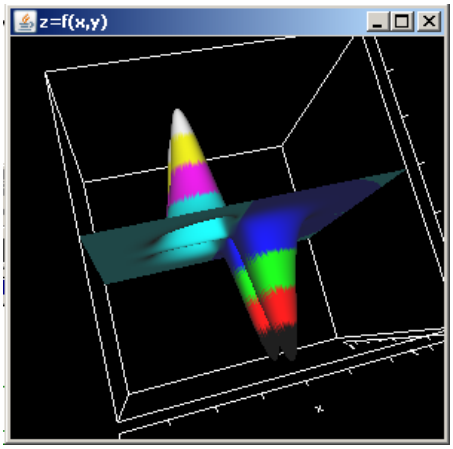
(bolum5_054.jpg,Resimaltı :)

PROBLEM 12

$z(x,y)=x^2 y\, e^{(-x^2-y^2)}$
fonksiyonunun
$-4\le x\le 4$ ve
$-4\le y\le 4$
noktaları arasında P(3,3) noktasından başlayarak minimum değerini bulunuz

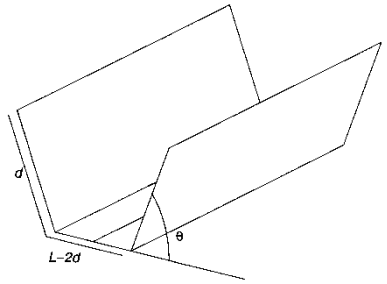


(bolum5_055.jpg,Resimaltı :)



(bolum5_056.jpg,Resimaltı :)

PROBLEM 13



(bolum5_057.jpg,Resimaltı :)

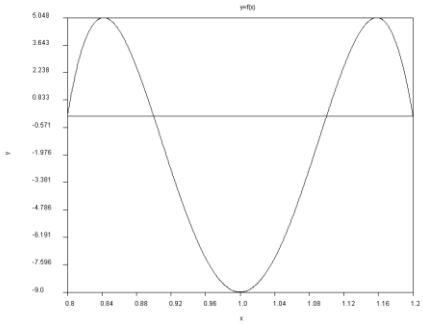
Bir evin su oluğunu yapmak için düz bir metal levha iki tarafından şekilde görüldüğü gibi eğilmiştir. Oluğun su alma kapasitesi

$V(d,\theta) = Z(d^2\sin\theta\cos\theta + (L-2d)d\sin\theta)$ şeklinde yazılabilir. Burada z oluğun boyudur (levha alanı zL dir). Oluğun su alma kapasitesini maksimum yapacak d ve **θ** değerlerini bulunuz. L=10 cm olsun.

PROBLEM 14

$f(x) = 2500x^6-15000x^5+14875x^4+40500x^3-97124x^2+73248x-19008$

fonksiyonunun x_0=0.8, x_1=1.2 arasındaki minimum değerini bulunuz.



(bolum5_058.jpg,Resimaltı :)

PROBLEM 15

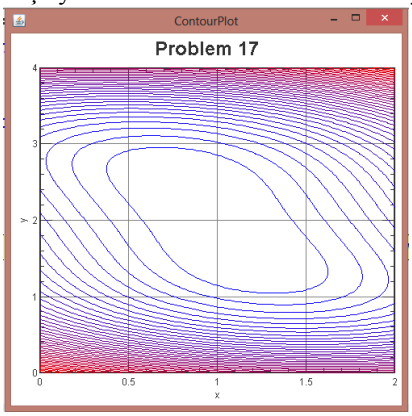
Şehir, A,B,C, ve D (0,0),(1,5),(5,2) ve (7,3) noktalarındadır. Tüm şehirlerden çıkan yollar P ortak noktasında birleşmektedir. Toplam yol uzunluğunu minimum yapan P noktasını bulunuz.

PROBLEM 16

$f(x_0, x_1, x_2, x_3) = (x_0+10\ x_1)^2 +5(x_2-10\ x_3)^2+(x_1-2\ x_2)^4+10(x_1-\ x_4)^4$
fonksiyonunun miimumunu (1,-1,0,1) noktasından başlayarak bulunuz.

PROBLEM 17

$f(x_0, x_1) = 3(x_0 - 1)^2 +2(x_1-1)\ (x_0-1)(x_1-2)+(x_1-\ 2)^4$
fonksiyonunun minumunu (1,2) noktasındadır. Üç boyutlu grafikte bu noktayı görünüz, sonra (0,0) noktasından başlayarak minimum bulma metodlarıyla minimumu bulunuz.



(bolum5_059.jpg,Resimaltı :)

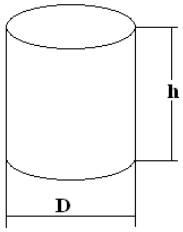
PROBLEM 18

$f(x) = \dfrac{15x}{(4x^2 - 3x + 4)}$ fonksiyonunun maksimumunu 0 ile 10 aralığında bulunuz.

PROBLEM 19

En temel optimizasyon problemlerinden birisi belli bir hacmi içinde barındıracağımız kutuyu en ucuza imal etmektir. Kutu maliyetleri temel olarak yüzey alanının fonksiyonudur. Bu yüzden verilen hacim için yüzey alanını minimize etmemiz gerekir. Örnek olarak kutu hacmini V=0.5 litre=$0.5\text{x}10^{-3}$ m^3 olarak alırsak :

Hacim $V = \dfrac{\pi D^2}{4} h$ veya buradan h yükseklik terimini çekersek $h = \dfrac{4V}{\pi D^2}$ olur.



(bolum5_060.jpg,Resimaltı :)

Silindirin yüzey alanı : $A = 2\frac{\pi D^2}{4} + \pi Dh = \frac{\pi D^2}{2} + \frac{4V}{D}$ olur.

Minimizasyon probleminin analitik çözümü birinci türevinin kökünü bulma işlemi olduğundan

$\frac{dA}{dD} = \pi D - \frac{4V}{D^2} = 0$

$D = \sqrt[3]{\frac{4V}{\pi}}$ olarak bulunur. Buradan çözüm $D = \sqrt[3]{\frac{4*0.5x10^{-3}}{\pi}}$ =0.086025401 m dir.

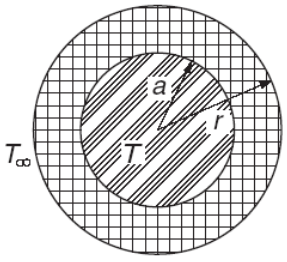
Bu değeri h teriminde yerine koyarsak $h = \frac{4*0.5x10^{-3}}{\pi D^2}$ = 0.086025401 m olarak bulunur.

Şimdi bu değerleri bilgisayar ortamında sayısal yöntemlerle çözelim : 0.01<=D<=0.2 arasında

a) Grafik yöntemle çözüm grafik çözüm : PlotT01.java
b) Altın oran (fibonnachi yöntemi) : FOPO1.java
c) Newton-Raphson kök bulma yöntemi : FOPO3.java
d) İkiye bölme kök bulma yöntemi : FOPO3Cjava
e) Newton-Raphson-ikiye bölme entegre kök bulma yöntemi : FOPO3A.java
f) Sekant(kiriş) kök bulma yöntemi : FOPO3B.java
g) Brent kök bulma yöntemi : FOPO3D.java
h) İkinci derece polinom geometrik arama optimizasyon yöntemi : FOPO2.java
i) Üçüncü derece polinom geometrik arama optimizasyon yöntemi : FSCO4E.java
j) Brent optimizasyon yöntemi : OPO9.java

Kitapta bu entegre yöntemlerin hepsi verilmemiştir, ancak kök bulma yöntemlerinde tanımlanmıştır, programları geliştiriniz.

PROBLEM 20



(bolum5_061.jpg,Resimaltı :)

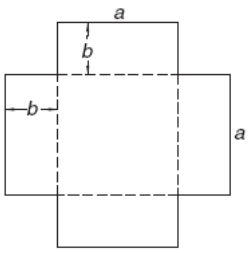
Elektrik akım taşıyan bir tel etrafında yarıçapı r olan plastik bir izolasyon bulunmaktadır. Teldeki elektriksel direnç ısı oluşturmaktadır. Bu ısı plastik izolasyondan ısı iletim mekanizmalarıyla akmakta ve ısı taşınımıyla T_∞ sıcaklığındaki çevre havasına taşınmaktadır. Telin sıcaklığı :

$T = \frac{q}{2\pi}\left(\frac{\ln(r/a)}{k} + \frac{1}{hr}\right) + T_\infty$ denklemiyle verilir. Burada

q = teldeki ısı oluşumu = 50 W/m
a = telin çapı = 5 mm = $5x10^{-3}$ m
k = ısıl iletim katsayısı = 0.2 W/(mK)
h=hava için taşınım katsayısı = 30 W/(m^2K)
T_∞ = çevre sıcaklığı = 300 K olarak verilmiştir.

T sıcaklığını minimum yapacak r yarıçapını bulunuz.

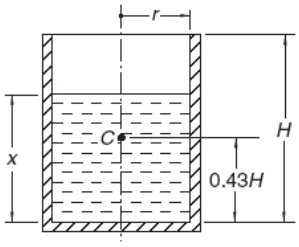
PROBLEM 21



(bolum5_062.jpg,Resimaltı :)

Mukavva kutu yapılmak için şekilde görüldüğü gibi kıvrılmaktadır. Kıvrıldıktan sonra kutu hacmi 1 m^3 olacaktır. En az miktarda mukavva harcamak için a ve b boyutlarının ne olması gerektiğini hesaplayınız.

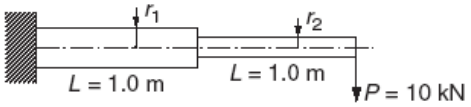
PROBLEM 22



(bolum5_063.jpg,Resimaltı :)

Silindir şeklindeki bir kabın kütlesi M ve boş ağırlık merkezi C=0.43H dir. Kabın içindeki suyun derinliği h ise su ve kabın ortak ağırlık merkezinin mümkün olduğu kadar aşağıda olması için x değerinin ne olması gerektiğini hesaplayınız.

PROBLEM 23



(bolum5_064.jpg,Resimaltı :)

iki değişik çaptaki silindir kesitli bir kiriş şekildeki gibi yerleştirilmiştir. Kirişin mümkün olan en düşük kesit alanına sahip olması istenmektedir. İşletme şartları

$\sigma_1 \le 180$ MPa, $\sigma_2 \le 180$ MPa, ve sehim $\delta \le 25$ mm dir

$\sigma_1 = \dfrac{8PL}{\pi\ r_1^3}$ =sol kısımdaki maksimum gerilim

$\sigma_2 = \dfrac{8PL}{\pi\ r_2^3}$ =sağ kısımdaki maksimum gerilim

$\delta = \dfrac{4PL^3}{3\pi\ E}\left(\dfrac{7}{r_1^4} + \dfrac{1}{r_2^4}\right)$ =uç noktadaki eğilme miktarı

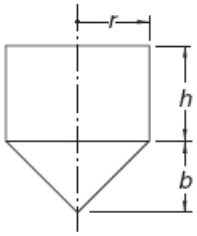
ve E =Young modülü=200 GPa dır. r_1 ve r_2 yi hesaplayınız.

PROBLEM 24

Şekilde görülen koni tabanlı açık üst taraflı silindirik kabın V hacminin 1 m^3 olması istenmektedir. Yüzey alanı S yi minimize eden r,h ve b boyutlarını bulunuz.

$V = \pi r^2\left(\dfrac{b}{3} + h\right)$

$S = \pi r\left(2h + \sqrt{b^2 + r^2}\right)$



(bolum5_065.jpg,Resimaltı :)

PROBLEM 25

İki molekül arasındaki Lennard-Jones potansiyeli

$V = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$ denklemiyle tanımlanır. Buradaki ε ve σ sabit değerlerdir. V potansiyelini minimum yapan

$\left(\frac{\sigma}{r}\right)$ değerini bulunuz.

PROBLEM 26

Hidrojen atomu için dalga fonksiyonu

$\psi = C(27 - 18\sigma + 2\sigma^2)e^{-\sigma/3}$ fonksiyonu ile tanımlanabilir. Buradaki

$\sigma = zr / a_0$

$C = \frac{1}{81\sqrt{3\pi}}\left(\frac{z}{a_0}\right)^{2/3}$

z = çekirdek yük miktarı

a_0 = Bohr yarıçapı

r =radyal mesafedir.

ψ dalga fonksiyonunu minimum yapan σ değerini bulunuz.

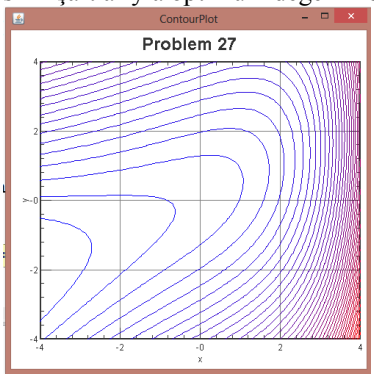
PROBLEM 27

$f = e^{x1} - x_1x_2 + x_2^2$ fonksiyonunun

$g = 2x_1 + x_2 - 2 \le 0$

$h = x_1^2 + x_2^2 - 4 = 0$

sınır şartlarıyla optimum değerini bulunuz.

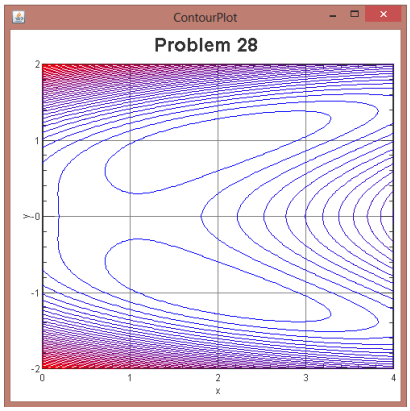


(bolum5_066.jpg,Resimaltı :)

```
class f7 extends f_xj
{
double func(double x[])
{
double first=0.0;
first=Math.exp(x[0])-x[0]*x[1]+x[1]*x[1];
return first;
}
}
```

PROBLEM 28

$f = (x-2)^2 + (x-2y^2)^2$ fonksiyonunun minimum noktalarını bulunuz.

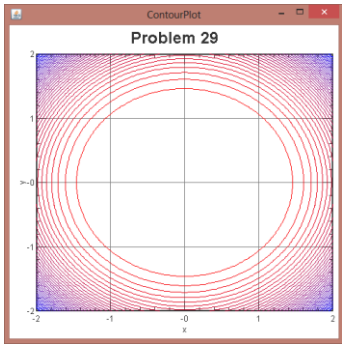


(bolum5_067.jpg,Resimaltı :)

```
class f7 extends f_xj
{
double func(double x[])
{
double first=0.0;
first=(x[0]-2.0)*(x[0]-2.0)+(x[0]-2.0*x[1]*x[1])*(x[0]-2.0*x[1]*x[1]);
return first;
}              }
```

PROBLEM 29

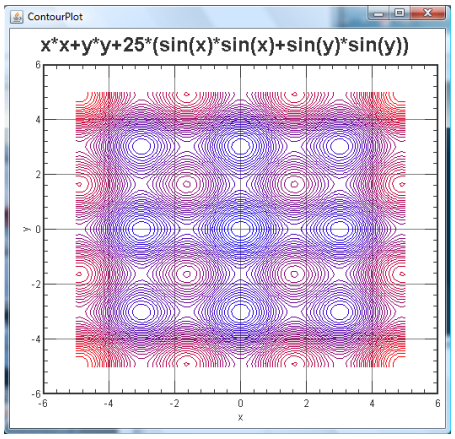
$f = 25x^2 - 12x^4 + 6xy + 25y^2 - 24x^2y^2 - 12y^4$ fonksiyonunun minimum noktasını bulunuz.



(bolum5_068.jpg,Resimaltı :)

PROBLEM 30

$f = x^2 + y^2 + 25[\sin^2(x) + \sin^2(y)]$ fonksiyonunun global minimumu f=0 (0,0) noktasındadır. Bir program kodu geliştirerek bulunuz.



(bolum5_066.jpg,Resimaltı :)

PROBLEM 31

Çeşitli göreceli zor test fonksiyonları aşağıda java kodu olarak verilmiştir. Bu fonksiyonların global optimum değerlerini istatistiki yöntemleri kullanarak bulunuz.

```
class f1 extends f_xj//Gold stein f(x)=3.0 @x=(0,-1) -2<x[i]<2 i=1,2
{
double func(double x[])
{
double first=0.0;double second=0.0;
first=(1.0+(x[0]+x[1]+1.0)*(x[0]+x[1]+1.0)*(19.0-14.0*x[0]+3.0*x[0]*x[0]-14.0*x[1]+6.0*x[0]*x[1]+3.0*x[1]*x[1]));
second=30.0+(2.0*x[0]-3.0*x[1])*(2.0*x[0]-3.0*x[1])*(18.0-32.0*x[0]+12.0*x[0]*x[0]+48.0*x[1]-36.0*x[0]*x[1]+27*x[1]*x[1]);
return first*second;
}
}

class f2 extends f_xj// Beale f(x)=0    @x=(3,0.5)   -4.5<x[i]<4.5, i = 1, 2.
{
double func(double x[])
{
double first=0.0;
first=((1.5-x[0]+x[0]*x[1])*(1.5-x[0]+x[0]*x[1]))+((2.25-x[0]+x[0]*x[1]*x[1])*(2.25-x[0]+x[0]*x[1]*x[1]))+((2.625-
x[0]+x[0]*x[1]*x[1]*x[1])*(2.625-x[0]+x[0]*x[1]*x[1]*x[1]));
return first;
}
}

class f3 extends f_xj// Bohachecsky 1 f(x)=0  @x=(0.0,0.0)   -5.0<x[i]<5.0, i = 1, 2.
{
double func(double x[])
{
double first=0.0;
first=x[0]*x[0]+2.0*x[1]*x[1]-0.3*(Math.cos(Math.PI*3.0*x[0]))-0.4*Math.cos(4.0*Math.PI*x[1])+0.7;
return first;
}
}

class f4 extends f_xj// Bohachecsky 2 f(x)=0  @x=(0.0,0.0)   -5.0<x[i]<5.0, i = 1, 2.
{
double func(double x[])
{
double first=0.0;
first=x[0]*x[0]+2.0*x[1]*x[1]-(0.3*(Math.cos(Math.PI*3.0*x[0]))*Math.cos(4.0*Math.PI*x[1]))+0.3;
return first;
}
}

class f5 extends f_xj// Bohachecsky 3 f(x)=0  @x=(0.0,0.0)   -5.0<x[i]<5.0, i = 1, 2.
{
double func(double x[])
{
double first=0.0;
first=x[0]*x[0]+2.0*x[1]*x[1]-(0.3*(Math.cos(Math.PI*3.0*x[0]+Math.PI*4.0*x[1])))+0.3;
return first;
}
}

class f6 extends f_xj// Booth  f(x)=0  @x=(1.0,3.0)   -10.0<x[i]<10.0, i = 1, 2.
{
double func(double x[])
{
double first=0.0;
first=(x[0]+2.0*x[1]-7.0)*(x[0]+2.0*x[1]-7.0)+(2.0*x[0]+x[1]-5.0)*(2.0*x[0]+x[1]-5.0);
return first;
}
}

class f7 extends f_xj// Branin  f(x)=0.397887  @x=(-pi,12.275),(pi,2.275),(9.42478,2.475)   -5.0<=x[0]<=10.0, 0.0<=x[1]<=15.0
{
double func(double x[])
{
double first=0.0;
first=((x[1]-(5.1*x[0]*x[0]/(4.0*Math.PI*Math.PI))+(5.0*x[0]/Math.PI)-6.0)*(x[1]-(5.1*x[0]*x[0]/(4.0*Math.PI*Math.PI))+(5.0*x[0]/Math.PI)-
6.0))+(10.0*(1.0-(1.0/(8.0*3.1415)))*Math.cos(x[0]))+10.0;
return first;
}
```

```
}

class f8 extends f_xj// Colville  f(x)=0.0  @x=(1,1,1,1)   -10.0<=x[i]<=10.0 i=0,1,2,3
{
double func(double x[])
{
double first=0.0;
first=(100.0*(x[0]-x[1]*x[1])*(x[0]-x[1]*x[1]))+((1.0-x[0])*(1.0-x[0]))+(90.0*(x[3]-x[2]*x[2])*(x[3]-x[2]*x[2]))+((1.0-x[2])*(1.0-
x[2]))+(10.1*((x[1]-1.0)*(x[1]-1.0)+(x[3]-1.0)*(x[3]-1.0)))+(19.8*(x[1]-1.0)*(x[3]-1.0));
return first;
}
}

class f9 extends f_xj// Easom  f(x)=-1.0  @x=(pi,pi)   -100.0<=x[i]<=100.0 i=0,1
{
double func(double x[])
{
double first=0.0;
first=-Math.cos(x[0])*Math.cos(x[1])*Math.exp(-(x[0]-Math.PI)*(x[0]-Math.PI)-(x[1]-Math.PI)*(x[1]-Math.PI));
return first;
}
}

class f10 extends f_xj// Himmelblau f(x)=0.0  @x=(3.0,2.0),(-2.8051,3.1313),(-3.7793,-3.2831),(3.5844,-1.8481)   -6.0<=x[i]<=6.0 i=0,1
{
double func(double x[])
{
double first=0.0;
first=(((x[0]*x[0]+x[1]-11.0)*(x[0]*x[0]+x[1]-11.0))+(x[0]+x[1]*x[1]-7.0)*(x[0]+x[1]*x[1]-7.0));
return first;
}
}

class f11 extends f_xj// Griewank f(x)=0.0  @x=(0,0)<---global minima     several local minimas      -600<x[i]<600 i=1,2,.. x.length
{
double func(double x[])
{
  double s=0.0;
  double fact=1.0;
  int m=x.length;
  for(int i=0;i<m;i++)
  {s+=x[i]*x[i];}
  for(int i=0;i<m;i++)
  {fact*=Math.cos(x[i]/Math.sqrt(i+1));}
  return (s/4000.0)+1.0+(-fact);
}
}

class f12 extends f_xj// Hartman3 f(x)=-3.86  @x=(0.114,0.556,0.852)   0.0<x[i]<1.0 for n=3 variable
{
double func(double x[])
{
double[][] A={{3.0,10.0,30.0},{0.1,10.0,35.0},{3.0,10.0,30.0},{0.1,10.0,35.0}};
double c[]={1.0,1.2,3.0,3.2};
double p[][]={{0.3689,0.1170,0.2673},{0.4699,0.4387,0.7470},{0.1091,0.8732,0.5547},{0.03815,0.5743,0.8828}}   ;
double sin;
double sout=0.0;
   for(int i=0;i<=3;i++)
   {  sin=0.0;
                for(int j=0;j<=3;)
               {sin+=A[i][j]*(x[j]-p[i][j])*(x[j]-p[i][j]);}
               sout+=c[i]*Math.exp(-sin);}

return -sout;
}
}

class f13 extends f_xj// Matyas function f(x)=0.0 @x(0,0)  -10.0<=x[i]<=10.0
{
double func(double x[])
{
return 0.26*(x[0]*x[0]+x[1]*x[1])-0.48*x[0]*x[1];
}
}

class f14 extends f_xj //Michalewicz n=x.length=2 f(x)=-1.8013    0<=x[i]<=pi
```

```
{                                   //    =5 f(x)=-4.687658
  double func(double x[])               //   =10 f(x)=-9.66015
  {
            int n=x.length;
            double m=10.0;
            double s=0.0;
            for(int i=0;i<n;i++)
            {s+=Math.sin(x[i])*Math.pow(Math.sin(((double)i+1.0)*x[i]*x[i]/3.1415),2.0*m);}
            return -s;
  }
}

class f15 extends f_xj //Perm function!!!!!! error
{
  double func(double x[])
  {
            int n=x.length;
            double b=0.5;

            double sin;
            double sout=0.0;
            for(int k=0;k<n;k++)
            {sin=0.0;
             for(int i=0;i<n;i++)
             {sin+=(Math.pow((double)i,(double)k)+b)*(Math.pow((x[i]/(double)i),(double)k)-1.0);}
             sout+=sin*sin;
            }
            return sout;
  }
}

class f16 extends f_xj //Powell function !!Error     Must--->mod(x.length,4)==0
{
  double func(double x[])
  {
            int n=x.length;
            int m=n/4;double s=0.0;
            for(int j=1;j<=m;j++)
            {s+=((x[4*j-4]+10.0*x[4*j-3])*(x[4*j-4]+10.0*x[4*j-3]))+(Math.sqrt(5.0)*(x[4*j-2]-x[4*j-1])*(x[4*j-2]-x[4*j-1]))+(Math.pow((x[4*j-3]-
2.0*x[4*j-2]),4.0))+(Math.sqrt(10.0)*Math.pow(x[4*j-4]-x[4*j-1],4.0));}
            return s;

  }
}

class f17 extends f_xj //Shekel function     f(x)=-10.1532 m=5; @x=(4,4,4,4) 0<=x[i]<=10.0
{                                   //f(x)=-10.4029 m=7;
  double func(double x[])               //f(x)=-10.5364 m=10;
  {
            int n=x.length;
            double
A[][]={{4.0,4.0,4.0,4.0},{1.0,1.0,1.0,1.0},{8.0,8.0,8.0,8.0},{6.0,6.0,6.0,6.0},{3.0,7.0,3.0,7.0},{2.0,9.0,2.0,9.0},{5.0,5.0,3.0,3.0},{8.0,1.0,8.0,1.0},{6.
0,2.0,6.0,2.0},{7.0,3.6,7.0,3.6}};
            double c[]={0.1,0.2,0.2,0.4,0.4,0.6,0.3,0.7,0.5,0.5};
            double sin=0.0;
            double sout=0.0;
            for(int i=0;i<10;i++)
            { sin=c[i];
              for(int j=0;j<n;j++)
              {sin+=(x[j]-A[i][j])*(x[j]-A[i][j]);}
              sout+=(1.0/sin);
            }
            return -sout;
            }
}

class f18 extends f_xj //Trid function     f(x)=-50.0 x.length=6;  -x.length^2<=x[i]<=x.length^2
{
  double func(double x[])
  {
            int n=x.length;
            double s1=0.0;double s2=0.0;
            for(int i=0;i<n;i++)
            {s1+=Math.pow(x[i]-1.0,2.0);}
            for(int i=1;i<n;i++)
            {s2+=x[i]*x[i-1];}
```

```
                return s1-s2;
                }
}

class f19 extends f_xj //Zakharov function     f(x)=0.0 @x=(0,0,..)   -5<=x[i]<=10
{
  double func(double x[])
  {
                int n=x.length;
                double s1=0.0;double s2=0.0;double s3=0.0;
                for(int i=0;i<n;i++)
                {s1+=x[i]*x[i];}
                for(int i=0;i<n;i++)
                {s2+=0.5*(double)i*x[i];}
                return s1+Math.pow(s2,2.0)+Math.pow(s2,4.0);

                }
}

class f20 extends f_xj //Levy function         f(x)=0   @x=(1,1,1...) -10<x[i]<10
{
  double func(double x[])
  {
                int n=x.length;
                double z[]=new double[n];
                for(int i=0;i<n;i++)
                {z[i]=1.0+((x[i]-1.0)/4.0);}
                double s=Math.pow(Math.sin(3.1415*z[0]),2.0);
                for(int i=0;i<n-1;i++)
                {s+=Math.pow((z[i]-1.0),2.0)*(1.0+10.0*Math.pow(Math.sin(3.1415*z[i]+1.0),2.0));}
                return s+Math.pow(z[n-1]-1.0,2.0)*(Math.pow(Math.sin(2.0*3.1415*z[n-1]),2.0)+1.0);
  }
}


class f21 extends f_xj //Dixon price function         f(x)=0    -10<x[i]<10
{
  double func(double x[])
  {
                int n=x.length;
                double s1=0.0;
                for(int i=1;i<n;i++)
                {s1+=((double)i+1.0)*(2.0*x[i]*x[i]-x[i-1])*(2.0*x[i]*x[i]-x[i-1]);}
                return s1+(x[0]-1.0)*(x[0]-1.0);
                }
}

class f22 extends f_xj //Salomon's function         f(x)=0      @x=(0,0,0...) -10<x[i]<10
{
  double func(double x[])
  {
                int n=x.length;
                double s1=0.0;
   for(int i=0;i<n;i++)
   {s1+=x[i]*x[i];}
   s1=Math.sqrt(s1);
                return -Math.cos(2.0*3.1415*s1)+0.1*s1+1.0;
  }
}

class f23 extends f_xj //Whitley's function         f(x)=0      @x=(0,0,0...) -10<x[i]<10
{
  double func(double x[])
  {
                int n=x.length;
                double s1=0.0;
   for(int i=0;i<n;i++)
   {
                  for(int j=0;j<n;j++)
                  {s1+=(Math.pow((100.0*(x[i]*x[i]-x[j])*(x[i]*x[i]-x[j])+(1.0-x[j])*(1.0-x[j])),2.0)/4000.0)-Math.cos((100.0*(x[i]*x[i]-x[j])*(x[i]*x[i]-
x[j])+(1.0-x[j])*(1.0-x[j])))+1.0;}
                }
                return s1;
  }
}
```

```
class f24 extends f_xj //quartic function          f(x)=0      @x=(0,0,0...) -10<x[i]<10
{
  double func(double x[])
  {
              int n=x.length;
              double s1=0.0;
   for(int i=0;i<n;i++)
   {s1+=((double)i+1.0)*Math.pow(x[i],4.0);}
   s1+=Math.random();
              return s1;
 }
}

class f25 extends f_xj //quartic function          f(x)=0      @x=(0,0,0...) -10<x[i]<10
{
  double func(double x[])
  {
              int n=x.length;
              double s1=0.0;
   for(int i=0;i<n;i++)
   {s1+=((double)i+1.0)*Math.pow(x[i],4.0);}
   s1+=Math.random();
              return s1;
 }
}

class f26 extends f_xj //Camel Back -6 Hump function          f(x)=-1.0316285       -5<x[i]<5
{
  double func(double x[])
  {
              int n=x.length;
   double s1=4.0*x[0]*x[0]-2.1*x[0]*x[0]*x[0]*x[0]+(x[0]*x[0]*x[0]*x[0]*x[0]*x[0]/3.0)+(x[0]*x[1])-4.0*x[1]*x[1]+4.0*x[1]*x[1]*x[1]*x[1];
   return s1;
  }
}

class f27 extends f_xj //Schwefel 2.22          f(x)=0    @x=(0,0,...)  -100<x[i]<100
{
  double func(double x[])
  {
              int n=x.length;
              double s1=0.0;
              double f1=1.0;
   for(int i=0;i<n;i++)
   {s1+=Math.abs(x[i]);f1*=Math.abs(x[i]);}
   return s1+f1;
  }
}

class f28 extends f_xj //Kowalik          f(x)=3.0748e-4    @x=(0,0,...)  -5<x[i]<5
{
  double func(double x[])
  {
              int n=x.length;
              double a[]={0.1957,0.1947,0.1735,0.16,0.0844,0.0627,0.0456,0.0342,0.0323,0.0235,0.0246};
   double b[]={0.25, 0.5, 1.0, 2.0, 4.0, 6.0, 8.0, 10.0, 12.0, 14.0, 16.0};
   double s1=0.0;
   for(int i=0;i<11;i++)
   {s1+=Math.pow((a[i]-((x[0]*(1.0+x[1]*b[i]))/(1.0+x[2]*b[i]+x[3]*b[i]*b[i]))),2.0);              }
   return s1;
  }
}

class f29 extends f_xj //Schaffer function          f(x)=0    @x=(0,0,...)  -100<x[i]<100
{
  double func(double x[])
  {
              int n=x.length;
              double s1=0.0;
              for(int i=0;i<n;i++)
              {s1+=x[i]*x[i];}
              double s2=Math.sqrt(s1);
              return 0.5+((Math.pow(Math.sin(s1),2.0)-0.5)/(1.0+0.001*s1));
              }
}
```

```
class f30 extends f_xj  // Rosenbrock's valley     f(x)=0.0     -2.048<x[i]<2.048
{
double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;
double ff=0.0;
for(int i=0;i<n-1;i++)
{ff+=(100.0*(x[i+1]-x[i]*x[i])*(x[i+1]-x[i]*x[i])+(1.0-x[i])*(1.0-x[i]));}
return ff;
}
}

class f31 extends f_xj // De Jong's first function      f(x)=0  @x=(0,0,...)     -5.12<x[i]<5.12
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0;
int n=x.length;
for(int i=0;i<n;i++)
{ff+=x[i]*x[i];}
return ff;
}
}

class f32 extends f_xj //Axis parallel hyper-ellipsoid 2.2    f(x)=0  @x=(0,0,...)     -5.12<x[i]<5.12
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0;
int n=x.length;

for(int i=0;i<n;i++)
{ff+=i*x[i]*x[i];}
return ff;
}
}

class f33 extends f_xj //Rotated hyper-ellipsoid function  -65.536<x[i]<65.536  f(x)=0   @x=(0,0,...)
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0;
int n=x.length;

for(int i=0;i<n;i++)
{
  for(int j=0;j<i;j++)
  {ff+=x[j]*x[j];}
}
return ff;
}
}

class f34 extends f_xj //Rastrigin's function 2.5        f(x)=0  @x=(0,0,...)     -5.12<x[i]<5.12
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff=0;
int n=x.length;
for(int i=0;i<n;i++)
{
ff+=x[i]*x[i]-10*Math.cos(2.0*Math.PI*x[i]);
}
return ff+10*n;
}
}

class f35 extends f_xj //sum of a different power function 2.8       f(x)=0   @x=(0,0,...)      -1<x[i]<1
{
public double func(double x[])
```

```
{
//çözümü istenen fonksiyon
double ff=0;
int n=x.length;
double top=0;
for(int i=0;i<n;i++)
{top+=Math.pow(Math.abs(x[i]),(i+2));}
return top;
}
}

class f36 extends f_xj //Ackley's function 2.9        f(x)=0;      @x=(0,0,0...)     -32.768<x[i]<32.768
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double a=20.0;
double b=0.2;
double c=2.*Math.PI;

int n=x.length;

double r=Math.PI/180;
double top1=0.0;
for(int i=0;i<n;i++)
{top1+=x[i]*x[i];}
top1=Math.sqrt(top1/n);
double top2=0.0;
for(int i=0;i<n;i++)
{top2+=Math.cos(r*c*x[i]);}
top2=top2/n;
double top=-a*Math.exp(-b*top1)-Math.exp(top2)+a+Math.exp(1);
return top;
}
}

class f37 extends f_xj //"Drop wave" function ++        i=1,2...    f(x)=-1.0  @x=(0,0)   -5.12<x[i]<5.12
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double r=Math.PI/180;
double top1=-(1+Math.cos((12*Math.sqrt(x[0]*x[0]+x[1]*x[1]))))/(0.5*(x[0]*x[0]+x[1]*x[1])+2.0);
return top1;
}
}

class f38 extends f_xj //Shubert's function   -10<x[i]<10   18 global minima
{
public double func(double x[])
{
//çözümü istenen fonksiyon

double r=Math.PI/180;
double top1=0;
double top2=0;
for(int i=1;i<=5;i++)
{top1+=i*Math.cos((i+1)*x[0]+1);}
for(int i=1;i<=5;i++)
{top2+=i*Math.cos((i+1)*x[1]+1);}
return -top1*top2;
}
}

class f39 extends f_xj //Fletcher and Powell function---> min(alfa)<x[i]<max(alfa)  f(x)=0 @x=(alfa[1],alfa[2],,,alfa[i])
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double a[][]={{-79.0, 56.0, -62.0, -9.0, 92.0, 48.0, -22.0, -34.0, -39.0, -40.0, -95.0, -69.0, -20.0, -66.0, -98.0, -66.0,-67.0, 37.0,-83.0, -45.0},{91.0, -
9.0, -18.0, -59.0, 99.0, -45.0, 88.0, -14.0, -29.0, 26.0, 71.0, -65.0, 19.0, 45.0, 88.0, 18.0, -11.0, -81.0, -10.0, 42.0},{-38.0, 8.0, -12.0, -73.0, 40.0, 26.0,
-64.0, 29.0, -82.0, -32.0, -89.0, -3.0, 88.0, 98.0, 53.0, 58.0, 45.0, -39.0, 34.0, -23.0},{-78.0,-18.0 -49.0, 65.0, 66.0, -40.0, 88.0, -95.0, -57.0, 10.0, -
98.0, -11.0, -16.0, -55.0, 33.0, 84.0, 21.0, -43.0, 45.0, 100.0,},{-1.0, -43.0, 93.0, -18.0, -76.0, -68.0, -42.0, 22.0, 46.0, -14.0, 69.0, 27.0, -12.0, -26.0,
57.0, -13.0, 0.0, 1.0, 56.0, 17.0},{34.0, -96.0, 26.0, -56.0, -36.0, -85.0, -62.0, 13.0, 93.0, 78.0, -43.0, 96.0, 77.0, 65.0, -34.0, -52.0, 82.0, 18.0, -59.0, -
55.0},{52.0, -46.0, -69.0, 99.0, -47.0, -72.0, -11.0, 55.0, -55.0, 91.0, -30.0, 7.0, -35.0, 23.0, -20.0, 55.0, 61.0, -39.0, -58.0, 13.0},{81.0, 47.0, 35.0,
55.0, 67.0, -13.0, 33.0, 14.0, 83.0, -42.0, 8.0, -45.0, -44.0, 12.0, 100.0, -9.0,-33.0, -11.0, 21.0, 14.0},{5.0, -43.0, -45.0, 46.0, 56.0, -94.0, -62.0, 52.0,
```

```
66.0, 55.0, -86.0, -29.0, -52.0, -71.0, -91.0, -46.0, 27.0, -27.0, 6.0, 67.0},{-50.0, 66.0, -47.0, -75.0, 89.0, -16.0, 82.0, 6.0, -85.0, -62.0, -30.0, 31.0, -7.0,
-75.0, -26.0, -24.0, 46.0, -95.0, -71.0, -57.0},{24.0, 98.0, -50.0, 68.0, -97.0, -64.0, -24.0, 81.0, -59.0, -7.0, 85.0, -92.0, 2.0, 61.0, 52.0, -59.0, -91.0,
74.0, -99.0, -95.0},{-30.0, -63.0, -32.0, -90.0, -35.0, 44.0, -64.0, 57.0, 27.0, 87.0, -70.0, -39.0, -18.0, -89.0, 99.0, 40.0, 14.0, -58.0 ,-5.0, -42.0},{56.0,
3.0, 88.0, 38.0, -14.0, -15.0, 84.0, -9.0, 65.0, -20.0, -75.0, -37.0, 74.0, 66.0, -44.0, 72.0, 74.0, 90.0, -83.0, -40.0},{84.0, 1.0, 73.0, 43.0, 84.0, -99.0, -
35.0, 24.0, -78.0, -58.0, 47.0, -83.0, 94.0, -86.0, -65.0, 63.0, -22.0, 65.0, 50.0, -40.0},{-21.0, -8.0, -48.0, 68.0, -91.0, 17.0, -52.0, -99.0, -23.0, 43.0, -
8.0, -5.0, -98.0, -17.0, -62.0, -79.0, 60.0, -18.0, 54.0, 74.0},{35.0, 93.0, -98.0, -88.0, -8.0, 64.0, 15.0, 69.0, -65.0, -86.0, 58.0, -44.0, -9.0, -94.0, 68.0, -
27.0, -79.0, -67.0, -35.0, -56.0},{-91.0, 73.0, 51.0, 68.0, 96.0, 49.0, 10.0, -13.0, -6.0, -23.0, 50.0, -89.0, 19.0, -67.0, 36.0, -97.0, 0.0, 3.0, 1.0,
39.0},{53.0, 66.0, 23.0, 10.0, -33.0, 62.0, -73.0, 22.0, -65.0, 37.0, -83.0, -65.0, 59.0, -51.0, -56.0, 98.0,-57.0, -11.0, -48.0, 88.0},{83.0, 48.0, 67.0,
27.0, 91.0, -33.0, -90.0, -34.0, 39.0, -36.0, -68.0, 17.0, -7.0, 14.0, 11.0, -10.0, 96.0, 98.0, -32.0, 56.0},{52.0, -52.0, -5.0, 19.0, -25.0, 15.0, -1.0, -11.0,
8.0, -70.0, -4.0, -7.0, -4.0, -6.0, 48.0, 88.0, 13.0, -56.0, 85.0, -65.0}};
double b[][]={{-65.0, -11.0, 76.0, 78.0, 30.0, 93.0, -86.0, -99.0, -37.0, 52.0, -20.0, -10.0, -97.0, -71.0, 16.0, 9.0, -99.0, -84.0, 90.0, -18.0, -94.0},
{59.0, 67.0, 49.0, -45.0, 52.0, -33.0, -34.0, 29.0, -39.0, -80.0, 22.0, 7.0, 3.0, -19.0, -15.0, 7.0, -83.0, -4.0, 84.0 -60.0, -4.0},
{21.0, -23.0, -80.0, 86.0, 86.0, -30.0, 39.0, -73.0, -91.0, 5.0, 83.0, -2.0, -45.0, -54.0, -81.0, -8.0, 14.0, 83.0, 73.0, 45.0, 32.0},
{-91.0, -75.0, 20.0, -64.0, -15.0, 17.0, -89.0, 36.0, -49.0, -2.0, 56.0, -6.0, 76.0, 56.0, 2.0, -68.0, -59.0, -70.0, 48.0, 2.0, 24.0},
{-79.0, 99.0, -31.0, -8.0, -67.0, -72.0, -43.0, -55.0, 76.0, -57.0, 1.0, -58.0, 3.0,-59.0, 30.0, 32.0, 57.0, 29.0, 66.0, 50.0, -80.0},
{-89.0, -35.0, -55.0, 75.0, 15.0, -6.0, -53.0, -56.0, -96.0, 87.0, -90.0, -93.0, 52.0, -86.0, -38.0, -55.0, -53.0, 94.0, 98.0, 4.0,-79.0},
{-76.0, 45.0, 74.0, 12.0, -12.0, -69.0, 2.0, 71.0, 75.0, -60.0, -50.0, 23.0, 0.0, 6.0, 44.0, -82.0, 37.0, 91.0, 84.0, -15.0, -63.0},
{-50.0, -88.0, 93.0, 68.0, 10.0, -13.0, 84.0, -21.0, 65.0, 14.0, 4.0, 92.0, 11.0, 67.0, -18.0, -51.0, 4.0, 21.0, -38.0, 75.0, -59.0},
{-23.0, -95.0, 99.0, 62.0, -37.0, 96.0, 27.0, 69.0, -64.0,-92.0, -12.0, 87.0, 93.0, -19.0, -99.0, -92.0, -34.0, -77.0, 17.0, -72.0, 29.0},
{-5.0, -57.0, -30.0, -6.0, -96.0, 75.0, 25.0, -6.0, 96.0, 77.0, -35.0, -10.0, 82.0, 82.0, 97.0, -39.0, -65.0, -8.0, 34.0, 72.0, 65.0},
{85.0, -9.0, -14.0, 27.0, -45.0, 70.0, 55.0, 26.0, -87.0, -98.0, -25.0, -12.0, 60.0, -45.0, -24.0, -42.0, -88.0, -46.0, -95.0, 53.0, 28.0},
{80.0, -47.0, 38.0, -6.0, 43.0, -59.0, 91.0, -41.0, 90.0, -63.0, 11.0, -54.0, 33.0, -61.0, 74.0, 96.0, 21.0, -77.0, -58.0, -75.0, -9.0},
{-66.0, -98.0, -4.0,96.0, -11.0, 88.0, -99.0, 5.0, 5.0, 58.0, -53.0, 52.0, -98.0, -97.0, 50.0, 49.0, 97.0, -62.0, 79.0, -10.0, -80.0},
{80.0, -95.0, 82.0, 5.0, -68.0, -54.0, 64.0, -2.0, 5.0, 10.0, 85.0, -33.0, -54.0, -30.0, -65.0, 58.0, 40.0, -21.0, -84.0, -66.0, -11.0},
{94.0, 85.0, -31.0, 37.0, -25.0, 60.0, 55.0, -13.0, 48.0, -23.0, -50.0, 84.0, -71.0, 54.0, 47.0, 18.0, -67.0, -30.0, 5.0, -46.0, 53.0},
{-29.0, 54.0, -10.0, -68.0, -54.0, -24.0, -16.0, 21.0, 32.0, 33.0, -27.0, 48.0, 37.0, -61.0, 97.0, 45.0, -90.0, 87.0, -95.0, 85.0, 67.0},
{76.0, -11.0, -48.0, 38.0, -7.0, 86.0, -55.0, 51.0, 26.0, 8.0, -96.0, 99.0, 69.0, -84.0, 41.0, 78.0, -53.0, 4.0, 29.0, 38.0, 16.0},
{-8.0, 48.0, 95.0, 47.0, 39.0, -11.0, -72.0, -95.0, -17.0, 33.0, 65.0, 96.0, -52.0, -17.0, -22.0, -15.0, -91.0, -41.0, -16.0, 23.0, 14.0},
{92.0, 87.0, 63.0, -63.0, -80.0, 96.0, -62.0, 71.0, -58.0, 17.0, -89.0, -35.0, -96.0, -79.0, 7.0, 46.0, -74.0, 88.0, 93.0, -44.0, 52.0},
{-21.0, 35.0, 16.0, -17.0, 54.0, -22.0, -93.0, 27.0, 88.0, 0.0, -67.0, 94.0, -24.0, -30.0, -90.0, -5.0, -48.0, 45.0, -90.0, 32.0, -81.0},
{-86.0, 31.0, -80.0, -79.0, -5.0, 11.0, -20.0, 9.0, 52.0, -38.0, 67.0, 64.0, -49.0, 23.0, -86.0, 39.0, -97.0, 76.0, 10.0, 81.0, 20.0}};


double alfa[]={-2.7910,2.5623,-1.0429,0.5097,-2.8096,1.1883,2.0771,-2.9926,0.0715,0.4142,-2.5010,1.7731,1.6473,0.4934,2.1038,-1.9930,0.3813,-
2.2144,-2.5572,2.9449};
int n=x.length;
double A[]=new double[x.length];
double B[]=new double[x.length];
double s1=0.0;
double s2=0.0;
for(int i=0;i<n;i++)
{s1=0.0;
  for(int j=0;j<n;j++)
  {s1+=a[i][j]*Math.sin(alfa[j])+b[i][j]*Math.cos(alfa[j]);}
 A[i]=s1;
}
for(int i=0;i<n;i++)
{s2=0.0;
  for(int j=0;j<n;j++)
  {s2+=a[i][j]*Math.sin(x[j])+b[i][j]*Math.cos(x[j]);}
 B[i]=s2;
}
double s3=0.0;
for(int i=0;i<n;i++)
{s3+=(A[i]-B[i])*(A[i]-B[i]);}
return s3;
}
}


class f40 extends f_xj //Step function   f(x)=0   @x(-0.5,-0.5)   -5.0<x[i]<5.0
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;
for(int i=0;i<n;i++)
{s1+=Math.pow(Math.abs(x[i]+0.5),2.0);}
return s1;
}
}



class f41 extends f_xj //Penalized function   f(x)=0   @x(-1.0,-1.0)   -5.0<x[i]<5.0
{
```

```
public double u(double x,double a,double k,double m)
{
  double c=0.0;

    if(x>a){c=k*Math.pow(x-a,m);return c;}
                if((x>=-a)||(x<=a)){c=0;return c;}
                if(x<-a){c=k*Math.pow(-x-a,m);return c;}

  return c;
}

public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;
double[] y=new double[n];
for(int i=0;i<n;i++)
{y[i]=(0.25*(x[i]+1.0))+1.0;}
s1=Math.pow(Math.sin(3.1415*y[0]),2.0)*10.0;

for(int i=0;i<n-1;i++)
{s1+=Math.pow((y[i]-1.0),2.0)*(10.0*Math.pow(Math.sin(3.1415*y[i+1]),2.0)+1.0);}
s1+=Math.pow((y[n-1]-1.0),2.0);
s1=s1*3.1415/(double)n;
double s2=0.0;
for(int i=0;i<n;i++)
{s2+=u(x[i],10.0,100.0,4.0);}
return s1+s2;
}
}

class f42 extends f_xj //Penalized2 function   f(x)=0   @x(1.0,1.0,....)   -5.0<x[i]<5.0
{
public double u(double x,double a,double k,double m)
{
  double c=0.0;

    if(x>a){c=k*Math.pow(x-a,m);return c;}
                if((x>=-a)||(x<=a)){c=0;return c;}
                if(x<-a){c=k*Math.pow(-x-a,m);return c;}

  return c;
}

public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;

s1=Math.pow(Math.sin(3.1415*x[0]),2.0);

for(int i=0;i<n-1;i++)
{s1+=Math.pow((x[i]-1.0),2.0)*(Math.pow(Math.sin(3.0*3.1415*x[i+1]),2.0)+1.0);}
s1+=Math.pow((x[n-1]-1.0),2.0)*(Math.pow(Math.sin(2.0*3.1415*x[n-1]),2.0)+1.0);
s1=s1*0.1;
double s2=0.0;
for(int i=0;i<n;i++)
{s2+=u(x[i],5.0,100.0,4.0);}
return s1+s2;
}
}

class f43 extends f_xj //Shekel's Foxholes
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;

double[][] a={{-32.0,-16.0,0.0,16.0,32.0,-32.0,-16.0,0.0,16.0,32.0,-32.0,-16.0,0.0,16.0,32.0,-32.0,-16.0,0.0,16.0,32.0,-32.0,-16.0,0.0,16.0,32.0},{-
32.0,-32.0,-32.0,-32.0,-32.0,-16.0,-16.0,-16.0,-16.0,-16.0,0.0,0.0,0.0,0.0,0.0,16.0,16.0,16.0,16.0,16.0,32.0,32.0,32.0,32.0,32.0}};


double sin=0.0;double sout=0.0;
for(int j=0;j<25;j++)
{   sin=0.0;
```

```
    for(int i=0;i<2;i++)
    {sin+=Math.pow((x[i]-a[i][j]),6.0);}
            sout+=1.0/((double)j+sin);
}
return 1.0/(0.002+sout);
}
}

class f44 extends f_xj //Shekel's Foxholes
{
public double func(double x[])
{
double a[][] = {
            {9.681, 0.667, 4.783, 9.095, 3.517, 9.325, 6.544, 0.211, 5.122, 2.020},
            {9.400, 2.041, 3.788, 7.931, 2.882, 2.672, 3.568, 1.284, 7.033, 7.374},
            {8.025, 9.152, 5.114, 7.621, 4.564, 4.711, 2.996, 6.126, 0.734, 4.982},
            {2.196, 0.415, 5.649, 6.979, 9.510, 9.166, 6.304, 6.054, 9.377, 1.426},
            {8.074, 8.777, 3.467, 1.863, 6.708, 6.349, 4.534, 0.276, 7.633, 1.567},
            {7.650, 5.658, 0.720, 2.764, 3.278, 5.283, 7.474, 6.274, 1.409, 8.208},
            {1.256, 3.605, 8.623, 6.905, 4.584, 8.133, 6.071, 6.888, 4.187, 5.448},
            {8.314, 2.261, 4.224, 1.781, 4.124, 0.932, 8.129, 8.658, 1.208, 5.762},
            {0.226, 8.858, 1.420, 0.945, 1.622, 4.698, 6.228, 9.096, 0.972, 7.637},
            {7.305, 2.228, 1.242, 5.928, 9.133, 1.826, 4.060, 5.204, 8.713, 8.247},
            {0.652, 7.027, 0.508, 4.876, 8.807, 4.632, 5.808, 6.937, 3.291, 7.016},
            {2.699, 3.516, 5.874, 4.119, 4.461, 7.496, 8.817, 0.690, 6.593, 9.789},
            {8.327, 3.897, 2.017, 9.570, 9.825, 1.150, 1.395, 3.885, 6.354, 0.109},
            {2.132, 7.006, 7.136, 2.641, 1.882, 5.943, 7.273, 7.691, 2.880, 0.564},
            {4.707, 5.579, 4.080, 0.581, 9.698, 8.542, 8.077, 8.515, 9.231, 4.670},
            {8.304, 7.559, 8.567, 0.322, 7.128, 8.392, 1.472, 8.524, 2.277, 7.826},
            {8.632, 4.409, 4.832, 5.768, 7.050, 6.715, 1.711, 4.323, 4.405, 4.591},
            {4.887, 9.112, 0.170, 8.967, 9.693, 9.867, 7.508, 7.770, 8.382, 6.740},
            {2.440, 6.686, 4.299, 1.007, 7.008, 1.427, 9.398, 8.480, 9.950, 1.675},
            {6.306, 8.583, 6.084, 1.138, 4.350, 3.134, 7.853, 6.061, 7.457, 2.258},
            {0.652, 2.343, 1.370, 0.821, 1.310, 1.063, 0.689, 8.819, 8.833, 9.070},
            {5.558, 1.272, 5.756, 9.857, 2.279, 2.764, 1.284, 1.677, 1.244, 1.234},
            {3.352, 7.549, 9.817, 9.437, 8.687, 4.167, 2.570, 6.540, 0.228, 0.027},
            {8.798, 0.880, 2.370, 0.168, 1.701, 3.680, 1.231, 2.390, 2.499, 0.064},
            {1.460, 8.057, 1.336, 7.217, 7.914, 3.615, 9.981, 9.198, 5.292, 1.224},
            {0.432, 8.645, 8.774, 0.249, 8.081, 7.461, 4.416, 0.652, 4.002, 4.644},
            {0.679, 2.800, 5.523, 3.049, 2.968, 7.225, 6.730, 4.199, 9.614, 9.229},
            {4.263, 1.074, 7.286, 5.599, 8.291, 5.200, 9.214, 8.272, 4.398, 4.506},
            {9.496, 4.830, 3.150, 8.270, 5.079, 1.231, 5.731, 9.494, 1.883, 9.732},
            {4.138, 2.562, 2.532, 9.661, 5.611, 5.500, 6.886, 2.341, 9.699, 6.500}};

            double c[] =
{0.806,0.517,0.100,0.908,0.965,0.669,0.524,0.902,0.531,0.876,0.462,0.491,0.463,0.714,0.352,0.869,0.813,0.811,0.828,0.964,0.789,0.360,0.369,0.992
,0.332,0.817,0.632,0.883,0.608,0.326};
            int dimension=x.length;
            double sum=0.0;
            double h=0.0;
            double sp=0.0;
            for (int j=0; j<30; j++)
            {
                        sp=0.0;
                        for (int i=0; i<dimension; i++)
                        {
                                    h=(x[i])-a[j][i];
                                    sp+=h*h;
                        }
                        sum-=1.0/(sp+c[j]);
            }


            return sum;
}
}

class f45 extends f_xj //Exp2 function   f(x)=0   @x(1.0,10.0)   0.0<x[i]<20.0
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;
for(int i=0;i<10;i++)
{s1+=Math.pow((Math.exp(-(double)i*x[0]/10.0)-(5.0*Math.exp(-(double)i*x[1]/10.0))-Math.exp(-(double)i/10.0)+5.0*Math.exp(-(double)i)),2.0);}
return s1;
```

```
}
}

class f46 extends f_xj //Stretched V function
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;
for(int i=0;i<=n-2;i++)
{s1+=Math.pow((x[i+1]*x[i+1]+x[i]*x[i]),0.25)*Math.sin(50.0*Math.pow((x[i+1]*x[i+1]+x[i]*x[i]),0.1))+1.0;}
return s1;
}
}

class f47 extends f_xj //Trecanni function   f(x)=0   @x(0.0,0.0,....)   -5.0<x[i]<5.0
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;
s1=Math.pow(x[0],4.0)+4.0*Math.pow(x[0],3.0)+4.0*Math.pow(x[0],2.0)+x[1]*x[1];
return s1;
}
}

class f48 extends f_xj //Trefethen4 function   f(x)=-3.306868   @x(-0.0244031,0.2106124)   -6.5<x[0]<6.5    -4.5<x[1]<4.5
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;
s1=Math.exp(Math.sin(50.0*x[0]))+Math.sin(60.0*Math.exp(x[1]))+Math.sin(70.0*Math.sin(x[0]))+Math.sin(Math.sin(80.0*x[1]))-
Math.sin(10.0*(x[0]+x[1]))+1.0/4.0*(x[0]*x[0]+x[1]*x[1]);
return s1;
}
}

class f49 extends f_xj //Paviani function
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;double fact=1.0;
for(int i=0;i<=n-1;i++)
{s1+=Math.pow(Math.log(x[i]-2.0),2.0)+Math.pow(Math.log(10.0-x[i]),2.0);}
for(int i=0;i<=n-1;i++)
{fact*=x[i];}
fact=Math.pow(fact,0.2);
return s1-fact;
}
}

class f50 extends f_xj //McCormick function      -1.5<=x[0]<=4 , -3.0<=x[1]<=4    f(x)=-1.9133  @x(-0.547,-1.54719)
{
public double func(double x[])
{
//çözümü istenen fonksiyon

double s1=Math.sin(x[0]+x[1])+Math.pow(x[0]-x[1],2.0)-1.5*x[0]+2.5*x[1]+1.0;
return s1;
}
}

class f51 extends f_xj //Leon function     f(x)=0.0  @x(1.0,1.0)
{
public double func(double x[])
{
//çözümü istenen fonksiyon

double s1=100.0*(x[1]-x[0]*x[0]*x[0])*(x[1]-x[0]*x[0]*x[0])+(x[0]-1.0)*(x[0]-1.0);
return s1;
}
}

class f52 extends f_xj //Hosaki function     f(x)=-2.3458  @x(4.0,2.0) Error!!!!
```

```
{
public double func(double x[])
{
//çözümü istenen fonksiyon

double s1=(1.0-8.0*x[0]+7.0*x[0]*x[0]-2.33*x[0]*x[0]*x[0]+0.25*x[0]*x[0]*x[0]*x[0])*x[1]*x[1]*Math.exp(-x[1]);
return s1;
}
}

class f53 extends f_xj //Hansen function     f(x)=-176.54  @x(-1.30,-1.42)
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double s1=0.0;double s2=0.0;
for(int i=0;i<=4;i++)
{s1+=((double)i+1.0)*Math.cos((double)i*x[0]+(double)i+1.0);}
for(int j=0;j<=4;j++)
{s2+=((double)j+1.0)*Math.cos(((double)j+2.0)*x[1]+(double)j+1.0);}
return s1*s2;
}
}

class f54 extends f_xj //Gear function     f(x)=0.0  @x(16,19,43,49)  12<=x[i]<=60
{
public double func(double x[])
{
//çözümü istenen fonksiyon

double s1=Math.pow((1.0/6.931-(x[0]*x[1])/(x[2]*x[3])),2.0);
return s1;
}
}

class f55 extends f_xj //Gear function     f(x)=0.0  @x(16,19,43,49)  12<=x[i]<=60
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double s1=0.0;int n=x.length;
for(int i=0;i<=n-2;i++)
{s1+=-(x[i+1]+47.0)*Math.sin(Math.sqrt(Math.abs(x[i+1]+0.5*x[i]+47.0)))+ Math.sin(Math.sqrt(Math.abs(x[i]-(x[i+1]+47.0))))*(-x[i]);}

return s1;
}
}

class f56 extends f_xj //Chichinadze function     f(x1, x2) = f(5.90133, 0.5) = -43.3159.  -30<=x[i]<=30
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double s1=x[0]*x[0]-12.0*x[0]+11.0+10.0*Math.cos(x[0]*3.1415/2.0)+8.0*Math.sin(5.0*3.1415*x[0])-((1.0/Math.sqrt(5.0))*Math.exp(-(x[1]-
0.5)*(x[1]-0.5)/2.0));

return s1;
}
}

class f57 extends f_xj //Zettl function     f(x1, x2) = f(-0.02990, 0.0) = -0.003791.  -30<=x[i]<=30
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double s1=Math.pow((x[0]*x[0]+x[1]*x[1]-2.0*x[0]),2.0)+0.25*x[0];
return s1;
}
}

class f58 extends f_xj //Plateau function     f(x1, x2) = f(0.0, 0.0,0.0,0.0,0.0) = 30.0  -5.12<=x[i]<=5.12
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;
```

```
for(int i=0;i<n;i++)
{s1+=Math.abs(x[i]);}
return s1+30.0;

}
}


class f59 extends f_xj //Xin She Yang function1     f(x) = f(0.0, 0.0,0.0,0.0,0.0,....) = 0.0  -2pi<=x[i]<=2pi
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;double s2=0.0;
for(int i=0;i<n;i++)
{s1+=Math.abs(x[i]);}
for(int i=0;i<n;i++)
{s2+=-Math.sin(x[i]*x[i]);}
return s1*Math.exp(-s2);
}
}

class f60 extends f_xj //     f(x) = f(1.0,0.5,0.333,...) = 0.0  -5<=x[i]<=5
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;
for(int i=0;i<n;i++)
{s1+=Math.random()*Math.abs(x[i]-(1.0/((double)i+1.0)));}
return s1;
}
}

class f61 extends f_xj //
{
public double func(double x[])
{
//çözümü istenen fonksiyon
int n=x.length;double s1=0.0;double s2=0.0;double s3=0.0;
for(int i=0;i<n;i++)
{s1+=Math.sin(x[i])*Math.sin(x[i]);}
for(int i=0;i<n;i++)
{s2+=x[i]*x[i];}
double c1=(s1-Math.exp(-s2));
for(int i=0;i<n;i++)
{s3+=Math.pow(Math.sin(Math.sqrt(Math.abs(x[i]))),2.0);}
return c1*Math.exp(-s3);
}
}

class f62 extends f_xj // Yang funtion     x(pi,pi)
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double alfa=1.0;
double beta=1.0;
int K=10;
double s1=0.0;
for(int j=0;j<K;j++)
{
  for(int i=0;i<K;i++)
  {
  s1+=Math.random()*Math.exp(-alfa*((x[0]-(double)i)*(x[0]-(double)i)+(x[1]-(double)j)*(x[1]-(double)j)));
  }

}
double s2=-5.0*Math.exp(-beta*((x[0]-3.1415)*(x[0]-3.1415)+(x[1]-3.1415)*(x[1]-3.1415)));

return s2-s1;

}
}
```

6. Eğri uydurma ve aradeğer hesabı

6.0 Giriş

Bilim dünyasında veriler çeşitli ölçüm aletleriyle ölçülerek elde edilir. Genellikle bilim insanları ölçtükleri bu değerlerin sonuçlarını çeşitli formüller şeklinde genelleştirirler. Sayısal olarak elde edilen verilerden matematiksel genel formüller oluşturma işlemine eğri uydurma diyoruz. Kullanılacak olan formüllerin formu genellikle ilgili fiziksel yasalardan yola çıkılarak irdelenen bilimdalı tarafından genellenir veya araştırıcı istediği formül formunu serbest olarak kendi seçer. Bu formüllerde genellikle veriye en iyi uyan bir katsayı setinin hesaplanması söz konusudur. Bu katsayılar formülün içine lineer olarak yerleştirilebildiği gibi lineer olmıyan bir formda da olabilirler. Katsayının formıuna göre eğri uydurma prosesi lineer ve lineer olmıyan diye 2 ye ayrılır. Her iki durumda da eğri uydurma temel olarak bir optimizasyon prosesidir. Aradeğr hesabı da eğri uydurarak veya direk hesapla elimizdeki verilere en uygun ra değerleri hesaplama yöntemidir. Genellikle aradeğer hesaplarında her zaman veri verilen noktalardan geçer. Eğri uydurmada böyle bir zorunluluk yoktur.

6.1 Polinom en küçük kareler metodu

Lisans seviyesinde yoğun olarak öğretilen ve hemen hemen her bilim insanının bildiği en yaygın eğri uydurma metodu polinom en küçük kareler metodudur.

$$f(x) = \sum_{j=0}^{m} a_j^{(m)} \phi_j(x)$$

j inci derece polinomu verilmiş olsun. Bu polinoma x_i, f_i, i=0...n verisini uydurmak istiyoruz. Bu veri seti için en uygun $a_j^{(m)}$ değerlerini bulmak istiyoruz. Bunun için

$$H(a_0^{(m)},....,a_m^{(m)}) = \sum_{i=1}^{n} w(x_i)\left[f_i - \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_i) \right]^2$$

fonksiyonunun minimum değerini bulmamız gerekir. Fonksiyondaki w(x) ağırlık fonksiyonu adını alır ve $w(x_i) \geq 0 \quad i = 1,....,n$ olmalıdır. Fonksiyonun minimum noktası türevinin 0 a eşit olduğu nokta olacaktır.

$$\frac{\partial H(a_0^{(m)},....,a_m^{(m)})}{\partial a_k^{(m)}} = 2\sum_{i=1}^{n} w(x_i)\left[f_i - \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_i) \right]\phi_k(x_i) = 0 \quad k = 0,.....,m$$

$$\left\{ \sum_{j=0}^{m} w(x_i)\phi_j(x_i)\phi_k(x_i) \right\}\left[a_j^{(m)}\right] = \left[\sum_{i=1}^{n} w(x_i)\phi_k(x_i) f_i \right] \quad k = 0,.....,m$$

Bu denklem temel olarak m+1 lineer denklem sistemidir. Şimdi problemin özel bir formuna göz atalım.

$$f(x) = \sum_{j=0}^{m} a_j^{(m)} \phi_j(x)$$

temel denkleminde $\phi_j(x) = x^j$ ise denklem ,

$$f(x) = \sum_{j=0}^{m} a_j^{(m)} x^j$$

halini alır. Aynı zamanda ağırlık fonksiyonunu da 1 olarak alalım($w(x_i)=1$). Bu durumda minimizasyon denklemi

$$\frac{\partial H(a_0^{(m)},....,a_m^{(m)})}{\partial a_k^{(m)}} = 2\sum_{i=1}^{n}\left[f_i - \sum_{j=0}^{m} a_j^{(m)} x_i^j \right] x_i^k = 0 \quad k = 0,.....,m$$

$$\sum_{k=0}^{m} a_j^{(m)} \left[\sum_{i=1}^{n} x_i^{j+k} \right] = \sum_{i=1}^{n} f_i\, x_i^k \qquad k = 0,....., m$$

şeklini alır. Bu matrisi açık formda yazacak olursak:

$$\begin{Bmatrix} n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & .... & \sum_{i=1}^{n} x_i^m \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & .... & \sum_{i=1}^{n} x_i^{m+1} \\ \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^5 & .... & \sum_{i=1}^{n} x_i^{m+2} \\ .... & .... & .... & .... & .... \\ \sum_{i=1}^{n} x_i^m & \sum_{i=1}^{n} x_i^{m+1} & \sum_{i=1}^{n} x_i^{m+2} & & \sum_{i=1}^{n} x_i^{2m} \end{Bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ .... \\ a_m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} f_i \\ \sum_{i=1}^{n} x_i\, f_i \\ \sum_{i=1}^{n} x_i^2 f_i \\ .... \\ \sum_{i=1}^{n} x_i^m f_i \end{bmatrix}$$

Bu matris lineer matris çözüm yöntemlerini kullanarak rahatlıkla çözülebilir. Belirlenen katsayılar matriste yerine konularak kullanılır. Aşağıda polinom en küçük kareler matrisi ugulama programımız verilmiştir.

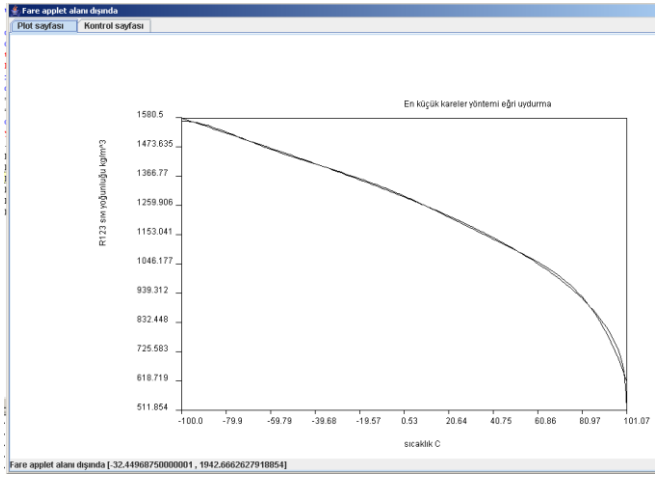
Program 6.1.1 En küçük kareler metodu programı

```
İmport java.io.*;
import javax.swing.*;

class SCO11B4
{
//en kucuk kareler metodu
  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
              p=k;
              buyuk=Math.abs(a[k][k]);
              for(ii=k+1;ii<n;ii++)
              {  dummy=Math.abs(a[ii][k]);
                if(dummy > buyuk) {buyuk=dummy;p=ii;}
              }
              if(p!=k)
              { for(jj=k;jj<n;jj++)
                { dummy=a[p][jj];
                  a[p][jj]=a[k][jj];
                  a[k][jj]=dummy;
                }
                dummy=b[p];
```

```
                b[p]=b[k];
                b[k]=dummy;
               }
               //gauss elemeyi çözme
   for(i=k+1;i<n;i++)
   {  carpan=a[i][k]/a[k][k];
     a[i][k]=0;
     for(j=k+1;j<n;j++)
     {   a[i][j]-=carpan*a[k][j]; }
        b[i]   =b[i]   -carpan*b[k];
              }
 }
 //geriye doğru yerine koyma
 x[n-1]=b[n-1]/a[n-1][n-1];
 for(i=n-2;i>=0;i--)
 {
   toplam=0;
   for(j=i+1;j<n;j++)
   {  toplam+=a[i][j]*x[j];}
   x[i]=(b[i]-toplam)/a[i][i];
 }
 return x;
}

public static double[] EKK(double xi[],double yi[],int n)
{
int l=xi.length;
int i,j,k;
int np1=n+1;
double A[][];
A=new double[np1][np1];
double B[];
B=new double[np1];
double X[];
X=new double[np1];
for(i=0;i<n+1;i++)
 {  for(j=0;j<n+1;j++)
    {if(i==0 && j==0) A[i][j]=l;
    else  for(k=0;k<l;k++) A[i][j] += Math.pow(xi[k],(i+j));
    }
    for(k=0;k<l;k++) { if(i==0)  B[i]+= yi[k];
                else     B[i] += Math.pow(xi[k],i)*yi[k];}
 }
System.out.println(Matrix.toString(A));
System.out.println(Matrix.toStringT(B));
X=pivotlugauss(A,B);
//X=B/A;
double max=0;
for(i=0;i<n+1;i++)
 if(Math.abs(X[i]) > max) max = Math.abs(X[i]);
for(i=0;i<n+1;i++)
 if((Math.abs(X[i]/max) > 0) && (Math.abs(X[i]/max) < 1.0e-100)) X[i]=0;
return X;
}

public static double funcEKK(double e[],double x)
{
// this function calculates the value of
// least square curve fitting function
int n=e.length;
double ff;
if(n!=0.0)
```

```
 { ff=e[n-1];
 for(int i=n-2;i>=0;i--)
  { ff=ff*x+e[i]; }
 }
else
 ff=0;
return ff;
}

public static double hata(double x[],double y[],double e[])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
 double total=0;
 for(k=0;k<n;k++)
 {
 total+=(y[k]-funcEKK(e,x[k]))*(y[k]-funcEKK(e,x[k]));
 }
total=Math.sqrt(total);
return total;
}

public static double[][]  funcEKK(double xi[],double yi[],int polinomkatsayisi,int aradegersayisi)
 {
               //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
               // ara değer sayısı 2 dir
                int n=xi.length;
               int nn=(n-1)*(aradegersayisi+1)+1;
               double z[][]=new double[2][nn];
   double E[]=EKK(xi,yi,polinomkatsayisi);
   System.out.println("katsayılar :"+Matrix.toStringT(E));
   double dx=0;
   int k=0;
   int i;
   for(i=0;i<(n-1);i++)
     {z[0][k]=xi[i];z[1][k]=funcEKK(E,z[0][k]);k++;
     for(int j=0;j<aradegersayisi;j++)
     {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
     z[0][k]=z[0][k-1]+dx;z[1][k]=funcEKK(E,z[0][k]);k++;}
     }
     z[0][k]=xi[i];z[1][k]=funcEKK(E,z[0][k]);
   return z;
 }


 public static void main(String args[]) throws IOException
 {
 double x[];
 double y[];
 String s1="R123trol.txt";
 JFileChooser fc=new JFileChooser();
 if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName(); }
 double a[][]=Text.readDoubleT(s1);
 x=a[0];
 y=a[1];
 double z[][]=funcEKK(x,y,6,0);
 System.out.println("E.K.K.\n"+Matrix.toStringT(z));
 Plot pp=new Plot(a[0],a[1]);
 pp.setPlabel("En küçük kareler yöntemi eğri uydurma");
 pp.setXlabel("sıcaklık C");
 pp.setYlabel("R123 sıvı yoğunluğu kg/m^3");
```

```
  pp.setPlotType(0,0);
  pp.addData(z[0],z[1]);
  pp.plot();
  }
}
```



(bolum6_000.jpg,Resimaltı: **En küçük kareler metodu programı grafik çıktısı**)

6.2 Ortogonal polinom en küçük kareler metodu

$p_j(x)$ j inci dereceden bir polinom olsun. m'inci dereceden en küçük kareler polinom fonksiyonunu da

$y_m(x) = \sum_{j=0}^{m} b_j^{(m)} p_j(x)$ şeklinde verilebilir. Eğer x_i,f_i i=1,…,n verisi için

$H(b_0^{(m)},....,b_m^{(m)}) = \sum_{i=1}^{n} w(x_i)\left[f_i - y_m(x_i)\right]^2$ fonksiyonunun minimumunu verecek b katsayılarını bulmak istersek, bir önceki bölümdekine benzer olarak

$$\frac{\partial H(b_0^{(m)},....,b_m^{(m)})}{\partial b_k^{(m)}} = 2\sum_{i=1}^{n} w(x_i)\left[f_i - \sum_{j=0}^{m} b_j^{(m)} p_j(x_i) \right] p_k(x_i) = 0 \quad k = 0,.....,m$$

$$\left\{ \sum_{j=0}^{m}\sum_{i=1}^{n} w(x_i) p_j(x_i) p_k(x_i) \right\} \left[b_j^{(m)}\right] = \left[\sum_{i=1}^{n} w(x_i) p_k(x_i) f_i \right] \quad k = 0,.....,m$$

denklemi daha kısa ifade edebilmek için:

$$d_{jk} = \left\{ \sum_{i=0}^{n} w(x_i) p_j(x_i) p_k(x_i) \right\} \quad k = 0,.....,m$$

$$\omega_{jk}\left[b_j^{(m)}\right] = \left[\sum_{i=1}^{n} w(x_i) p_k(x_i) f_i \right] \quad k = 0,.....,m$$

olarak adlandırır isek, bu durumda denklemimizi

$\sum_{j=0}^{m} d_{jk}\left[b_j^{(m)}\right] = \omega_k \quad k = 0,.....,m$ şeklinde yazabiliriz.

Halini alır. Denklemin bu şekilde yazılması hesaplama bakımından bana bir avantaj getirmemiştir. Ancak $p_k(x_i)$ fonksiyonunu özel bir form olan ortogonal polinom olarak alır isek, ortogonal polinomların özelliği gereği :

Eğer $j \neq k$ ise $\sum_{i=0}^{n} w(x_i) p_j(x_i) p_k(x_i) = 0$ olur. Bu durumda sistemimiz çok daha basit bir denklem olan

$d_{kk}\left[b_k^{(m)}\right] = \omega_k \quad k = 0,.....,m$ formunu alır. Burada $b_j^{(m)}$ değerini çözmek için denklem sistemi çözümü yerini basit bölme işlemine bırakmış olur.

$$\left[b_k^{(m)}\right] = \frac{\omega_k}{d_{kk}} \quad k = 0,.....,m$$

Ortogonal polinom olarak çeşitli polinomlar seçmek mümkündür. Programımızda kulanacağımız ortogonal polinom :
$p_{j+1}(x) = (x - \alpha_{j+1})p_j(x) - \beta_j p_{j-1}(x) \quad j = 0,1,....$

$p_0(x) = 1 \quad p_{-1}(x) = 0$

şeklindedir. α_{j+1} ve β_j hesaplanması gereken polinom sabitleridir.

$$\beta_k = \frac{\sum_{i=1}^{n} w(x_i) x_i p_{k-1}(x_i) p_k(x_i)}{\sum_{i=1}^{n} w(x_i) x_i [p_{k-1}(x_i)]^2} = \frac{\sum_{i=1}^{n} w(x_i)[p_k(x_i)]^2}{\sum_{i=1}^{n} w(x_i)[p_{k-1}(x_i)]^2}$$

$$\alpha_k = \frac{\sum_{i=1}^{n} w(x_i) x_i [p_k(x_i)]^2}{\sum_{i=1}^{n} w(x_i)[p_{k-1}(x_i)]^2}$$

Bu sabitleri kullanarak en küçük kareler hesabını şu şekilde gerçekleştiririz.

$$y_m(x) = \sum_{j=0}^{m} b_j p_j(x)$$

$$b_j = \frac{\omega_j}{\gamma_j}$$

$$\omega_j = \sum_{i=1}^{n} w_i f_i p_j(x_i)$$

$$\gamma_j = \sum_{i=1}^{n} w_i [p_j(x_i)]^2$$

Ortogonal polinom eğri uydurma , denklem sistem çözmediği için denklem sistemi çözümünden oluşan hataları da barındırmıyacaktır, bu yüzden polinom eğri uydurmaya göre tercih edilebilir, ancak polinom denkleminin basitliği bu denkleme yoğun bir uygulama alanı getirmiştir.

Program 6.2.1 Ortogonal polinom eğri uydurma

```
import java.io.*;

class SCO11A
{
public static double[][] OPEKK(double xi[],double fi[],int m)
//ortogonal polinom en küçük kareler metodu
//Referans : A First Course in Numerical Analysis
// Anthony Ralston,Philip Rabinowitz, Mc Graw Hill ISBN 0-07-051158-6
//m polinom derecesi xi fi girdi verisi
{
int i,j,k;
```

```
int n=xi.length;
int mp2=n+2;
int mp1=n+1;
double p[][]=new double[mp2][n];
double gamma[]=new double[mp1];
double beta[]=new double[mp1];
double omega[]=new double[mp1];
double alpha[]=new double[mp1];
double b[]=new double[mp1];
double wi[]=new double[n];
double a[][]=new double[3][mp1];
for(i=0;i<n;i++)
 {
  p[1][i]=1.0;
  p[0][i]=0.0;
  wi[i]=1.0;
 }
gamma[0]=0;
for(i=0;i<n;i++)
 {
 gamma[0]+=wi[i];
 }
beta[0]=0.0;
for(j=0;j<m+1;j++)
 {
 omega[j]=0;
 for(i=0;i<n;i++)
   {omega[j]+=wi[i]*fi[i]*p[j+1][i];}
 b[j]=omega[j]/gamma[j];
if( j != m)
   {
   alpha[j+1]=0;
   for(i=0;i<n;i++)
    { alpha[j+1]+=wi[i]*xi[i]*p[j+1][i]*p[j+1][i]/gamma[j];}
   for(i=0;i<n;i++)
    {p[j+2][i]=(xi[i]-alpha[j+1])*p[j+1][i]-beta[j]*p[j][i];}
    gamma[j+1]=0;
   for(i=0;i<n;i++)
    {gamma[j+1]+=wi[i]*p[j+2][i]*p[j+2][i];}
   beta[j+1]=gamma[j+1]/gamma[j];
   }
}//end of j
for(j=0;j<m+1;j++)
 {
 a[0][j]=b[j];
 a[1][j]=alpha[j];
 a[2][j]=beta[j];
 }
return a;
}

public static double funcOPEKK(double a[][],double x)
{
// polinom değerleri hesaplama fonksiyonu
double yy=0;
int k;
int m=a[0].length-1;
int mp2=m+2;
double q[];
q=new double[mp2];
//vector<double> q(m+2,0.0);
for(k=m-1;k>=0;k--)
```

```
  {
  q[k]=a[0][k]+(x-a[1][k+1])*q[k+1]-a[2][k+1]*q[k+2];
  yy=q[k];
  }
return yy;
}

public static double[][]  funcOPEKK(double xi[],double yi[],int polinomkatsayisi,int aradegersayisi)
 {
              //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
              // ara değer sayısı 2 dir
               int n=xi.length;
              int nn=(n-1)*(aradegersayisi+1)+1;
              double z[][]=new double[2][nn];
   double E[][]=OPEKK(xi,yi,polinomkatsayisi);
   double dx=0;
   int k=0;
   int i;
   for(i=0;i<(n-1);i++)
     {z[0][k]=xi[i];z[1][k]=funcOPEKK(E,z[0][k]);k++;
     for(int j=0;j<aradegersayisi;j++)
     {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
     z[0][k]=z[0][k-1]+dx;z[1][k]=funcOPEKK(E,z[0][k]);k++;}
     }
     z[0][k]=xi[i];z[1][k]=funcOPEKK(E,z[0][k]);
   return z;
  }

  public static void main(String args[]) throws IOException
  {
  double x[]={0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0};
  double y[]={0.0,0.099833,0.1986693,0.2955202,0.38941834,0.479425,0.5646424,0.6442176,0.717356,0.783326,0.84147};
  double z3[][]=funcOPEKK(x,y,3,8);
  Text.print(Text.T(z3),"Ortogonal polinom en küçük kareler ");
  Plot pp=new Plot(x,y);
  pp.setPlabel("Ortogonal polinom en küçük kareler");
  pp.setPlotType(0,20);
  pp.addData(z3[0],z3[1]);
  pp.plot();
  }
}
```

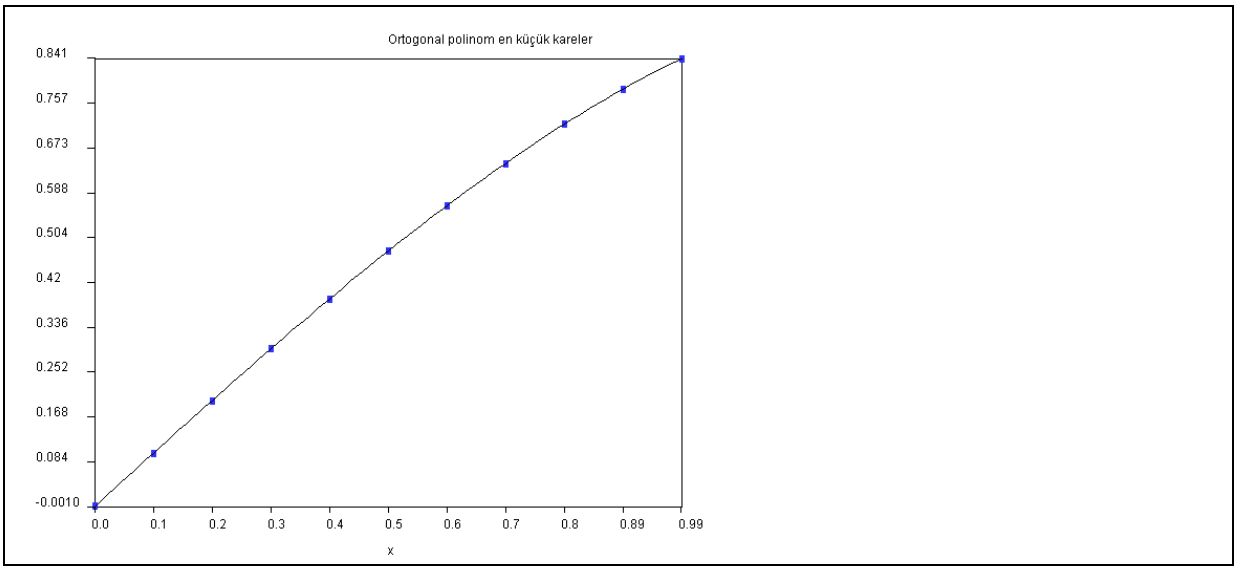
Çıktı 6.2-1 Ortogonal polinom eğri uydurma veri çıktısı

0.0	-1.3331314685310902E-4	0.4666666666666668	0.4497794627972029
0.011111111111111112	0.011022273501971321	0.4777777777777779	0.4596700067265246
0.022222222222222223	0.02217203082505974	0.48888888888888904	0.46950495493798394
0.03333333333333333	0.03331477390831393	0.5	0.4792831225174825
0.044444444444444446	0.04444931783763565	0.5111111111111111	0.48900332455092227
0.05555555555555556	0.05557447769892668	0.5222222222222221	0.49866437612420494
0.06666666666666667	0.06668906857808862	0.5333333333333332	0.5082650923232322
0.07777777777777778	0.0777919055610234	0.5444444444444443	0.5178042882339059
0.08888888888888889	0.0888818037336327	0.5555555555555554	0.5272807789421277
0.1	0.09995757818181823	0.5666666666666664	0.5366933795337994
0.11111111111111112	0.1110180439914818	0.5777777777777775	0.5460409050948227
0.12222222222222223	0.1220620162485252	0.5888888888888886	0.5553221707110993
0.13333333333333333	0.13308831003885008	0.6	0.5645359914685315
0.14444444444444443	0.14409574044835827	0.611111111111111	0.5736811824530201

0.15555555555555553	0.15508312256295145
0.16666666666666663	0.16604927146853146
0.17777777777777773	0.17699300225099998
0.18888888888888883	0.1879131299962589
0.2	0.19880846979020986
0.2111111111111111	0.20967783671875456
0.2222222222222222	0.22052004586779472
0.2333333333333333	0.23133391232323228
0.2444444444444444	0.24211825117096894
0.25555555555555554	0.25287187749690637
0.26666666666666666	0.2635936063869464
0.2777777777777778	0.2742822529269907
0.2888888888888889	0.2849366322029412
0.3	0.2955555593006993
0.3111111111111111	0.30613784930616705
0.32222222222222224	0.31668231730524615
0.33333333333333337	0.32718777838383845
0.3444444444444445	0.3376530476278455
0.3555555555555556	0.3480769401231691
0.36666666666666675	0.35845827095571103
0.3777777777777779	0.3687958552113731
0.388888888888889	0.379088507976057
0.4	0.38933504433566435
0.41111111111111115	0.3995342793760972
0.4222222222222223	0.40968502818325714
0.4333333333333334	0.4197861058430459
0.44444444444444453	0.42983632744136524
0.45555555555555566	0.439834508064117

0.6222222222222221	0.5827565587504675
0.6333333333333332	0.5917609354467753
0.6444444444444443	0.6006931276278452
0.6555555555555553	0.6095519503795791
0.6666666666666664	0.6183362187878786
0.6777777777777775	0.6270447479386455
0.6888888888888886	0.6356763529177815
0.7	0.6442298488111887
0.711111111111111	0.6527040507047684
0.7222222222222221	0.6610977736844225
0.7333333333333332	0.6694098328360527
0.7444444444444442	0.6776390432455609
0.7555555555555553	0.6857842199988488
0.7666666666666664	0.693844178181818
0.7777777777777775	0.7018177328803704
0.7888888888888885	0.7097036991804079
0.8	0.7175008921678322
0.8111111111111111	0.7252081269285447
0.8222222222222222	0.7328242185484475
0.8333333333333333	0.740347982113442
0.8444444444444443	0.7477782327094304
0.8555555555555554	0.7551137854223142
0.8666666666666665	0.7623534553379951
0.8777777777777775	0.7694960575423752
0.8888888888888886	0.7765404071213559
0.9	0.7834853191608392
0.9111111111111111	0.7903296087467265
0.9222222222222222	0.7970720909649198
0.9333333333333332	0.8037115809013209
0.9444444444444443	0.8102468936418314
0.9555555555555554	0.8166768442723531
0.9666666666666665	0.8230002478787879
0.9777777777777775	0.8292159195470372

Çıktı 6.2-2



(bolum6_001.jpg,Resimaltı: **Ortogonal polinom eğri uydurma grafik çıktısı**)

Ortogonal fonksiyonlarla eğri uydurmaya ikinci bir örnek daha verelim. Bu örneğimizde ortogonal fonksiyon olarak Chebychev polinomlarını kullanacağız. Chebychev polinomları (-1,1) bölgesinde tanımlanır. Genel tanımı
$T_n(x)=\cos(n \arccos(x))$ şeklindedir. Bu terimi açarsak
$T_0(x)=1$
$T_1(x)=x$
$T_2(x)=2x^2-x$
…..
$T_{n+1}(x)=2x\,T_n(x)-T_{n-1}(x)$
Şeklinde bir dizi olarak hesaplanabilen bir polinom fonksiyon olduğunu görürüz. Chebchev fonksiyonları için ağırlık fonksiyonu :

$w(x)=\frac{1}{\sqrt{1-x^2}}$ fonksiyonudur. Chebchev fonksiyonunun kökleri

$t_k=\cos(\pi\frac{2k+1}{2n+2})$ k=0,…..n buradaki n polinomun derecesidir. Eğri uydurma için verilerimizin kök noktalarındaki değerini bilmemiz gerekir. Ancak Chebchev polinomlarının kökleri (-1,1) bölgesinde olduğundan bundan başka bir (a,b) bölgesinde verilmiş bir verimiz varsa bunu değişken değişimi ile ayarlamamız gerekir.

$$x_k=\left(\frac{b-a}{2}\right)t_k+\left(\frac{a+b}{2}\right)$$

$$t_k=2\left(\frac{x_k-a}{b-1}\right)-1$$

eğer x_k değerlerine karşı gelen y_k değerleri bize verilmişse

$P_n(x)=\sum_{k=0}^{n}C_kT_k(x)$ polinomunun katsayılarını en küçük kareler metodu kullanarak hesaplıyabiliriz. Chebyschev polinomu ortogonal olduğundan bu işlem için bir matris çözümü gerekmez, katsayılar direk olarak bulunur.

$$C_0=\frac{1}{n+1}\sum_{k=0}^{n}y_kT_0(t_k)$$

$$C_j = \frac{2}{n+1}\sum_{k=0}^{n} y_k T_j(t_k) \qquad \text{j=1,2,...n}$$

Örnek problem olarak f(x)=e^x fonksiyonunun 0 ile 2 arasındaki değerlerini değerlendirerek bir veri seti oluşturduk

x_k	y_k
0.010178558	1.010231
0.090368005	1.094577
0.244250426	1.276664
0.459359183	1.583059
0.718267443	2.050877
1	2.718282
1.281732557	3.602877
1.540640817	4.66758
1.755749574	5.787784
1.909631995	6.750604
1.989821442	7.314228

x_k değerlerinin Chebychev kök değerlerinde olduğuna tekrar dikkatinizi çekelim.

Program 6.2-2 Chebychev ortogonal polinom eğri uydurma

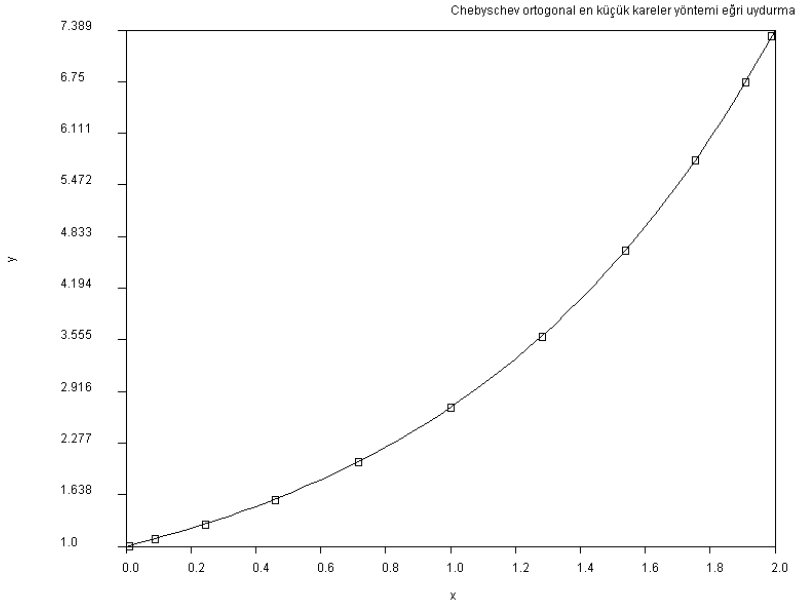
```
import java.io.*;
//E.K:K: Chebyshev ortogonal polinomlari kullanarak eğri uydurma

class SCO11K
{
        public static double Ti(double x,int equation_ref)
        {//Chebysev fonksiyonu
        double T[]=new double[equation_ref+1];
        T[0]=1.0;
        if(equation_ref>=1) T[1]=x;
        for(int i=2;i<=equation_ref;i++) {T[i]=2.0*x*T[i-1]-T[i-2];
  //System.out.println("T["+i+"]="+T[i]);
  }
        return T[equation_ref];
        }
public static double[] xi(double a,double b,int N)
{
  int nn=N+1;
  double x[]=new double[nn];
  double t[]=new double[nn];
  for(int k=0;k<nn;k++)
   {t[k]=Math.cos((2.0*N+1-2.0*k)*Math.PI/(2.0*N+2.0));
    x[k]=(b-a)/2.0*t[k]+(a+b)/2.0;
   }
  return x;
}

public static double[] Chebyshev(double a,double b,double y[],int N)
{  //a b arası eşit aralıklı noktalar
   int nn=N+1;
   System.out.println("N="+N);
   int k,j;
   double d=Math.PI/(2.0*N+2);
   double x[]=new double[nn];
```

```
    double t[]=new double[nn];
    // -1<=x<=1 arasinda değer dönüşümü uygula
   for(k=0;k<nn;k++)
    {t[k]=Math.cos((2.0*N+1-2.0*k)*Math.PI/(2.0*N+2.0));
     x[k]=(b-a)/2.0*t[k]+(a+b)/2.0;
    }
    double C[]=new double[nn];
                double z=0;
    for(k=0;k<nn;k++)
      { z+=y[k];}
    C[0]=z/(N+1);
    for(j=1;j<=N;j++)
    {z=0;
     for(k=0;k<=N;k++)
     { z+=y[k]*Ti(t[k],j);}
    C[j]=2.0*z/(N+1);
    }
    return C;
}

public static double funcChebyshev(double C[],double a,double b,double x)
{
   double t=2.0*(x-a)/(b-a)-1;
   int n=C.length;
   double ff=0;
   if(n!=0.0)
   {  for(int i=0;i<n;i++)
     {ff+=C[i]*Ti(t,i);}
   }
   return ff;
}

public static double[][]  funcChebyshev(double a,double b,double y[],int polinomkatsayisi,int verisayisi)
 {
  int n=verisayisi+1;
  double z[][]=new double[2][n];
  double C[]=Chebyshev(a,b,y,polinomkatsayisi);
  double dx=(b-a)/(double)verisayisi;
  z[0][0]=a;
  for(int i=0;i<=verisayisi;i++)
    {if(i==0) z[0][i]=a;
     else z[0][i]=z[0][i-1]+dx;
                  z[1][i]=funcChebyshev(C,a,b,z[0][i]);       }
   return z;
 }
  public static void main(String args[]) throws IOException
  {
  double a=0;
  double b=2.0;
  double y[]={1.010230536,1.094577019,1.276664,1.583059208,2.05087687,2.718281828,
          3.60287651,4.66758038,5.787784491,6.750604088,7.314227631};
  double x[]= xi(a,b,10);
  double z[][]=funcChebyshev(a,b,y,10,50);
  //Text.print(Text.T(z),"Chebyschev ortogonal en küçük kareler yöntemi eğri uydurma");
  System.out.println(Matrix.toString(Matrix.T(z)));
  Plot pp=new Plot(x,y);
  pp.setPlabel("Chebyschev ortogonal en küçük kareler yöntemi eğri uydurma");
  pp.setPlotType(0,20);
  pp.addData(z[0],z[1]);
  pp.plot();
  }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO11K
N=4
      1.000000000000000      0.654102661771397
      1.080000000000000      0.935060925987126
      1.160000000000000      1.230908463581520
      1.240000000000000      1.539354374222004
      1.320000000000000      1.858275680855379
      1.400000000000000      2.185717329707832
      1.480000000000000      2.519892190284927
      1.560000000000000      2.859181055371607
      1.640000000000001      3.202132641032199
      1.720000000000001      3.547463586610407
      1.800000000000001      3.894058454729319
      1.880000000000001      4.240969731291400
      1.960000000000001      4.587417825478498
      2.040000000000001      4.932791069751842
      2.120000000000001      5.276645719852034
      2.200000000000001      5.618705954799068
      2.280000000000001      5.958863876892311
      2.360000000000001      6.297179511710510
      2.440000000000001      6.633880808111798
      2.520000000000001      6.969363638233682
      2.600000000000001      7.304191797493054
      2.680000000000002      7.639097004586184
      2.760000000000002      7.974978901488720
      2.840000000000002      8.312905053455697
      2.920000000000002      8.654110949021529
      3.000000000000002      9.000000000000005
      3.080000000000002      9.352143541484297
      3.160000000000002      9.712280831846963
      3.240000000000002     10.082319052739930
      3.320000000000002     10.464333309094520
      3.400000000000002     10.860566629121418
      3.480000000000002     11.273429964310708
      3.560000000000002     11.705502189431837
      3.640000000000002     12.159530102533648
      3.720000000000002     12.638428424944353
      3.800000000000002     13.145279801271545
      3.880000000000003     13.683334799402210
      3.960000000000003     14.256011910502700
      4.040000000000003     14.866897549018752
      4.120000000000003     15.519746052675487
      4.200000000000003     16.218479682477400
      4.280000000000003     16.967188622708374
      4.360000000000003     17.770130980931670
      4.440000000000003     18.631732787989918
      4.520000000000003     19.556587998005150
```

4.600000000000003	20.549458488378760
4.680000000000003	21.615274059791530
4.760000000000003	22.759132436203622
4.840000000000003	23.986299264854573
4.920000000000004	25.302208116263316
5.000000000000004	26.712460484228153

Çıktı 6.2-4 Chebychev ortogonal polinom eğri uydurma, grafik çıktısı



(bolum6_002.jpg,Resimaltı: **Chebychev Ortogonal polinom eğri uydurma grafik çıktısı**)

Bu programda yapacağımız küçük bir değişiklikle, Chebychev polinom programını verilen bir fonksiyona verilen aralıkta Chebychev serisi uydurmada da kullanabiliriz. Örnek olarak yine $f(x)=e^x$ fonksiyonuna 0 ile 2 aralığında Chebychev polinomu uyduralım.

Program 6.2.3 Chebychev ortogonal polinomları ile fonksiyona eğri uydurma

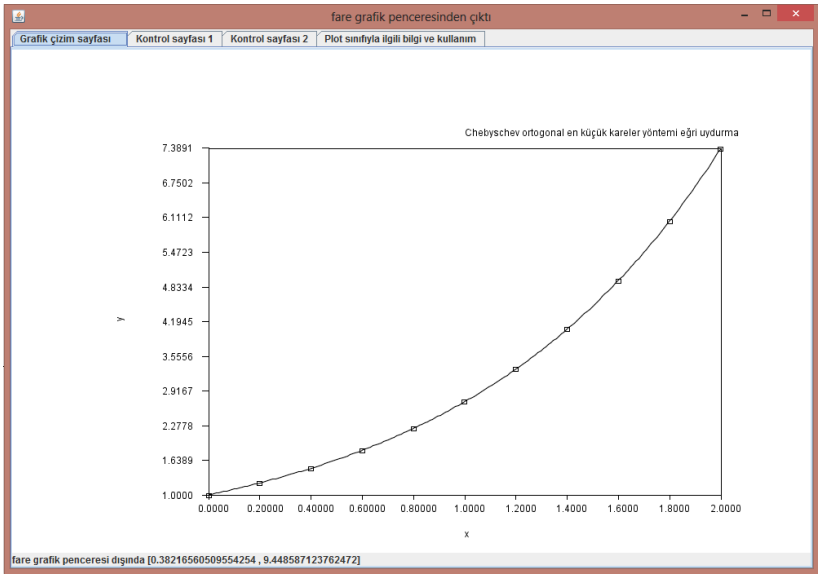
```
import java.io.*;
//E.K:K: Chebyshev ortogonal polinomlari kullanarak eğri uydurma

class f1 extends f_x
{
public double func(double x)
{
//çözümü istenen fonksiyon
double ff;
ff=Math.exp(x);
return ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

class SCO11I
{
        public static double Ti(double x,int equation_ref)
        {//Chebysev fonksiyonu
```

```
            double T[]=new double[equation_ref+1];
            T[0]=1.0;
            if(equation_ref>=1) T[1]=x;
            for(int i=2;i<=equation_ref;i++) {T[i]=2.0*x*T[i-1]-T[i-2];
   //System.out.println("T["+i+"]="+T[i]);
   }
            return T[equation_ref];
            }
public static double[] xi(double a,double b,int N)
{
  int nn=N+1;
  double x[]=new double[nn];
  double t[]=new double[nn];
  for(int k=0;k<nn;k++)
   {t[k]=Math.cos((2.0*N+1-2.0*k)*Math.PI/(2.0*N+2.0));
    x[k]=(b-a)/2.0*t[k]+(a+b)/2.0;
   }
  return x;
}

public static double[] Chebyshev(double a,double b,f_x f,int N)
{  //a b arası eşit aralıklı noktalar
   int nn=N+1;
   System.out.println("N="+N);
   int k,j;
   double d=Math.PI/(2.0*N+2);
   double x[]=new double[nn];
   double t[]=new double[nn];
   double y[]=new double[nn];
    // -1<=x<=1 arasinda değer dönüşümü uygula
   for(k=0;k<nn;k++)
    {t[k]=Math.cos((2.0*N+1-2.0*k)*Math.PI/(2.0*N+2.0));
     x[k]=(b-a)/2.0*t[k]+(a+b)/2.0;
     y[k]=f.func(x[k]);
     //System.out.println("t"+k+"="+t[k]+"x"+k+"="+x[k]+"y"+k+"="+y[k]);
    }
    double C[]=new double[nn];
            double z=0;
    for(k=0;k<nn;k++)
      { z+=y[k];}
    C[0]=z/(N+1);
    //System.out.println("C"+0+"="+C[0]);
    for(j=1;j<=N;j++)
    {z=0;
     for(k=0;k<=N;k++)
     { z+=y[k]*Ti(t[k],j);}
    C[j]=2.0*z/(N+1);
    }
    return C;
```

```
}

public static double funcChebyshev(double C[],double a,double b,double x)
{
   double t=2.0*(x-a)/(b-a)-1;
   //System.out.println("t="+t);
   int n=C.length;
   double ff=0;
   if(n!=0.0)
   {  for(int i=0;i<n;i++)
     {ff+=C[i]*Ti(t,i);}
   }
   return ff;
}

public static double[][]  funcChebyshev(double a,double b,f_x f,int polinomkatsayisi,int verisayisi)
 {
  int n=verisayisi+1;
  double z[][]=new double[2][n];
  double C[]=Chebyshev(a,b,f,polinomkatsayisi);
  double dx=(b-a)/(double)verisayisi;
  z[0][0]=a;
  for(int i=0;i<=verisayisi;i++)
    {if(i==0) z[0][i]=a;
     else z[0][i]=z[0][i-1]+dx;
               z[1][i]=funcChebyshev(C,a,b,z[0][i]);
    //System.out.println("dx="+dx+"i = "+i+"x="+z[0][i]+"y="+z[1][i]);
    }
   return z;
 }

  public static void main(String args[]) throws IOException
  {
  f1 ff=new f1();
  double a=0;
  double b=2.0;
  double z[][]=funcChebyshev(a,b,ff,20,10);
  Text.print(Text.T(z),"Chebyschev ortogonal en küçük kareler yöntemi eğri uydurma");
  System.out.println("dx=\n"+Matrix.toStringT(xi(a,b,10)));
  }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO11I
N=20
      0.000000000000000      0.999999999999992
      0.200000000000000      1.221402758160167
      0.400000000000000      1.491824697641268
      0.600000000000000      1.822118800390509
```

```
      0.800000000000000      2.225540928492473
      1.000000000000000      2.718281828459038
      1.200000000000000      3.320116922736554
      1.400000000000000      4.055199966844676
      1.600000000000000      4.953032424395110
      1.800000000000000      6.049647464412941
      2.000000000000000      7.389056098930627

dx=
      0.010178558119067
      0.090368004645482
      0.244250425645742
      0.459359182544403
      0.718267443158570
      1.000000000000000
      1.281732556841430
      1.540640817455598
      1.755749574354258
      1.909631995354518
      1.989821441880933
```



(bolum6_003.jpg,Resimaltı: **Chebychev Ortogonal polinom eğri uydurma grafik çıktısı**)

6.3 Genelleştirilmiş en küçük kareler metodu

Bölüm 6-1 de en küçük kareler metodunun genel hali $f(x)=\sum_{j=0}^{m} a_j^{(m)}\phi_j(x)$ fonksiyonu için verilmişti.

$$\frac{\partial H(a_0^{(m)},....,a_m^{(m)})}{\partial a_k^{(m)}} = 2\sum_{i=1}^{n} w(x_i)\left[f_i - \sum_{j=0}^{m} a_j^{(m)}\phi_j(x_i)\right]\phi_k(x_i) = 0 \quad k=0,.....,m$$

$$\left\{\sum_{j=0}^{m} w(x_i)\phi_j(x_i)\phi_k(x_i)\right\}\left[a_j^{(m)}\right] = \left[\sum_{i=1}^{n} w(x_i)\phi_k(x_i) f_i\right] \quad k=0,.....,m$$

ağırlık katsayısı $w(x_i)$ =1 olarak seçilirse genel en küçük kareler metodu

$$\begin{Bmatrix} \sum_{i=1}^{n}\phi_0^2(x_i) & \sum_{i=1}^{n}\phi_0(x_i)\phi_1(x_i) & \sum_{i=1}^{n}\phi_0(x_i)\phi_2(x_i) & .... & \sum_{i=1}^{n}\phi_0(x_i)\phi_m(x_i) \\ \sum_{i=1}^{n}\phi_0(x_i)\phi_1(x_i) & \sum_{i=1}^{n}\phi_1^2(x_i) & \sum_{i=1}^{n}\phi_1(x_i)\phi_2(x_i) & .... & \sum_{i=1}^{n}\phi_1(x_i)\phi_m(x_i) \\ \sum_{i=1}^{n}\phi_0(x_i)\phi_2(x_i) & \sum_{i=1}^{n}\phi_1(x_i)\phi_2(x_i) & \sum_{i=1}^{n}\phi_2^2(x_i) & .... & \sum_{i=1}^{n}\phi_2(x_i)\phi_m(x_i) \\ .... & .... & .... & .... & .... \\ \sum_{i=1}^{n}\phi_0(x_i)\phi_m(x_i) & \sum_{i=1}^{n}\phi_1(x_i)\phi_m(x_i) & \sum_{i=1}^{n}\phi_2(x_i)\phi_m(x_i) & .... & \sum_{i=1}^{n}\phi_m^2(x_i) \end{Bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ .... \\ a_m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n}\phi_0(x_i)f(x_i) \\ \sum_{i=1}^{n}\phi_1(x_i)f(x_i) \\ \sum_{i=1}^{n}\phi_2(x_i)f(x_i) \\ .... \\ \sum_{i=1}^{n}\phi_m(x_i)f(x_i) \end{bmatrix}$$

formunu alır. Genel en küçük kareler yöntemi bazı problemler için özel polinom gibi önceden tam olarak bilinen fonksiyonlara göre daha problem spesifik fonksiyonların tanımlanmasına olanak verir. Örneğin gaz R142B gazı yoğunluk denklemi kaynaklarda

$$\rho'/\rho_c = \sum_{i=-10}^{13}(E_{i+11})\tau^{i/3} + E_{25}\ln(\theta)$$

$\tau = (T_c - T)/T_c$ ve $\theta = T/T_c$ ρ' kaynayan sıvının yoğunluğu; ρ_c kritik yoğunluk; T sıcaklık; T_c kritik sıcaklık; şeklinde verilmiştir. Görüldüğü gibi böyle bir denklem standart polinom tanımına uymamaktadır, ancak tüm katsayıları lineer olduğundan rahatlıkla en küçük kareler yöntemi kullanılarak katsayıları hesaplanabilir. Program 6.3-1 de Genel en küçük kareler metodu tanımlanmıştır.

Program 6.3.1 Genel en küçük kareler metodu ile eğri uydurma

```
import java.io.*;
import java.util.*;
//prototip fonksiyon sınıfı
abstract class f_xr
{
 abstract double func(double x,int equation_ref);
}

class fa extends f_xr
{
            double func(double t,int equation_ref)
            {
  double y=1.0;
  double T=t+273.15;
  double Tc=374.23;
  double tr=T/Tc;
  double xx=1.0;
   if(equation_ref==0) xx=1.0;
   else if(equation_ref==1) xx=Math.pow((1.0-tr),1.0/3.0);
   else if(equation_ref==2) xx=Math.pow((1.0-tr),2.0/3.0);
   else if(equation_ref==3) xx=(1.0-tr);
   else if(equation_ref==4) xx=Math.pow((1.0-tr),4.0/3.0);
 return xx;
            }
}

class SCO11G1
{
//Genel en küçük kareler metodu en kucuk kareler metodu

 public static double[] pivotlugauss(double a[][],double b[])
 { //kısmi pivotlu gauss eleme yöntemi
```

```
 int n=b.length;
 double x[]=new double[n];
 double carpan=0;
 double toplam=0;
 double buyuk;
 double dummy=0;
 //gauss eleme
 int i,j,k,p,ii,jj;
 for(k=0;k<(n-1);k++)
 { //pivotlama
              p=k;
              buyuk=Math.abs(a[k][k]);
              for(ii=k+1;ii<n;ii++)
              {  dummy=Math.abs(a[ii][k]);
                if(dummy > buyuk) {buyuk=dummy;p=ii;}
              }
              if(p!=k)
              { for(jj=k;jj<n;jj++)
                { dummy=a[p][jj];
                  a[p][jj]=a[k][jj];
                  a[k][jj]=dummy;
                }
                dummy=b[p];
                b[p]=b[k];
                b[k]=dummy;
              }
              //gauss elemeyi çözme
  for(i=k+1;i<n;i++)
  {  carpan=a[i][k]/a[k][k];
    a[i][k]=0;
    for(j=k+1;j<n;j++)
    {   a[i][j]-=carpan*a[k][j]; }
       b[i]   =b[i]   -carpan*b[k];
              }
 }
 //geriye doğru yerine koyma
 x[n-1]=b[n-1]/a[n-1][n-1];
 for(i=n-2;i>=0;i--)
 {
  toplam=0;
  for(j=i+1;j<n;j++)
  {  toplam+=a[i][j]*x[j];}
  x[i]=(b[i]-toplam)/a[i][i];
 }
 return x;
}

public static double[] EKKgenel(double xi[],double yi[],int n)
{
fa f=new fa();
int l=xi.length;
int i,j,k;
int np1=n+1;
double A[][];
A=new double[np1][np1];
double B[];
B=new double[np1];
double X[];
X=new double[np1];
for(i=0;i<n+1;i++)
 {  B[i]=0;
              for(j=0;j<n+1;j++)
```

```
    {
                A[i][j]=0.0;
    for(k=0;k<l;k++) A[i][j]+=f.func(xi[k],i)*f.func(xi[k],j);
    }
    for(k=0;k<l;k++) B[i]+= f.func(xi[k],i)*yi[k];
 }
//System.out.println("A = \n"+Matrix.toString(A));
//System.out.println("B = \n"+Matrix.toStringT(B));
X=pivotlugauss(A,B);
//X=B/A;
double max=0;
for(i=0;i<n+1;i++)
 if(Math.abs(X[i]) > max) max = Math.abs(X[i]);
for(i=0;i<n+1;i++)
 if((Math.abs(X[i]/max) > 0) && (Math.abs(X[i]/max) < 1.0e-100)) X[i]=0;
return X;
}

public static double funcEKKgenel(double e[],double x)
{
// this function calculates the value of
// least square curve fitting function
fa f=new fa();
int n=e.length;
double ff=0;
if(n!=0.0)
 {
 for(int i=n-1;i>=0;i--)
  {ff+=e[i]*f.func(x,i);}
 }
return ff;
}

public static double hata(double x[],double y[],double e[])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
 double total=0;
 for(k=0;k<n;k++)
 {
 total+=(y[k]-funcEKKgenel(e,x[k]))*(y[k]-funcEKKgenel(e,x[k]));
 }
total=Math.sqrt(total);
return total;
}

 public static double[][]  funcEKKgenel(double xi[],double yi[],int polinomkatsayisi,int aradegersayisi)
 {
              //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
              // ara değer sayısı 2 dir
               int n=xi.length;
              int nn=(n-1)*(aradegersayisi+1)+1;
              double z[][]=new double[2][nn];
   double E[]=EKKgenel(xi,yi,polinomkatsayisi);
   double dx=0;
   int k=0;
   int i;
   for(i=0;i<(n-1);i++)
     {z[0][k]=xi[i];z[1][k]=funcEKKgenel(E,z[0][k]);k++;
     for(int j=0;j<aradegersayisi;j++)
     {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
```

```
    z[0][k]=z[0][k-1]+dx;z[1][k]=funcEKKgenel(E,z[0][k]);k++;}
    }
    z[0][k]=xi[i];z[1][k]=funcEKKgenel(E,z[0][k]);
  return z;
}

 public static void main(String args[]) throws IOException
 {
 //çok büyük veri setlerinde veriyi dosyadan okumamız gerekir
 double c[][]=Text.readDouble("R123trol.txt");
 double b[]=EKKgenel(c[0],c[1],4);
 System.out.println("Genel EKK Denklem katsayıları : \n"+Matrix.toStringT(b));
 double z[][]=funcEKKgenel(c[0],c[1],4,4);
 System.out.println("Genel E.K.K. çözüm seti \n"+Matrix.toStringT(z));
 Plot pp=new Plot(c[0],c[1]);
 pp.setPlabel("Genel En küçük kareler yöntemi ile R134a sıcaklık - doyma sıvı yoğunluğu eğrisi uydurma");
 pp.setXlabel("T derece C");
 pp.setYlabel("ro sıvı kg/m^3");
 pp.addData(z[0],z[1]);
 pp.plot();
 }}
```

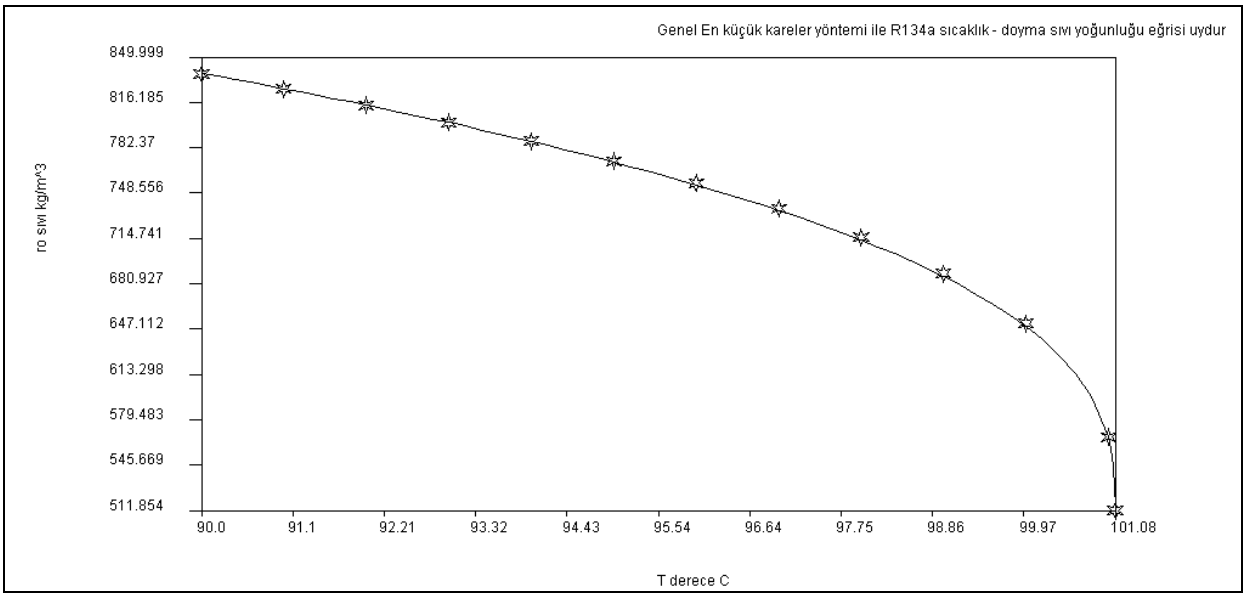
Genel en küçük kareler metodu ile eğri uydurma girdi R123trol.txt dosyası verisi (çıktıyla karşılaştırma amacıyla verilmiştir)

-100.0	1580.5	-59	1468	-18	1349.9	23	1213.3	64	1030.9
-99	1577.8	-58	1465.2	-17	1346.9	24	1209.6	65	1025.3
-98	1575	-57	1462.4	-16	1343.8	25	1205.9	66	1019.6
-97	1572.3	-56	1459.6	-15	1340.8	26	1202.1	67	1013.8
-96	1569.5	-55	1456.9	-14	1337.7	27	1198.3	68	1008
-95	1566.8	-54	1454.1	-13	1334.6	28	1194.4	69	1002
-94	1564.1	-53	1451.3	-12	1331.5	29	1190.6	70	995.9
-93	1561.3	-52	1448.5	-11	1328.4	30	1186.7	71	989.7
-92	1558.6	-51	1445.7	-10	1325.3	31	1182.8	72	983.4
-91	1555.8	-50	1442.9	-9	1322.1	32	1178.8	73	977
-90	1553.1	-49	1440.1	-8	1319	33	1174.9	74	970.4
-89	1550.4	-48	1437.3	-7	1315.8	34	1170.8	75	963.7
-88	1547.6	-47	1434.5	-6	1312.6	35	1166.8	76	956.9
-87	1544.9	-46	1431.6	-5	1309.4	36	1162.7	77	949.9
-86	1542.1	-45	1428.8	-4	1306.2	37	1158.6	78	942.7
-85	1539.4	-44	1426	-3	1303	38	1154.5	79	935.4
-84	1536.7	-43	1423.2	-2	1299.8	39	1150.3	80	927.8
-83	1533.9	-42	1420.3	-1	1296.5	40	1146.1	81	920.1
-82	1531.2	-41	1417.5	0	1293.3	41	1141.9	82	912.1
-81	1528.5	-40	1414.6	1	1290	42	1137.6	83	903.9
-80	1525.7	-39	1411.8	2	1286.7	43	1133.3	84	895.5
-79	1523	-38	1408.9	3	1283.4	44	1128.9	85	886.7
-78	1520.2	-37	1406	4	1280.1	45	1124.5	86	877.6
-77	1517.5	-36	1403.1	5	1276.7	46	1120	87	868.2
-76	1514.8	-35	1400.2	6	1273.4	47	1115.6	88	858.4
-75	1512	-34	1397.4	7	1270	48	1111	89	848.1
-74	1509.3	-33	1394.5	8	1266.6	49	1106.4	90	837.3
-73	1506.5	-32	1391.5	9	1263.2	50	1101.8	91	826
-72	1503.8	-31	1388.6	10	1259.8	51	1097.1	92	814
-71	1501	-30	1385.7	11	1256.3	52	1092.4	93	801.1
-70	1498.3	-29	1382.8	12	1252.9	53	1087.6	94	787.4
-69	1495.5	-28	1379.8	13	1249.4	54	1082.8	95	772.3
-68	1492.8	-27	1376.9	14	1245.9	55	1077.9	96	755.8
-67	1490	-26	1373.9	15	1242.3	56	1072.9	97	737.1
-66	1487.3	-25	1371	16	1238.8	57	1067.9	98	715.4
-65	1484.5	-24	1368	17	1235.2	58	1062.8	99	688.6
-64	1481.8	-23	1365	18	1231.6	59	1057.7	100	651.4

-63	1479	-22	1362	19	1228	60	1052.5	101	566.4
-62	1476.3	-21	1359	20	1224.4	61	1047.2	101.08	511.8544781
-61	1473.5	-20	1356	21	1220.7	62	1041.8		
-60	1470.7	-19	1353	22	1217	63	1036.4		

Genel en küçük kareler metodu ile eğri uydurma çıktı verisi ve eğrikatsayıları (girdi ile karşılaştırma amacıyla verilmiştir)

	-71. 1501.7299347279	-27. 1377.2622706812	17. 1234.4035036263	61. 1047.4818802714	
Genel EKK Denklem	-70. 1499.0287268081	-26. 1374.2685648970	18. 1230.8097052146	62. 1042.1737623283	
katsayıları :	-69. 1496.3226762626	-25. 1371.2657339058	19. 1227.1953534784	63. 1036.7890891627	
	-68. 1493.6117182465	-24. 1368.2536313807	20. 1223.5599890727	64. 1031.3244464559	
513.679650684301800	-67. 1490.8957868557	-23. 1365.2321078238	21. 1219.9031375392	65. 1025.7761766140	
	-66. 1488.1748151038	-22. 1362.2010104729	22. 1216.2243086195	66. 1020.1403542152	
833.307055283695100	-65. 1485.4487348986	-21. 1359.1601832038	23. 1212.5229955273	67. 1014.4127582035	
	-64. 1482.7174770169	-20. 1356.1094664300	24. 1208.7986741779	68. 1008.5888402882	
966.028185763514100	-63. 1479.9809710797	-19. 1353.0486969980	25. 1205.0508023717	69. 1002.6636888991	
	-62. 1477.2391455262	-18. 1349.9777080786	26. 1201.2788189269	70. 996.6319879133	
1134.347502074785800	-61. 1474.4919275872	-17. 1346.8963290542	27. 1197.4821427598	71. 990.4879692009	
	-60. 1471.7392432573	-16. 1343.8043854011	28. 1193.6601719070	76. 957.8418743543	
819.801514346869600	-59. 1468.9810172669	-15. 1340.7016985676	29. 1189.8122824856	77. 950.8693290442	
	-58. 1466.2171730535	-14. 1337.5880858475	30. 1185.9378275871	78. 943.7254580358	
Genel E.K.K. çözüm seti	-57. 1463.4476327309	-13. 1334.4633602479	31. 1182.0361360980	79. 936.3978754687	
-100. 1578.2165827046	-56. 1460.6723170592	-12. 1331.3273303515	32. 1178.1065114440	80. 928.8727091594	
-99. 1575.6316577780	-55. 1457.8911454130	-11. 1328.1798001743	33. 1174.1482302487	81. 921.1343422374	
-98. 1573.0433953946	-54. 1455.1040357480	-10. 1325.0205690163	34. 1170.1605409015	82. 913.1650944916	
-97. 1570.4517536061	-53. 1452.3109045686	-9. 1321.8494313062	35. 1166.1426620266	83. 904.9448253848	
-96. 1567.8566898712	-52. 1449.5116668921	-8. 1318.6661764403	36. 1162.0937808445	84. 896.4504339617	
-95. 1565.2581610449	-51. 1446.7062362137	-7. 1315.4705886135	37. 1158.0130514173	85. 887.6552210907	
-94. 1562.6561233671	-50. 1443.8945244694	-6. 1312.2624466439	38. 1153.8995927659	86. 878.5280649497	
-93. 1560.0505324507	-49. 1441.0764419982	-5. 1309.0415237890	39. 1149.7524868504	87. 869.0323386222	
-92. 1557.4413432699	-48. 1438.2518975032	-4. 1305.8075875549	40. 1145.5707763987	88. 859.1244643913	
-91. 1554.8285101479	-47. 1435.4207980106	-3. 1302.5603994958	41. 1141.3534625709	89. 848.7519445048	
-90. 1552.2119867440	-46. 1432.5830488290	-2. 1299.2997150058	42. 1137.0995024429	90. 837.8506177163	
-89. 1549.5917260413	-45. 1429.7385535059	-1. 1296.0252831006	43. 1132.8078062932	91. 826.3407360630	
-88. 1546.9676803332	-44. 1426.8872137835	0. 1292.7368461894	44. 1128.4772346732	92. 814.1211798647	
-87. 1544.3398012100	-43. 1424.0289295534	1. 1289.4341398364	45. 1124.1065952386	93. 801.0606100850	
-86. 1541.7080395452	-42. 1421.1635988090	2. 1286.1168925114	46. 1119.6946393206	94. 786.9833237755	
-85. 1539.0723454810	-41. 1418.2911175969	3. 1282.7848253285	47. 1115.2400582064	95. 771.6453646423	
-84. 1536.4326684140	-40. 1415.4113799668	4. 1279.4376517725	48. 1110.7414791030	96. 754.6912462091	
-83. 1533.7889569805	-39. 1412.5242779197	5. 1276.0750774123	49. 1106.1974607465	97. 735.5679238476	
-82. 1531.1411590409	-38. 1409.6297013536	6. 1272.6967996005	50. 1101.6064886222	98. 713.3300508506	
-81. 1528.4892216640	-37. 1406.7275380087	7. 1269.3025071581	51. 1096.9669697491	99. 686.1025628037	
-80. 1525.8330911115	-36. 1403.8176734097	8. 1265.8918800441	52. 1092.2772269828	100. 648.9650044718	
-79. 1523.1727128208	-35. 1400.8999908070	9. 1262.4645890083	53. 1087.5354927791	101. 566.7273283809	
-78. 1520.5080313887	-34. 1397.9743711146	10. 1259.0202952267	54. 1082.7399023564	101.08 513.6836557615	
-77. 1517.8389905537	-33. 1395.0406928475	11. 1255.5586499184	55. 1077.8884861845		
-76. 1515.1655331784	-32. 1392.0988320559	12. 1252.0792939427	56. 1072.9791617161		
-75. 1512.4876012310	-31. 1389.1486622569	13. 1248.5818573756	57. 1068.0097242686		
-74. 1509.8051357671	-30. 1386.1900543648	14. 1245.0659590632	58. 1062.9778369448		
-73. 1507.1180769099	-29. 1383.2228766182	15. 1241.5312061522	59. 1057.8810194680		
-72. 1504.4263638304	-28. 1380.2469945043	16. 1237.9771935947	60. 1052.7166357857		



(bolum6_004.jpg,Resimaltı: **Genel en küçük kareler metodu ile eğri uydurma, grafik çıktısı**)

Burada gerçek veri ile çözülen gerçek bir en küçük kareler program uygulaması görülmektedir. Eğrinin uyma miktarlarını görebilmek için veriler aynı noktalarda uydurduğumuz eğriden elde edilen değerler ve orijinal veri de listelenmiştir. Bir verinin ne kadar iyi uyduğu kullanıcının gereksinmelerine bağlıdır. Gerçek verilerle yapılan herhangi bir eğri uydurma formülü kabul edilmeden önce detaylı incelenmesi gerekir.

6.4 Genel en küçük kareler metodu ile birden fazla değişkenli fonksiyonların eğri uydurması

Çok değişkenli bir fonksiyonu en küçük kareler metoduna uydurmak temel olarak tek değişkenli fonksiyonu uydurma ile aynıdır, sadece kullanılan fonksiyonun tanımını çok boyutlu olarak yapmamız gerekir.

$f(x_0, x_1,..., x_{n-1}, x_n) = \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_0, x_1,..., x_{n-1}, x_n)$ j inci derece polinomu verilmiş olsun. Bu polinoma x_0, x_1, x_2,..., x_n ,f_i, i=0...n verisini uydurmak istiyoruz. Bu veri seti için en uygun $a_j^{(m)}$ değerlerini bulmak istiyoruz. Bunun için $H(a_0^{(m)},....,a_m^{(m)}) = \sum_{i=1}^{n} w(x_0, x_1,..., x_{n-1}, x_n)\left[f_i - \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_0, x_1,..., x_{n-1}, x_n) \right]^2$ fonksiyonunun minimum değerini bulmamız gerekir. Fonksiyondaki w $(x_0, x_1,..., x_{n-1}, x_n)$ ağırlık fonksiyonu adını alır ve $w(x_0, x_1,..., x_{n-1}, x_n) \geq 0 \quad i = 1,....,n$ olmalıdır. Fonksiyonun minimum noktası türevinin 0 a eşit olduğu nokta olacaktır.

$$\frac{\partial H(a_0^{(m)},....,a_m^{(m)})}{\partial a_k^{(m)}} = 2\sum_{i=1}^{n} w(x_0, x_1,..., x_{n-1}, x_n)\left[f_i - \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_0, x_1,..., x_{n-1}, x_n) \right]\phi_k(x_0, x_1,..., x_{n-1}, x_n) = 0 \quad k = 0,.....,m$$

$$\left\{ \sum_{j=0}^{m} w(x_0, x_1,..., x_{n-1}, x_n)\phi_j(x_0, x_1,..., x_{n-1}, x_n)\phi_k(x_0, x_1,..., x_{n-1}, x_n) \right\}\left[a_j^{(m)}\right] = \left[\sum_{i=1}^{n} w(x_0, x_1,..., x_{n-1}, x_n)\phi_k(x_0, x_1,..., x_{n-1}, x_n) f_i \right] \quad k = 0,..,m$$

Ağırlık fonksiyonunu 1 olarak alalım. Bu durumda temel denklem

$$\left\{ \sum_{j=0}^{m} \phi_j(x_0, x_1,..., x_{n-1}, x_n)\phi_k(x_0, x_1,..., x_{n-1}, x_n) \right\}\left[a_j^{(m)}\right] = \left[\sum_{i=1}^{n} \phi_k(x_0, x_1,..., x_{n-1}, x_n) f_i \right] \quad k = 0,..,m$$

şeklini alır. Örnek genel fonksiyon olarak iki boyutlu 4üncü derecen polinom denklemini alalım

$f(x,y) = a_0 + a_1x + a_2y + a_3x^2 + a_4y^2 + a_5xy + a_6x^3 + a_7y^3 + a_8x^2y + a_9xy^2 + a_{10}x^4 + a_{11}y^4 + a_{12}x^2y^2 +$

$a_{13}x^3y + a_{14}xy^2$

Program 6.4.1 Genel en küçük kareler metodu ile ikinci derece polinom eğri uydurma

```
 import java.io.*;
import java.util.*;
import javax.swing.*;
import java.awt.event.*;

//yüzey eğri uydurma
//f(x,y)=a0+a1*x+a2*y+a3*x^2+a4*y^2+a5*xy+a6*x^3+a7*y^3+a8*x^2y
//     +a9*xy^2+a10*x^4+a11*y^4+a12*x^2y^2+a13x^3y+a14*xy^3
abstract class f_xir
{
  abstract double func(double x[],int equation_ref);
}

class fa extends f_xir
{
                double func(double x[],int i)
                {//genel bir yüzey modeli
   double xx=0.0;
   //density function
      if(i==0)      {xx=1.0;         }
   else if (i==1) {xx=x[0];        }
   else if (i==2) {xx=x[1];        }
   else if (i==3) {xx=x[0]*x[0];    }
   else if (i==4) {xx=x[1]*x[1];    }
   else if (i==5) {xx=x[0]*x[1];    }
   else if (i==6) {xx=x[0]*x[0]*x[0];}
   else if (i==7) {xx=x[1]*x[1]*x[1];}
   else if (i==8) {xx=x[0]*x[0]*x[1];}
   else if (i==9) {xx=x[0]*x[1]*x[1];}
   else if (i==10){xx=x[0]*x[0]*x[0]*x[0];}
   else if (i==11){xx=x[1]*x[1]*x[1]*x[1];}
   else if (i==12){xx=x[0]*x[0]*x[1]*x[1];}
   else if (i==13){xx=x[0]*x[0]*x[0]*x[1];}
   else if (i==14){xx=x[0]*x[1]*x[1]*x[1];}
   return xx;
                }
}

class SCO11G5
{
//  Genel en küçük kareler metodu en kucuk kareler metodu
                public static double[][] Transpose(double [][] left)
   { //transpose matrix (if A=a(i,j) Transpose(A)=a(j,i)
   int i,j;
   int n=left.length;
   int m=left[0].length;
   double b[][];
   b=new double[m][n];
   for(i=0;i<n;i++)
    {for(j=0;j<m;j++)  {b[j][i]=left[i][j];} }
   return b;
   }

  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
```

```
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
                p=k;
                buyuk=Math.abs(a[k][k]);
                for(ii=k+1;ii<n;ii++)
                {  dummy=Math.abs(a[ii][k]);
                  if(dummy > buyuk) {buyuk=dummy;p=ii;}
                }
                if(p!=k)
                { for(jj=k;jj<n;jj++)
                  { dummy=a[p][jj];
                    a[p][jj]=a[k][jj];
                    a[k][jj]=dummy;
                  }
                  dummy=b[p];
                  b[p]=b[k];
                  b[k]=dummy;
                }
                //gauss elemeyi çözme
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
           b[i]   =b[i]   -carpan*b[k];
                }
  }
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double[] EKKgenel(double c[][],int n)
{
int n1=c.length;
  int n2=c[0].length-1;
  System.out.println("n1="+n1+"n2="+n2);
  double xi[][]=new double[n1][n2];
  double yi[]=new double[n1];
  for(int i=0;i<n1;i++){for(int j=0;j<n2;j++){xi[i][j]=c[i][j];};yi[i]=c[i][n2];}
  return EKKgenel(xi,yi,n);
}

public static double[] EKKgenel(double xi[][],double yi[],int n)
{ // n boyutlu yüzey eğri uydurma
  // yüzey EKK katsayılarını hesaplar
fa f=new fa();
int l=xi.length;
int i,j,k;
```

```
int np1=n+1;
double A[][];
A=new double[np1][np1];
double B[];
B=new double[np1];
double X[];
X=new double[np1];
for(i=0;i<n+1;i++)
 {  B[i]=0;
            for(j=0;j<np1;j++)
   {
            A[i][j]=0.0;
   for(k=0;k<l;k++) A[i][j]+=f.func(xi[k],i)*f.func(xi[k],j);
   }
   for(k=0;k<l;k++) B[i]+= f.func(xi[k],i)*yi[k];
 }
//System.out.println("A = \n"+Matrix.toString(A));
//System.out.println("B = \n"+Matrix.toStringT(B));
X=pivotlugauss(A,B);
//X=B/A;
double max=0;
for(i=0;i<n+1;i++)
 if(Math.abs(X[i]) > max) max = Math.abs(X[i]);
for(i=0;i<n+1;i++)
 if((Math.abs(X[i]/max) > 0) && (Math.abs(X[i]/max) < 1.0e-100)) X[i]=0;
//Text.printT(X);
return X;
}

public static double funcEKKgenel(double e[],double x[])
{
// tek bir nokta için çok boyutlu eğri değeri hesaplar
fa f=new fa();
int n=e.length;
double ff=0;
if(n!=0.0)
 {
 for(int i=n-1;i>=0;i--)
  {ff+=e[i]*f.func(x,i);}
 }
return ff;
}

public static double[] funcEKKgenel(double e[],double xi[][])
{
// tek bir nokta için çok boyutlu eğri değeri hesaplar
fa f=new fa();
int n=e.length;
double ff[]=new double[xi.length];
for(int k=0;k<xi.length;k++)
{
if(n!=0.0)
 {
 for(int i=n-1;i>=0;i--)
  {ff[k]+=e[i]*f.func(xi[k],i);}
 }
}
return ff;
}

public static double[][] cikti(double c[][],int polinomkatsayisi,int aradegersayisi)
{ int n1=c.length;
```

```
  int n2=c[0].length-1;
  System.out.println("n1="+n1+"n2="+n2);
  double xi[][]=new double[n1][n2];
  double yi[]=new double[n1];
  for(int i=0;i<n1;i++){for(int j=0;j<n2;j++){xi[i][j]=c[i][j];};yi[i]=c[i][n2];
  }
  return cikti(xi,yi,polinomkatsayisi,aradegersayisi);
}


public static double[][] cikti(double xi[][],double yi[],int polinomkatsayisi,int aradegersayisi)
{
                int n=xi.length;
                int nk=xi[0].length;
                int nn=(n-1)*(aradegersayisi+1)+1;
    double E[]=EKKgenel(xi,yi,polinomkatsayisi);
    double x[][]=new double[nn][nk];
    double c[][]=new double[nn][nk+1];
    double yy[]=new double[nn];
    double dx[]=new double[nk];
    int k=0;
    int i,j,w;
    for(i=0;i<(n-1);i++)
      {   for(w=0;w<nk;w++){x[k][w]=xi[i][w];dx[w]=(xi[i+1][w]-xi[i][w])/((double)aradegersayisi+1.0);};k++;
         for(j=0;j<aradegersayisi;j++)
         { for(w=0;w<nk;w++){x[k][w]=x[k-1][w]+dx[w];};k++;}
      }
    for(w=0;w<nk;w++){x[k][w]=xi[i][w];}
    yy=funcEKKgenel(E,x);
    for(i=0;i<x.length;i++)
     {for(w=0;w<nk;w++) c[i][w]=x[i][w];
      c[i][nk]=yy[i];
     }
    return c;
}

public static double funcEKKgenel(double e[],double x,double y)
{
// yüzey uydurma fonksiyonu
// tek bir nokta için iki boyutlu eğri değeri hesaplar
double xx[]=new double[2];
xx[0]=x;
xx[1]=y;
return funcEKKgenel(e,xx);
}

public static double hata(double c[][],double e[])
{int n1=c.length;
  int n2=c[0].length-1;
  System.out.println("n1="+n1+"n2="+n2);
  double xi[][]=new double[n1][n2];
  double yi[]=new double[n1];
  for(int i=0;i<n1;i++){for(int j=0;j<n2;j++){xi[i][j]=c[i][j];};yi[i]=c[i][n2];
  }
return hata(xi,yi,e);
}

public static double hata(double x[][],double y[],double e[])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
```

```
 double total=0;
 for(k=0;k<n;k++)
 {
 total+=(y[k]-funcEKKgenel(e,x[k]))*(y[k]-funcEKKgenel(e,x[k]));
 }
total=Math.sqrt(total);
return total;
}

  public static void main(String[] args)
  {
  String s1=JOptionPane.showInputDialog("dosya adı : ");
  double c[][]=Text.readDouble(s1);
  //Text.print(c);
  double b[]=EKKgenel(c,10);
  System.out.println("polinom katsayıları :  \n"+Matrix.toStringT(b));
  double xi[][]=cikti(c,10,2);
  System.out.println("cikti \n"+Matrix.toString(xi));
  System.out.println("hata \n"+Matrix.toString(hata(c,b)));

  }
}
```

Genel en küçük kareler metodu ile ikinci derece polinom eğri uydurma

0	0	1	1.2	0	5.368	2.4	0	22.984	3.6	0	64.216
0	0.4	1.624	1.2	0.4	7.24	2.4	0.4	27.256	3.6	0.4	72.04
0	0.8	2.952	1.2	0.8	10.2	2.4	0.8	33	3.6	0.8	81.72
0	1.2	5.368	1.2	1.2	14.632	2.4	1.2	40.6	3.6	1.2	93.64
0	1.6	9.256	1.2	1.6	20.92	2.4	1.6	50.44	3.6	1.6	108.184
0	2	15	1.2	2	29.448	2.4	2	62.904	3.6	2	125.736
0	2.4	22.984	1.2	2.4	40.6	2.4	2.4	78.376	3.6	2.4	146.68
0	2.8	33.592	1.2	2.8	54.76	2.4	2.8	97.24	3.6	2.8	171.4
0	3.2	47.208	1.2	3.2	72.312	2.4	3.2	119.88	3.6	3.2	200.28
0	3.6	64.216	1.2	3.6	93.64	2.4	3.6	146.68	3.6	3.6	233.704
0.4	0	1.624	1.6	0	9.256	2.8	0	33.592	4	0	85
0.4	0.4	2.536	1.6	0.4	11.8	2.8	0.4	38.92	4	0.4	94.264
0.4	0.8	4.28	1.6	0.8	15.56	2.8	0.8	45.848	4	0.8	105.512
0.4	1.2	7.24	1.6	1.2	20.92	2.8	1.2	54.76	4	1.2	119.128
0.4	1.6	11.8	1.6	1.6	28.264	2.8	1.6	66.04	4	1.6	135.496
0.4	2	18.344	1.6	2	37.976	2.8	2	80.072	4	2	155
0.4	2.4	27.256	1.6	2.4	50.44	2.8	2.4	97.24	4	2.4	178.024
0.4	2.8	38.92	1.6	2.8	66.04	2.8	2.8	117.928	4	2.8	204.952
0.4	3.2	53.72	1.6	3.2	85.16	2.8	3.2	142.52	4	3.2	236.168
0.4	3.6	72.04	1.6	3.6	108.184	2.8	3.6	171.4	4	3.6	272.056
0.8	0	2.952	2	0	15	3.2	0	47.208			
0.8	0.4	4.28	2	0.4	18.344	3.2	0.4	53.72			
0.8	0.8	6.568	2	0.8	23.032	3.2	0.8	61.96			
0.8	1.2	10.2	2	1.2	29.448	3.2	1.2	72.312			
0.8	1.6	15.56	2	1.6	37.976	3.2	1.6	85.16			
0.8	2	23.032	2	2	49	3.2	2	100.888			
0.8	2.4	33	2	2.4	62.904	3.2	2.4	119.88			
0.8	2.8	45.848	2	2.8	80.072	3.2	2.8	142.52			
0.8	3.2	61.96	2	3.2	100.888	3.2	3.2	169.192			
0.8	3.6	81.72	2	3.6	125.736	3.2	3.6	200.28			

Örnek verimiz katsayılar matrisi 1 olacak şekilde oluşturulmuştur. Bu veriye eğri uydurduğumuzda :

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO11G5
```

n1=110n2=2

polinom katsayıları :

0.999999999994438
1.000000000047910
1.000000000000768
0.999999999941774
0.999999999998342
1.000000000001277
1.000000000022649
1.000000000000404
0.999999999999890
0.999999999999775
-0.000000000002770

n1=110n2=2

cikti

0.000000000000000	0.000000000000000	0.999999999994438
0.000000000000000	0.133333333333333	1.153481481475994
0.000000000000000	0.266666666666667	1.356740740735274
0.000000000000000	0.400000000000000	1.623999999994506
0.000000000000000	0.533333333333333	1.969481481475919
0.000000000000000	0.666666666666667	2.407407407401740
0.000000000000000	0.800000000000000	2.951999999994199
0.000000000000000	0.933333333333333	3.617481481475521
0.000000000000000	1.066666666666667	4.418074074067935
0.000000000000000	1.200000000000000	5.367999999993669
0.000000000000000	1.333333333333333	6.481481481474952
0.000000000000000	1.466666666666667	7.772740740734011
0.000000000000000	1.600000000000000	9.255999999993078
0.000000000000000	1.733333333333333	10.945481481474370
0.000000000000000	1.866666666666667	12.855407407400127
0.000000000000000	2.000000000000000	14.999999999992570
0.000000000000000	2.133333333333333	17.393481481473930
0.000000000000000	2.266666666666667	20.050074074066433
0.000000000000000	2.400000000000000	22.983999999992310
0.000000000000000	2.533333333333333	26.209481481473784
0.000000000000000	2.666666666666666	29.740740740733084
0.000000000000000	2.800000000000000	33.591999999992440
0.000000000000000	2.933333333333333	37.777481481474084
0.000000000000000	3.066666666666666	42.311407407400230
0.000000000000000	3.200000000000000	47.207999999993150
0.000000000000000	3.333333333333334	52.481481481475010
0.000000000000000	3.466666666666667	58.146074074068070
0.000000000000000	3.600000000000000	64.215999999994550
0.133333333333333	2.400000000000000	24.268148148146103
0.266666666666667	1.200000000000000	6.514074074077112

……

0.400000000000000	0.000000000000000	1.624000000005665
0.400000000000000	0.133333333333333	1.859259259265062
0.400000000000000	0.266666666666667	2.158518518524403
0.400000000000000	0.400000000000000	2.536000000005916
4.000000000000000	2.400000000000000	178.023999999995600
4.000000000000000	2.533333333333333	186.547259259254700
4.000000000000000	2.666666666666666	195.518518518513820
4.000000000000000	2.800000000000000	204.951999999995170
4.000000000000000	2.933333333333333	214.861925925921000
4.000000000000000	3.066666666666666	225.262518518513600
4.000000000000000	3.200000000000000	236.167999999995100
4.000000000000000	3.333333333333334	247.592592592587720
4.000000000000000	3.466666666666667	259.550518518513740
4.000000000000000	3.600000000000000	272.055999999995300

```
n1=110n2=2
hata
    0.000000000046617
> Terminated with exit code 0.
```

6.5 Newton interpolasyonu

İnterpolasyon proseslerinin en küçük kareler eğri uydurmasından temel farkı interpolasyon değerlerinin veri noktalarından tam olarak geçmesidir. n+1 veri noktasından geçen Newton interpolasyon(aradeğer) formülünü

$f_n(x)=b_0+b_1(x-x_0)+....+b_n(x-x_0)\,(x-x_1)\ldots\,(x-x_n)$ formülünden elde edebiliriz. Katsayıları

$b_0=f(x_0)$
$b_1=f[x_1,x_0]$
$b_2= f[x_2,x_1,x_0]$
…
$b_n= f[x_n,x_{n-1},....,x_0]$ şeklinde yazabiliriz. Burada

$$f[x_i, x_j]=\frac{f(x_i)-f(x_j)}{x_i-x_j}$$

$$f[x_i, x_j, x_k]=\frac{f[x_i, x_j]-f[x_j, x_k]}{x_i-x_k}$$

benzer biçimde n. Dereceden sonlu bölünmüş fark da

$$f[x_n, x_{n-1},...., x_1, x_0]=\frac{f[x_n, x_{n-1},...., x_1, x_0]-f[x_n, x_{n-1},...., x_1, x_0]}{x_n-x_0}$$

şeklinde yazılabilir. Bu denklem Newton'un bölünmüş fark interpolasyon polinomu olarak adlandırılır.
Bir örnek problem verelim : A ilinin nüfüs dağılımı aşağıdaki gibi verilmiştir. Newton interpolasyon metodu ile 1945 yılının nüfusunu hesaplamak istiyelim.

yıl	1940	1950	1960	1970	1980	1990
nüfus	132165	151336	179323	203302	226542	249633

Newton fark tablosu :

xi	yi	f[xi,xj]	f[xi,xj,xk]	f[xi,xj,xk,xl]	f[xi,xj,xk,xl,xn]	f[xi,xj,xk,xl,xn,xm]
1940	132165	1917.1	44.08	-2.137333333	0.067054167	-0.001564333
1950	151336	2798.7	-20.04	0.544833333	-0.0111625	
1960	179323	2397.9	-3.695	0.098333333		
1970	203302	2324	-0.745			
1980	226542	2309.1				
1990	249633					

Şeklinde oluşacaktır. Buradan derecesini arttırarak çözümlere ulaşırsak :

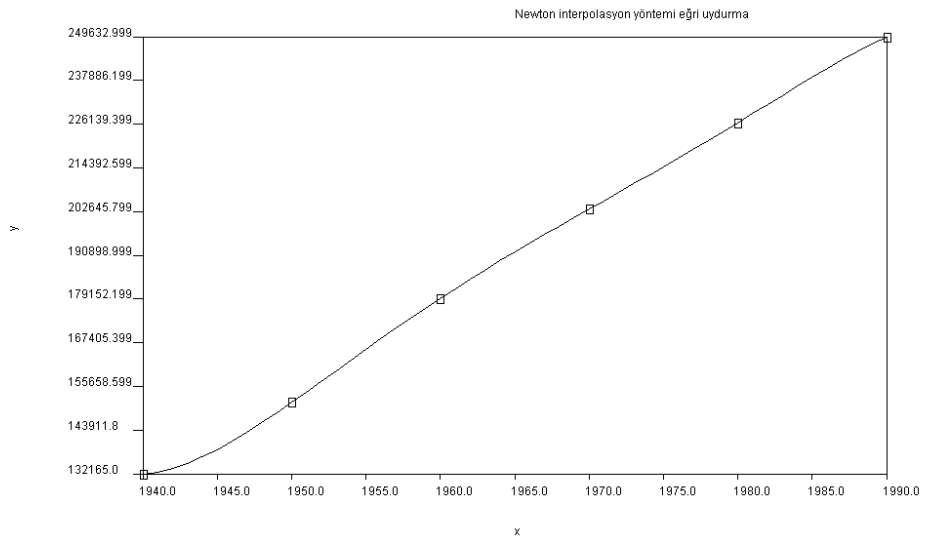
141750.5
140648.5
139847
139218.4
138705.1

Şeklinde çözümleri hesaplayabiliriz. Newton interpolasyon formülünün en önemliavantajı polinom katsayısını arttırmak için son polinom terimini hesaplayarak bir önceki dereceden polinom için hesaplanan değere ekleyebilme olasılığımızın olmasıdır. Şimdi aynı örnek problemi program üzerinden verelim

Program 6.5-1 Newton bölünmüş fark interpolasyonu denklemi

```
import java.io.*;

class SCO11C
{
  public static double[][]  Newton(double xi[],double yi[])
 {
               int n=xi.length;
               double F[][]=new double[n+1][n];
               for(int i=0;i<n;i++)
                 F[i][0]=yi[i];
   for(int i=1;i<n;i++)
   { for(int j=1;j<=i;j++)
      {F[i][j]=(F[i][j-1]-F[i-1][j-1])/(xi[i]-xi[i-j]);}
   }
   for(int i=0;i<n;i++)
   {F[n][i]=xi[i];}
   //System.out.println(Matrix.toString(F));
   return F;
 }

 public static double  funcNewton(double F[][],double x)
 {
               int n=F[0].length;
   double carpim=1.0;
   double toplam=0;
   for(int i=0;i<n;i++)
    {   carpim=F[i][i];
                  for(int j=0;j<=(i-1);j++)
      {carpim*=(x-F[n][j]);}
      toplam+=carpim;
    }
   return toplam;
 }

 public static double[][]  funcNewton(double xi[],double yi[],int aradegersayisi)
{
              //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
              // ara değer sayısı 2 dir
              int n=xi.length;
              int nn=(n-1)*(aradegersayisi+1)+1;
              double z[][]=new double[2][nn];
  double Q[][]=Newton(xi,yi);
  double dx=0;
  int k=0;
  int i;
  for(i=0;i<(n-1);i++)
    {  z[0][k]=xi[i];z[1][k]=funcNewton(Q,z[0][k]);k++;
      for(int j=0;j<aradegersayisi;j++)
        {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
      z[0][k]=z[0][k-1]+dx;z[1][k]=funcNewton(Q,z[0][k]);k++;}
    }
    z[0][k]=xi[i];z[1][k]=funcNewton(Q,z[0][k]);
  return z;
}

 public static void main(String args[]) throws IOException
 {
 double x[]={1940,1950,1960,1970,1980,1990};
 double y[]={132165,151336,179323,203302,226542,249633};
```

```
 double z1[][]=funcNewton(x,y,9);
 System.out.println("Newton interpolasyonu\n"+Matrix.toStringT(z1));
 Text.print(Text.T(z1),"Newton interpolasyonu");
 Plot pp=new Plot(x,y);
 pp.setPlabel("Newton interpolasyon yöntemi eğri uydurma");
 pp.setPlotType(0,20);
 pp.addData(z1[0],z1[1]);
 pp.plot();
 }
}
```



(bolum6_005.jpg,Resimaltı: **Newton bölünmüş fark interpolasyonu grafik çıktısı**)

Newton bölünmüş fark interpolasyonu rakam çıktısı

1940.0 132165.0	1956.0 168453.33574399998	1971.0 205566.7306065	1986.0 241050.42534399996
1941.0 132684.8307565	1957.0 171242.42802049997	1972.0 207832.72300800003	1987.0 243351.74387050004
1942.0 133658.281408	1958.0 173986.55059199999	1973.0 210105.9632045	1988.0 245567.776192
1943.0 135023.3483545	1959.0 176680.98561849998	1974.0 212391.793856	1989.0 247671.68846850004
1944.0 136723.01625599997	1960.0 179323.0	1975.0 214694.7265625	1990.0 249633.0
1945.0 138705.0703125	1961.0 181911.65765650003	1976.0 217018.254144	
1946.0 140921.908544	1962.0 184447.631808	1977.0 219364.6629205	
1947.0 143330.3540705	1963.0 186933.01725449998	1978.0 221734.844992	
1948.0 145891.467392	1964.0 189371.142656	1979.0 224128.1105185	
1949.0 148570.3586685	1965.0 191766.3828125	1980.0 226542.00000000003	
1950.0 151336.0	1966.0 194123.97094400003	1981.0 228972.09655649998	
1951.0 154161.0377065	1967.0 196449.8109705	1982.0 231411.83820800006	
1952.0 157021.60460800002	1968.0 198750.289792	1983.0 233852.33015450006	
1953.0 159897.13230449997	1969.0 201032.0895685	1984.0 236282.15705599997	
1954.0 162770.16345599998	1970.0 203302.0	1985.0 238687.1953125	
1955.0 165626.1640625			

6.6 Lagrange interpolasyonu

Lagrange interplasyon polinomu Newton interpolasyon polinomunun yeniden formülleştirilmesidir. Lagrange interpolasyonunda bölünmüş farkların hesaplanması gerekmez.

$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$ Burada

$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$ şeklindedir. $\prod$ çarpım sembolüdür.

Örneğin doğrusal polinom (n=1) için

$$f_1(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1)$$

ikinci dereceden polinom için ise :

$$f_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2)$$

şeklinde açık formülü yazabiliriz.

Program 6.6-1 Lagrange interpolasyon polinomu

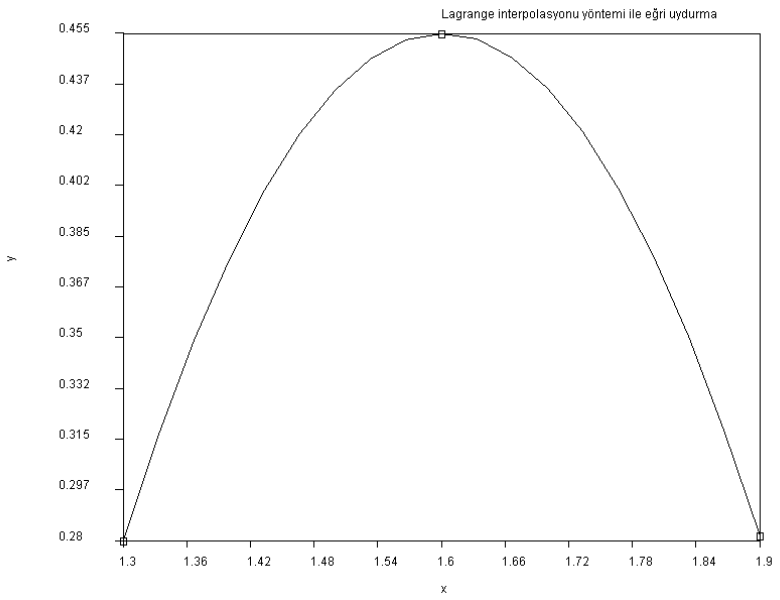
```
import java.io.*;

class SCO11D
{
 public static double[][]  Lagrange(double xi[],double yi[])
 {
              int n=xi.length;
   double L[][]=new double[2][n];
   for(int i=0;i<n;i++)
   {L[0][i]=yi[i];
   for(int j=0;j<n;j++)
   {if(i!=j) L[0][i]/=(xi[i]-xi[j]);}
   }
   for(int i=0;i<n;i++)
   {L[1][i]=xi[i];}
   return L;
 }


 public static double  funcLagrange(double L[][],double x)
 {
              int n=L[0].length;
   double carpim=1.0;
   double toplam=0;
   for(int i=0;i<n;i++)
   {carpim=L[0][i];
   for(int j=0;j<n;j++)
   {if(i!=j) carpim*=(x-L[1][j]);}
   toplam+=carpim;
   }
   return toplam;
 }


 public static double[][]  funcLagrange(double xi[],double yi[],int aradegersayisi)
 {
              //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
              // ara değer sayısı 2 dir
              int n=xi.length;
              int nn=(n-1)*(aradegersayisi+1)+1;
              double z[][]=new double[2][nn];
   double Q[][]=Lagrange(xi,yi);
```

```
    double dx=0;
    int k=0;
    int i;
    for(i=0;i<(n-1);i++)
      {  z[0][k]=xi[i];z[1][k]=funcLagrange(Q,z[0][k]);k++;
        for(int j=0;j<aradegersayisi;j++)
          {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
        z[0][k]=z[0][k-1]+dx;z[1][k]=funcLagrange(Q,z[0][k]);k++;}
      }
      z[0][k]=xi[i];z[1][k]=funcLagrange(Q,z[0][k]);
    return z;
  }

  public static void main(String args[]) throws IOException
  {
  double x[]={1.3,1.6,1.9};
  double y[]={0.280086,0.4554022,0.281818186};

  double z1[][]=funcLagrange(x,y,8);
  System.out.println("Lagrange interpolasyonu\n"+Matrix.toStringT(z1));
   //Text.print(Text.T(z1),"Lagrange interpolasyon  yöntemi ile eğri uydurma");
  Plot pp=new Plot(x,y);
  pp.setPlabel("Lagrange interpolasyonu yöntemi ile eğri uydurma");
  pp.setPlotType(0,20);
  pp.addData(z1[0],z1[1]);
  pp.plot();
  }
}
```

Çıktı 6.6-1 Lagrange interpolasyon polinomu

```
Lagrange interpolasyonu
  1.3000000000   0.2800860000
  1.3333333333   0.3167952180
  1.3666666667   0.3491970259
  1.4000000000   0.3772914238
  1.4333333333   0.4010784116
  1.4666666667   0.4205579894
  1.5000000000   0.4357301571
  1.5333333333   0.4465949148
  1.5666666667   0.4531522624
  1.6000000000   0.4554022000
  1.6333333333   0.4533447275
  1.6666666667   0.4469798450
  1.7000000000   0.4363075524
  1.7333333333   0.4213278498
  1.7666666667   0.4020407372
  1.8000000000   0.3784462144
  1.8333333333   0.3505442817
  1.8666666667   0.3183349389
  1.9000000000   0.2818181860
```

Çıktı 6.6-2 Lagrange interpolasyon polinomu (grafik sonuç)



(bolum6_006.jpg,Resimaltı: **Lagrange interpolasyonu grafik çıktısı**)

6.7 Hermit interpolasyonu

Hermit interpolasyon fonksiyonun kendisinin yanında türevlerin bilgisinide kullanan bir interpolasyon çeşididir.
Genel formül

$H(x) = Q_{0,0}+ Q_{1,1}(x-x_0)+ Q_{2,2}(x-x_0)^2+ Q_{3,3}(x-x_0)^2(x-x_1)+ Q_{4,4}(x-x_0)^2(x-x_1)^2+\ldots+$
$Q_{2n+1,2n+1}(x-x_0)^2(x-x_1)^2\ldots(x-x_{n-1})^2(x-x_n)$

Şeklinde verilebilir. Katsayıların hesaplanması Algoritma 6.6-1 ve Tablo 6.6-1 de açıklanmıştır.

Algoritma 6.7-1 Hermit interpolasyon

GİRDİ $x_0,x_1,\ldots,x_n$ değerleri; $f(x_0), f(x_1), \ldots,f(x_n)$ değerleri; $f'(x_0), f'(x_1), \ldots,f'(x_n)$ değerleri
ÇIKTI : $Q_{0,0},Q_{1,1},\ldots\ldots, Q_{2n+1,2n+1}$ değerleri
$H(x)= Q_{0,0}+ Q_{1,1}(x-x_0)+ Q_{2,2}(x-x_0)^2+ Q_{3,3}(x-x_0)^2(x-x_1)+ Q_{4,4}(x-x_0)^2(x-x_1)^2+\ldots+ Q_{2n+1,2n+1}(x-x_0)^2(x-x_1)^2\ldots(x-x_{n-1})^2(x-x_n)$
Adım 1: for i=0,1,…,n
{
Adım 2: $z_{2i}=x_i$;
$z_{2i+1}=x_i$;
$Q_{2i,0}=f(x_i)$;
$Q_{2i+1,0}=f'(x_i)$;
Adım 3: if $i \neq 0$ then $Q_{2i,j} = \dfrac{Q_{2i,0} - Q_{2i-1,0}}{z_{2i} - z_{2i-1}}$
} Adım 1 in sonu
Adım 4: for i=2,3,…,2n+1
{ for j=2,3,…,i $Q_{i,j} = \dfrac{Q_{i,j-1} - Q_{i-1,j-1}}{z_i - z_{i-j}}$ }

Tablo 6.7-1 Hermit interpolasyon fark denklemlerinin oluşturma sistemi

Z	f(z)	Birinci fark denklemi	İkinci fark denklemi
$Z_0 = x_0$	$f[z_0]=f(x_0)$	$f[z_0, z_1] = f(x_0)$	$f[z_0, z_1, z_2] = \frac{f[z_1, z_2] - f[z_0, z_1]}{z_2 - z_0}$
$Z_1 = x_0$	$f[z_1]=f(x_0)$	$f[z_1, z_2] = \frac{f[z_2] - f[z_1]}{z_2 - z_1}$	$f[z_1, z_2, z_3] = \frac{f[z_2, z_3] - f[z_1, z_2]}{z_2 - z_1}$
$Z_2 = x_1$	$f[z_2]=f(x_1)$	$f[z2, z3] = f(x1)$	$f[z_2, z_3, z_4] = \frac{f[z_3, z_4] - f[z_2, z_3]}{z_4 - z_2}$
$Z_3 = x_1$	$f[z_3]=f(x_1)$	$f[z_3, z_4] = \frac{f[z_4] - f[z_3]}{z_4 - z_3}$	$f[z_3, z_4, z_5] = \frac{f[z_4, z_5] - f[z_3, z_4]}{z_5 - z_4}$
$Z_4 = x_2$	$f[z_4]=f(x_2)$	$f[z4, z5] = f(x2)$	
$Z_5 = x_2$	$f[z_5]=f(x_2)$		

Önce bir örnek probleme bakalım :

k	x_k	$f(x_k)$	$f'(x_k)$
0	1.3	0.6200860	-0.5220232
1	1.6	0.4554022	-05698959
2	1.9	0.2818186	-0.5811571

Tablo 6.7-1 deki fark tablosunu oluşturursak:

1.3	0.620086					
		-0.5220232				
1.3	0.620086		-0.08974267			
		-0.548946		0.06636556		
1.6	0.4554022		-0.069833		0.00266667	
		-0.5698959		0.06796556		-0.00277469
1.6	0.4554022		-0.02905367		0.00100185	
		-0.578612		0.06856667		
1.9	0.2818186		-0.00848367			
		-0.5811571				
1.9	0.2818186					

x=1.5 noktasındaki interpolasyon değeri:

$H_5(1.5)$ =0.620086+(1.5-1.3)*(-0.5220232)+ $(1.5-1.3)^2$*(1.5-1.6)*(0.06636556+

$(1.5-1.3)^2$* $(1.5-1.6)^2$*(0.00266667)+ $1.5-1.3)^2$* $(1.5-1.6)^2$*(1.5-1.9) (-0.00277469)=0.5118277

Şimdi de aynı hesaplamayı yapan örnek programımızı inceleyelim. Yukarıda hesapladığımız nokta program çıktısında koyu olarak belirtilmiştir.

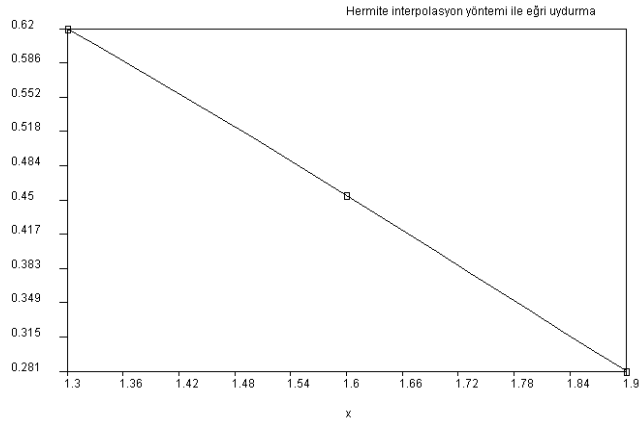
Program 6-7-1 Hermit interpolasyonu

```
import java.io.*;

class SCO11E
{
  public static double [] Hermite(double xi[],double fi[],double dfi[])
  {
  //m polinom derecesi
  int i,j;
  int m=xi.length;
  int n=m-1;
  double Q[][]=new double[2*n+2][2*n+2];
  double QQ[]=new double[2*n+2];
```

```
 double z[]=new double[2*n+2];
 for(i=0;i<=n;i++)
  {
              z[2*i]=xi[i];
              z[2*i+1]=xi[i];
              Q[2*i][0]=fi[i];
              Q[2*i+1][0]=fi[i];
              Q[2*i+1][1]=dfi[i];
              if(i!=0) { Q[2*i][1]=(Q[2*i][0]-Q[2*i-1][0])/(z[2*i]-z[2*i-1]);}
              }
              for(i=2;i<=(2*n+1);i++)
                {for(j=2;j<=i;j++){Q[i][j]=(Q[i][j-1]-Q[i-1][j-1])/(z[i]-z[i-j]);}
                }
              for(i=0;i<=(2*n+1);i++)
               {QQ[i]=Q[i][i];}
              //System.out.println(QQ.length+"\n"+Matrix.toStringT(QQ));
              return QQ;
 }

 public static double funcHermite(double QQ[],double xi[],double x)
 {
               //Hermit polinom
 int i,j;
 int m=QQ.length-1;
 double xx=1;
 double toplam=QQ[0];
 for(i=1;i<m;i+=2)
  {  j=(i-1)/2;
               xx*=(x-xi[j]);
               toplam+=QQ[i]*xx;
               xx*=(x-xi[j]);
               toplam+=QQ[i+1]*xx;
  }
 return toplam;
 }

public static double[][]  funcHermite(double xi[],double yi[],double dyi[],int aradegersayisi)
{
              //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
              // ara değer sayısı 2 dir
              int n=xi.length;
              int nn=(n-1)*(aradegersayisi+1)+1;
              double z[][]=new double[2][nn];
  double Q[]=Hermite(xi,yi,dyi);
  double dx=0;
  int k=0;
  int i;
  for(i=0;i<(n-1);i++)
    {  z[0][k]=xi[i];z[1][k]=funcHermite(Q,xi,z[0][k]);k++;
       for(int j=0;j<aradegersayisi;j++)
         {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
          z[0][k]=z[0][k-1]+dx;z[1][k]=funcHermite(Q,xi,z[0][k]);k++;
         }
    }
    z[0][k]=xi[i];z[1][k]=funcHermite(Q,xi,z[0][k]);
  return z;
}

 public static void main(String args[]) throws IOException
 {
 double x[]={1.3,1.6,1.9};
 double y[]={0.620086,0.4554022,0.281818186};
```

```
  double dy[]={-0.5220232,-0.5698959,-0.5811571};
  double z1[][]=funcHermite(x,y,dy,8);
  Text.print(Text.T(z1),"Hermit interpolasyon yöntemi ile eğri uydurma");
  System.out.println("Hermite interpolasyonu\n"+Matrix.toStringT(z1));
  Plot pp=new Plot(x,y);
  pp.setPlabel("Hermite interpolasyon yöntemi ile eğri uydurma");
  pp.setPlotType(0,20);
  pp.addData(z1[0],z1[1]);
  pp.plot();

  }
}
```

Çıktı 6-7-1 Hermit interpolasyonu grafik çıktısı



(bolum6_007.jpg,Resimaltı: **Hermit interpolasyonu grafik çıktısı**)

Hermite interpolasyonu	
1.3000000000	0.6200860000
1.3333333333	0.6025660584
1.3666666667	0.5848174157
1.4000000000	0.5668545838
1.4333333333	0.5486921534
1.4666666667	0.5303447937
1.5000000000	**0.5118272527**
1.5333333333	0.4931543568
1.5666666667	0.4743410113
1.6000000000	0.4554022000
1.6333333333	0.4363529853
1.6666666667	0.4172085082
1.7000000000	0.3979839884
1.7333333333	0.3786947244
1.7666666667	0.3593560931
1.8000000000	0.3399835500
1.8333333333	0.3205926294
1.8666666667	0.3011989443
1.9000000000	0.2818181860

6.8 Vandermonde interpolasyonu

Eğer interpolasyon polinomu

$$y = c_0 + c_1 x + c_2 x^2 + \ldots + c_N x^N$$

Şeklinde yazılacak olursa ve c_i katsayıları interpolasyon katsayıları ise N veri seti için N'inci dereceden bir polinomu Vandermonde matrisi olarak yazıp katsayıların çözümüne ulaşmak mümkündür.

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^N \\ 1 & x_1 & x_1^2 & \ldots & x_1^N \\ \ldots & \ldots & .. & \ldots & \ldots \\ 1 & x_{N-1} & x_{N-1}^2 & \ldots & x_{N-1}^N \\ 1 & x_N & x_N^2 & & x_N^N \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \ldots \\ c_{N-1} \\ c_N \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \ldots \\ y_{N-1} \\ y_N \end{bmatrix}$$

Bu matris yapısı itibariyle yüksek hata olasılığı içerdiğinde sadece göreceli küçük veri sayıları için anlamlı bir çözüm verebilir. Alttaki programda Vandermonde matrisinin Gauss elimine yöntemiyle çözüldüğü bir versiyonu verilmiştir. Burada Vandermonde

metodunun bir interpolasyon metodu olduğunu, en küçük kareler tipi metod olmadığını yineleyelim. Veri tüm noktalardan geçmektedir.

Program 6.8.1 Gauss eliminasyonuyla çözülmüş Vandermonde matrisi

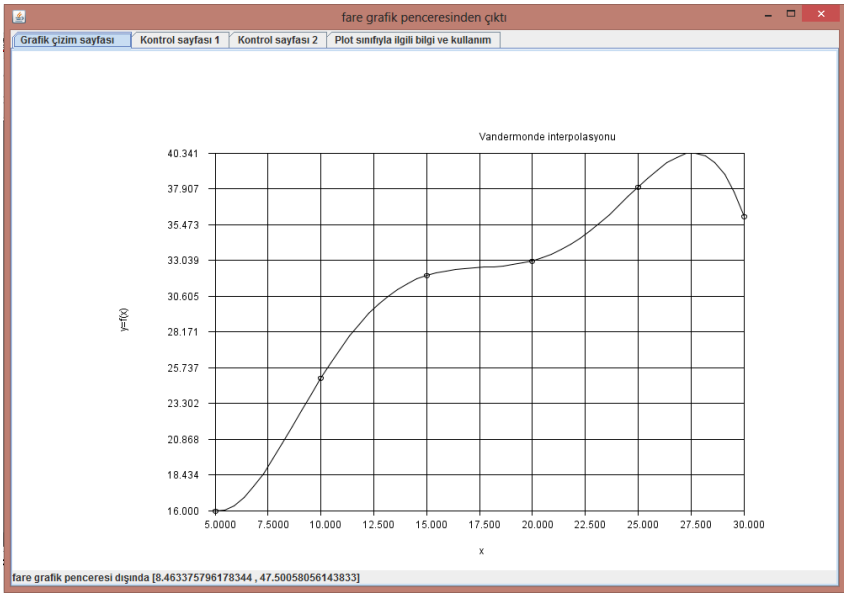
```
import java.io.*;
import java.util.*;
import javax.swing.*;

class Vandermonde
{
//Vandermonde interpolation
  public static double[] gausswithpartialpivot(double a[][],double b[])
  { //Gauss elimination with partial pivoting
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss elimination
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //partial pivoting
              p=k;
              buyuk=Math.abs(a[k][k]);
              for(ii=k+1;ii<n;ii++)
              {  dummy=Math.abs(a[ii][k]);
                 if(dummy > buyuk) {buyuk=dummy;p=ii;}
              }
              if(p!=k)
              { for(jj=k;jj<n;jj++)
                { dummy=a[p][jj];
                  a[p][jj]=a[k][jj];
                  a[k][jj]=dummy;
                }
                dummy=b[p];
                b[p]=b[k];
                b[k]=dummy;
              }
              //
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
           b[i]   =b[i]   -carpan*b[k];
              }
  }
  //backward substitution
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
```

```
}


public static double[] Vandermonde_coef(double xi[],double yi[])
{
int n=xi.length;
fa f=new fa();
int l=xi.length;
int i,j,k;
double A[][];
A=new double[n][n];
double B[];
B=new double[n];
double X[];
X=new double[n];
double cc=1.0;
for(i=0;i<n;i++)
  {  B[i]=yi[i];
     cc=1.0;
              for(j=0;j<n;j++)
     {
              A[i][j]=cc;
              cc*=xi[i];
     }
  }
X=gausswithpartialpivot(A,B);
return X;
}

public static double func(double c[],double x)
{
// this function calculates the value of
// least square curve fitting function
int n=c.length;
double ff=0;
double xx=1.0;
if(n!=0.0)
  { xx=1.0;
  for(int i=0;i<n;i++)
   {ff+=c[i]*xx;
    xx*=x;
   }
  }
return ff;
}

public static double[][]  Vandermonde(double xi[],double yi[],int aradegersayisi)
 {
              //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
              // ara değer sayısı 2 dir
              int n=xi.length;
              int nn=(n-1)*(aradegersayisi+1)+1;
              double z[][]=new double[2][nn];
    double E[]=Vandermonde_coef(xi,yi);
    System.out.println("coefficients :\n"+Matrix.toStringT(E));
    double dx=0;
    int k=0;
    int i;
```

```
    for(i=0;i<(n-1);i++)
      {z[0][k]=xi[i];z[1][k]=func(E,z[0][k]);k++;
      for(int j=0;j<aradegersayisi;j++)
      {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
      z[0][k]=z[0][k-1]+dx;z[1][k]=func(E,z[0][k]);k++;}
      }
      z[0][k]=xi[i];z[1][k]=func(E,z[0][k]);
    return z;
  }

  public static void main(String args[]) throws IOException
  {
  double x[];
  double y[];
  String s1=JOptionPane.showInputDialog("Data file name (input1.txt)");
  // JFileChooser fc=new JFileChooser();
  // if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName();
}
  double a[][]=Text.readDoubleT(s1);
  x=a[0];
  y=a[1];
  double z[][]=Vandermonde(x,y,10);
  System.out.println("Vandermonde interpolation curve fitting\n"+Matrix.toStringT(z));
  Plot pp=new Plot(a[0],a[1]);
  pp.setPlabel("Vandermonde interpolation curve fitting");
  pp.setXlabel("x");
  pp.setYlabel("y=f(x)");
  pp.setPlotType(0,22);
  pp.addData(z[0],z[1]);
  pp.setPlotType(1,0);
  pp.setGrid(1,1);
  pp.plot();
   }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" Vandermonde
katsayisificients :
      58.000000000000256
     -20.883333333333393
       3.561666666666668
      -0.250333333333333
       0.007933333333333
      -0.000093333333333

Vandermonde interpolasyonu
       5.000000000000000      15.999999999999998
       5.454545454545454      16.004588608577380
       5.909090909090908      16.311305114528896
       6.363636363636362      16.863682932735518
       6.818181818181817      17.610558146177183
       7.272727272727271      18.505852183469727
       7.727272727272725      19.508354496401660
       8.181818181818180      20.581505237471248
```

8.636363636363635	21.693177937423368
9.090909090909090	22.815462182786650
9.545454545454545	23.924446293409970
10.000000000000000	24.999999999999840
10.454545454545455	26.025557121656910
10.909090909090910	26.987898243413312
11.363636363636365	27.876933393769410
11.818181818181820	28.685484722230658
12.272727272727275	29.409069176844400
12.727272727272730	30.045681181737205
13.181818181818185	30.595575314651597
13.636363636363640	31.061048984482873
14.090909090909095	31.446225108816172
14.545454545454550	31.756834791463277
15.000000000000000	31.999999999999602
15.454545454545455	32.184016243301530
15.909090909090910	32.318135249082290
16.363636363636363	32.412347641430000
16.818181818181817	32.477165618344145
17.272727272727270	32.523405629272220
17.727272727272723	32.561971052647490
18.181818181818176	32.603634873424880
18.636363636363630	32.658822360618245
19.090909090909083	32.737393744838386
19.545454545454536	32.848426895827740
20.000000000000000	32.999999999999886
20.454545454545453	33.198974237973970
20.909090909090907	33.450776462113880
21.363636363636360	33.759181874064154
21.818181818181813	34.126096702286590
22.272727272727266	34.551340879596694
22.727272727272720	35.032430720703020
23.181818181818173	35.564361599741574
23.636363636363626	36.139390627812986
24.090909090909080	36.746819330521475
24.545454545454533	37.372776325511380
25.000000000000000	38.000000000000230
25.454545454545453	38.607621188318600
25.909090909090907	39.170945849451530
26.363636363636360	39.661237744565140
26.818181818181813	40.045501114553190
27.272727272727266	40.286263357569396
27.727272727272720	40.341357706565990
28.181818181818173	40.163705906824134
28.636363636363626	39.701100893505554
29.090909090909080	38.895989469174080
29.545454545454533	37.685254981341586
30.000000000000000	36.000000000000000



(bolum6_008.jpg,Resimaltı: **Vandermonde interpolasyonu grafik çıktısı**)

Vandermonde interpolasyonu kolaylıkla Lagrange interpolasyonu üzerinden de tanımlanabilir. Matrisin her kolonu Lagrange polinomu değerine eşittir. LAgrange polinomunun

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{N} \frac{x - x_j}{x_i - x_j} = \sum_{k=0}^{N} A_{ik} x^k$$

Formunda olduğunu hatırlayalım. Denklm bir çarpım olduğundan aynı zamanda bir polinoma da eşit olur. Denklemi dirak-delta fonksiyonu cinsinden yazarsak

$$L_i(x_j) = \delta_{ji} = \sum_{k=0}^{N} A_{ik} x^k$$

$$\delta_{ji} = \begin{Bmatrix} 0 \;\; if \;\; j \neq i \\ 1 \;\; if \;\; j \;\; = i \end{Bmatrix}$$

Vandermonde katsayıları cinsinden düşünürsek

$$c_i = \sum_{k=0}^{N} A_{ki} y_k$$

Şeklinde yazılabilir. Bu formun avantajı bize matris sistemi çözmeden katsayıları verebilmesidir.

Program 6.8.1 Lagrange interpolasyon denklemi üzerinden tanımlanan Vandermonde interpolasyonu

```
 import java.io.*;
import java.util.*;
import javax.swing.*;

class Vandermonde1
{
//Vandermonde interpolation
public static double[] Vandermonde_coef(double x[], double y[])
{ // Vandermonde interpolasyon katsayılarını LAgrange interpolasyon formülü
  //üzerinden hesaplar
int n=x.length-1;
double cof[]=new double[n+1];
```

```
int k,j,i;
double phi,ff,b;
double s[]=new double[n+1];
s[n] = -x[0];
for (i=1;i<=n;i++) {
for (j=n-i;j<=(n-1);j++)
s[j] -= x[i]*s[j+1];
s[n] -= x[i];
}
for (j=0;j<=n;j++) {
phi=n+1;
for (k=n;k>=1;k--)
phi=k*s[k]+x[j]*phi;
ff=y[j]/phi;
b=1.0;
for (k=n;k>=0;k--)
{
cof[k] += b*ff;
b=s[k]+x[j]*b;
}
}
return cof;
}

public static double func(double c[],double x)
{
// polinom foksiyonu
int n=c.length;
double ff=0;
double xx=1.0;
if(n!=0.0)
  { xx=1.0;
  for(int i=0;i<n;i++)
   {ff+=c[i]*xx;
    xx*=x;
   }
  }
return ff;
}

public static double[][]  Vandermonde(double xi[],double yi[],int aradegersayisi)
 {
            //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
            // ara değer sayısı 2 dir
            int n=xi.length;
            int nn=(n-1)*(aradegersayisi+1)+1;
            double z[][]=new double[2][nn];
   double E[]=Vandermonde_coef(xi,yi);
   System.out.println("katsayılar :\n"+Matrix.toStringT(E));
   double dx=0;
   int k=0;
   int i;
   for(i=0;i<(n-1);i++)
     {z[0][k]=xi[i];z[1][k]=func(E,z[0][k]);k++;
     for(int j=0;j<aradegersayisi;j++)
     {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
     z[0][k]=z[0][k-1]+dx;z[1][k]=func(E,z[0][k]);k++;}
     }
```

```
      z[0][k]=xi[i];z[1][k]=func(E,z[0][k]);
    return z;
 }

  public static void main(String args[]) throws IOException
  {
  double x[];
  double y[];
  String s1=JOptionPane.showInputDialog("Data file name (input1.txt)");
  // JFileChooser fc=new JFileChooser();
  // if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName();
}
  double a[][]=Text.readDoubleT(s1);
  x=a[0];
  y=a[1];
  double z[][]=Vandermonde(x,y,2);
  System.out.println("Vandermonde interpolasyonu\n"+Matrix.toString(Matrix.T(z)));
  Plot pp=new Plot(a[0],a[1]);
  pp.setPlabel("Vandermonde interpolasyonu");
  pp.setXlabel("x");
  pp.setYlabel("y=f(x)");
  pp.setPlotType(0,22);
  pp.addData(z[0],z[1]);
  pp.setPlotType(1,0);
  pp.setGrid(1,1);
  pp.plot();
  }
}
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" Vandermonde1
katsayılar :
      58.000000000000000
     -20.883333333333322
       3.561666666666663
      -0.250333333333333
       0.007933333333333
      -0.000093333333333

Vandermonde interpolasyonu
       5.000000000000000      15.999999999999966
       6.666666666666667      17.342935528120652
       8.333333333333334      20.949245541838035
      10.000000000000000      24.999999999999766
      11.666666666666666      28.425240054869330
      13.333333333333332      30.759945130314990
      15.000000000000000      31.999999999999332
      16.666666666666668      32.458161865568530
      18.333333333333336      32.620027434840920
      20.000000000000000      32.999999999998580
      21.666666666666668      33.997256515773074
      23.333333333333336      35.751714677638350
      25.000000000000000      37.999999999997500
```

26.666666666666668	39.931412894372440
28.333333333333336	40.043895747595340
30.000000000000000	35.999999999995910

The same data is entered as the previous version. Note the small differences in the coeficient vector and interpolation values.

6.9 Kübik şerit interpolasyonu

İnterpolasyon yapmanın diğer bir yolu tüm noktalardan geçen polinomlarla noktaları bağlamaktır. Örneğin üçüncü dereceden bir polinom düşünülebilir.

$r_k(x)=a_k(x-x_k)^3+ b_k(x-x_k)^2+ c_k(x-x_k)+y_k \quad 1 \le k \le n$

interpolasyon prosesinde polinomların veri noktalarından geçmesi gerekir.

$r_k(x_{k+1})=y_{k+1} \qquad 1 \le k \le n$

aynı zamanda birinci türevlerin de sürekli olması gerekir.

$r'_{k-1}(x_k)= r'_k(x_k) \qquad 1 \le k \le n$

üçüncü dereceden polinom için ikinci türevleri de eşitleyebiliriz.

$r''_{k-1}(x_k)= r''_k(x_k) \quad 1 \le k \le n$

tüm sistemi çözmek için iki şart daha gerekir. Bu şartlar

$r''_1(x_1)=0$

$r''_{n-1}(x_n)=0$

olarak alınırsa buna doğal kübik şerit interpolayonu adını veririz. Başka sınır şartları belirlememiz de mümkündür.

$h_k=x_{k+1}-x_k \quad 1 \le k \le n$

Tum bu şartla bir denklem sistemi olarak bir araya toplanırsa :

$a_k h_k^3+ b_k h_k^2+c_k h_k = y_{k+1}-y_k, \quad 1 \le k \le n$

$3a_{k-1} h_{k-1}^2+ 2b_{k-1} h_{k-1}+c_{k-1}-c_k = 0 \quad , \quad 1 \le k \le n$

$6a_{k-1} h_{k-1}+ 2b_{k-1}+2b_k = 0 \quad , \quad 1 \le k \le n$

$2b_0 =0$

$6a_{n-1} h_{n-1}+ 2b_{n-1} = 0$

seti oluşur bu set 3n-3 denklem içerir.bu sayıda denklemi bir arada çözme işlemi matris çözümlemesine oldukça ağır bir yük getirebilir artı hata olasılıklarını arttırır. Toplam çözülmesi gereken denklem sayısını azaltmanın bir yolu değiştirilmiş özel bir üçüncü dereceden polinom kullanmaktır. Eğer kübik polinomumuz

$\mathbf{s_k(x)=a_k(x-x_k)+ b_k(x_{k+1}-x)+ [(x-x_k)^3 c_{k+1} +(x_{k+1}-x)^3 c_k]/(6h_k) \quad 1 \le k \le n}$

şeklinde verilmiş ise

$s'_k(x)=a_k- b_k+ [(x-x_k)^2 c_{k+1} - (x_{k+1}-x)^2 c_k]/(2h_k) \qquad 1 \le k \le n$

$s''_k(x)=[(x-x_k) c_{k+1} + (x_{k+1}-x) c_k]/h_k \qquad 1 \le k \le n$

olur burada a_k ve b_k c_k nın fonksiyonu olarak yazılabilir.

$b_k=[6y_k-h_k c_k]/(6h_k), \qquad 1 \le k \le n$

$a_k=[6y_{k+1}-h_k^2 c_{k+1}]/(6h_k), \qquad 1 \le k \le n$

Bu durumda çözülmesi gereken denklem sistemi sadece c_k terimlerine dönüşür.

$$h_{k-1}c_{k-1}+ 2(h_{k-1}- h_k)c_k+ h_k c_{k+1} = 6\left[\frac{y_{k+1}-y_k}{h_k} - \frac{y_k-y_{k-1}}{h_{k-1}}\right] \quad , \quad 1 \le k \le n$$

bu sistemde toplam n-2 denklem mevcuttur.

$$w_k = \frac{y_{k+1}-y_k}{h_k}, \quad 1 \le k \le n$$

tanımını yaparsak çözülecek denklem sistemini

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 & \dots & 0 & 0 & 0 \\ 0 & h_2 & 2(h_2+h_3) & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 2(h_{n-3}+h_{n-2}) & h_{n-2} & 0 \\ 0 & 0 & 0 & \dots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \dots \\ c_{n-2} \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} A \\ 6(w_2-w_1) \\ 6(w_3-w_2) \\ \dots \\ 6(w_{n-2}-w_{n-3}) \\ 6(w_{n-1}-w_{n-2}) \\ B \end{bmatrix}$$

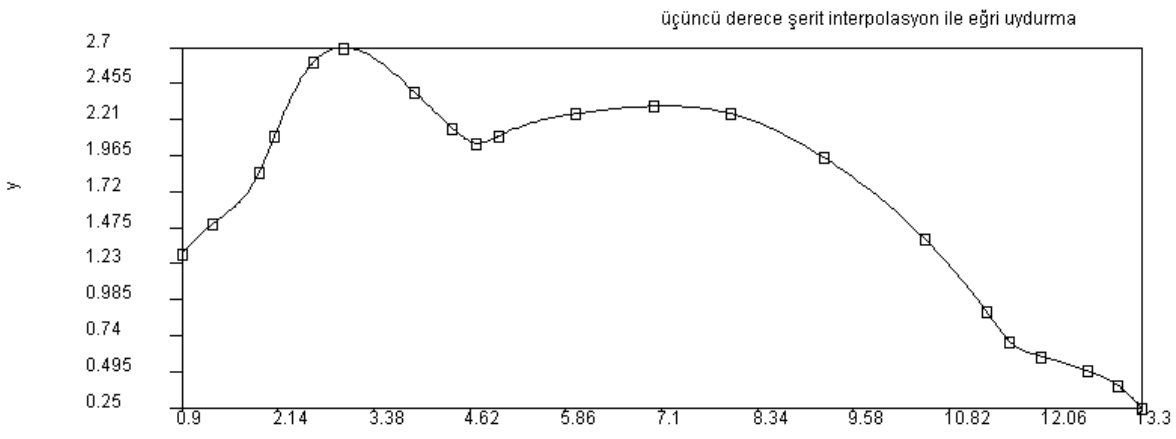
burada A ve B kullanıcı tarafından verilmesi gereken sınır şartlarıdır.

Program 6.9.1 Şerit (spline) interpolasyonu

```
import java.io.*;

class SCO11F
{

  public static double [] thomas(double a[][],double r[])
  {
  //
  int n=a.length;
  double f[]=new double[n];
  double e[]=new double[n];
  double g[]=new double[n];
  double x[]=new double[n];
  for(int i=0;i<n;i++)     {f[i]=a[i][i];}
  for(int i=0;i<(n-1);i++) {g[i]=a[i][i+1];}
  for(int i=0;i<(n-1);i++) {e[i+1]=a[i+1][i];}
  for(int k=1;k<n;k++)
     {e[k]=e[k]/f[k-1];
      f[k]=f[k]-e[k]*g[k-1];
     }
  for(int k=1;k<n;k++)
     {r[k]=r[k]-e[k]*r[k-1];
     }
  x[n-1]=r[n-1]/f[n-1];
  for(int k=(n-2);k>=0;k--)
     {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
  return x;
  }

  public static double [] thomas(double f[],double e[],double g[],double r[])
  {
  int n=f.length;
  double x[]=new double[n];
  for(int k=1;k<n;k++)
     {e[k]=e[k]/f[k-1];
      f[k]=f[k]-e[k]*g[k-1];
     }
  for(int k=1;k<n;k++)
     {r[k]=r[k]-e[k]*r[k-1];
     }
  x[n-1]=r[n-1]/f[n-1];
  for(int k=(n-2);k>=0;k--)
     {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
  return x;
  }


  public static double [][] cubic_spline(double xi[],double yi[],double c0,double cn)
```

```
 {
 int n=xi.length;
 double h[]=new double[n];
 double w[]=new double[n];
 double f[]=new double[n];
 double e[]=new double[n];
 double g[]=new double[n];
 double d[]=new double[n];
 double x[]=new double[n];
 double S[][]=new double[4][n];
 int k;
 for(k=0;k<(n-1);k++)
   {h[k]=xi[k+1]-xi[k];
    w[k]=(yi[k+1]-yi[k])/h[k];
              }
 d[0]=c0;
 d[n-1]=cn;
 for(k=1;k<(n-1);k++)
   {d[k]=6.0*(w[k]-w[k-1]);}
 f[0]=1.0;
 f[n-1]=1.0;
 g[0]=0.0;
 g[n-1]=0.0;
 e[0]=0.0;
 e[n-1]=0.0;
 for(k=1;k<(n-1);k++)
 {f[k]=2.0*(h[k]+h[k-1]);e[k]=h[k-1];g[k]=h[k];}
 S[2]=thomas(f,e,g,d);
 S[3]=xi;
 for(k=0;k<(n-1);k++)
  {S[0][k]=(6.*yi[k+1]-h[k]*h[k]*S[2][k+1])/(6.0*h[k]);
   S[1][k]=(6.*yi[k]-h[k]*h[k]*S[2][k])/(6.0*h[k]);
  }
 return S;
 }

 public static double funcSpline(double S[][],double x)
 {
 int n=S[0].length;
 double xx1=0;
 double xx2=0;
 double y=0;
 double hk=0;
 for(int k=0;k<(n-1);k++)
   {if(S[3][k]<=x && x<=S[3][k+1])
    {hk=(S[3][k+1]-S[3][k]);
     xx1=(x-S[3][k]);
     xx2=(S[3][k+1]-x);
     y=S[0][k]*xx1+S[1][k]*xx2+(xx1*xx1*xx1*S[2][k+1]+xx2*xx2*xx2*S[2][k])/(6.0*hk);
     break;
     }
   }
     if(y==0 && S[3][n-2]<=x )
     {
               int k=n-2;
               hk=(S[3][k+1]-S[3][k]);
     xx1=(x-S[3][k]);
     xx2=(S[3][k+1]-x);
     y=S[0][k]*xx1+S[1][k]*xx2+(xx1*xx1*xx1*S[2][k+1]+xx2*xx2*xx2*S[2][k])/(6.0*hk);
     }
 return y;
 }

public static double[][]  funcSpline(double xi[],double yi[],int aradegersayisi)
{
          //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
```

```
            // ara değer sayısı 2 dir
            int n=xi.length;
            int nn=(n-1)*(aradegersayisi+1)+1;
            double z[][]=new double[2][nn];
  double S[][]=cubic_spline(xi,yi,0,0);
  double dx=0;
  int k=0;
  int i;
  for(i=0;i<(n-1);i++)
    {  z[0][k]=xi[i];z[1][k]=funcSpline(S,z[0][k]);k++;
      for(int j=0;j<aradegersayisi;j++)
        {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
      z[0][k]=z[0][k-1]+dx;z[1][k]=funcSpline(S,z[0][k]);k++;}
    }
    z[0][k]=xi[i];z[1][k]=funcSpline(S,z[0][k]);
  return z;
}

 public static void main(String args[]) throws IOException
 {
 double x[]={0.9,1.3,1.9,2.1,2.6,3.0,3.9,4.4,4.7,5.0,6.0,7.0,8.0,9.2,10.5,11.3,11.6,12.0,12.6,13.0,13.3};
 double y[]={1.3,1.5,1.85,2.1,2.6,2.7,2.4,2.15,2.05,2.1,2.25,2.3,2.25,1.95,1.4,0.9,0.7,0.6,0.5,0.4,0.25};
 double z3[][]=funcSpline(x,y,20);
 System.out.println("Spline \n"+Matrix.toStringT(z3));
 Text.print(Text.T(z3),"üçüncü derece şerit interpolasyon ile eğri uydurma");
 Plot pp=new Plot(x,y);
 pp.setPlabel("üçüncü derece şerit interpolasyon ile eğri uydurma");
 pp.setPlotType(0,20);
 pp.addData(z3[0],z3[1]);
 pp.plot();
 }
}
```



(bolum6_009.jpg,Resimaltı: **Kübik şerit interpolasyonu grafik çıktısı**)

6.10 B şerit interpolasyonu

Bu bölümümüzde B şerit interpolasyonunu inceleyeceğiz. Genel formda B şerit eğrilerinin –sonsuzdan artı sonsuza kadar bir bölgede tanımlanması gerekir. Biz burada kübik B şerit interpolasyonu üzerinde duracağımızdan temel olarak uydurmak istediğimiz eğri verilen $k_0,\ldots,k_N$ nokta için:

$S(x)=\sum_{i=-3}^{N} a_i B_i(x)$ fonksiyonuna eğri uyduracağız. Buradaki $B_i(x)$ fonksiyonuna B şerit fonksiyonu adını vereceğiz.

B şerit fonksiyonu k_i, i=-3,-2,0,…,N,N+1,N+2 noktaları için, ve [a,b] intervalindeki bir x noktasında ve k_0=a, k_N=b olmak üzere

$\sum_{i=-3}^{N} B_i(x) = 1$ şartını sağlamalıdır. S(x) denkleminde N+3 katsayı mevcuttur [a,b] bölgesi ise N+1 fonksiyon değeri sunar, görüldüğü gibi bu şartlar tüm katsayıları çözmeye yeterli değildir, ek şartlara ihtiyaç vardır. Biz burada B şerit fonksiyonlarını iteratif proses yardımı ile tanımlayacağız. B şerit interpolasyon fonksiyonlarının toplamının bire eşit olma şartı tüm polinom dereceleri için geçerli olacaktır. Bu durumda kübik şerit fonksiyonu için (n=1,2,3,4)

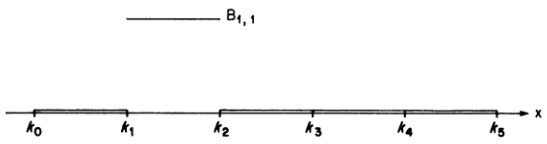
$\sum_{i=-3}^{N+3} B_{i,n}(x) = 1$ yazılabilir.

B şerit fonksiyonları için genel iteratif denklem olarak :

$B_{i,n}(x) = \frac{(x-k_i)}{(k_{i+n-1}-k_i)} B_{i,n-1}(x) + \frac{(k_{i+n}-x)}{(k_{i+n}-k_{i+1})} B_{i+1,n-1}(x)$ denklemini kullanabiliriz. Burada i=-3,…N-1 aralığında değişmektedir. Polinom üssü olarak ta n=1,2,3,4 alınacaktır. İteratif prosese başlayabilmek için bir başlangıç değerine ihtiyacımız vardır. Bunun için birinci dereceden $B_{i,1}$ şerit fonksiyonlarını tanımlayacağız.

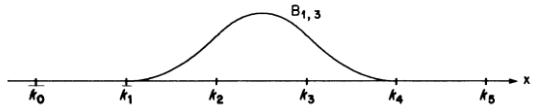
$$B_{i,1} = \begin{cases} 0 & x < k_i \\ 1 & k_i \le x < k_{i+1} \\ 0 & k_{i+1} \le x \end{cases} \quad i = -3,-2,...,N+3$$

Fonksiyonun i=1 için görünümü şekil 6.9-1 de verilmiştir.



(bolum6_010.jpg,Resimaltı: **Birinci derece B şerit fonksiyonu**)

Birinci dereceden denklemi kullanarak iterasyon denklemi yardımıyla üçüncü dereceden denklemin katsayılarını hesaplayabiliriz. Kübik şerit fonksiyonunun şekli Şekil 6.9-2 de verilmiştir.



(bolum6_011.jpg,Resimaltı: **Üçüncü derece B şerit fonksiyonu çıktısı**)

Kübik B şerit interpolasyon denklemini oluşturmak için katsayılar matrisini de hesaplamamız gerekir. Şerit interpolasyon denklemine $S(k_j)=f_j$ j=0,1,2,…N değerini yerleştirirsek denklem

$\sum_{i=-3}^{N-1} a_i B_i(k_j) = f_j \quad j = 0,1,....,N$ formunu alır. Denklemde sadece B_{j-3}, B_{j-2}, B_{j-1} sıfırdan farklıdır. Diğer değerler sıfır olur. Tüm katsayıları çözmek için 2 ek şart daha gerekir. Bunlar için kübik şerit interpolasyonunda kullandığımız ikinci türevlerin sıfıra eşitlenmesi (doğal kübit şerit interpolasyonu) şartı uyarlanabilir.

$$S''(k_0) = 0$$

$$S''(k_N) = 0$$

Veya üstteki denkleme uygulanmış formunda

$$a_{-3}B''_{-3}(k_0) + a_{-2}B''_{-2}(k_0) + a_{-1}B''_{-1}(k_0) = 0$$

$$a_{N-3}B''_{N-3}(k_N) + a_{N-2}B''_{N-2}(k_N) + a_{N-1}B''_{N-1}(k_N) = 0$$

Denklemdeki B şerit fonksiyonlarının ikinci türevleri

$$B''_{-3}(k_0) = \frac{6}{(k_1 - k_{-1})(k_1 - k_{-2})}$$

$$B''_{-2}(k_0) = \frac{-6}{(k_1 - k_{-1})}\left\{\frac{1}{(k_2 - k_{-1})} + \frac{1}{(k_1 - k_{-2})}\right\}$$

$$B''_{-1}(k_0) = \frac{6}{(k_1 - k_{-1})(k_2 - k_{-1})}$$

$$B''_{N-3}(k_N) = \frac{6}{(k_{N+1} - k_{N-1})(k_{N+1} - k_{N-2})}$$

$$B''_{N-2}(k_N) = \frac{6}{(k_{N+1} - k_{N-1})}\left\{\frac{1}{(k_{N+2} - k_{N-1})} + \frac{1}{(k_{N+1} - k_{N-2})}\right\}$$

$$B''_{N-1}(k_N) = \frac{6}{(k_{N+1} - k_{N-1})(k_{N+2} - k_{N-1})}$$

Verinin bir parçası olarak verilmemiş olan k_{-3}, k_{-2}, k_{-1}, k_{N+1}, k_{N+2}, k_{N+3} değerleri küçükten büyüğe sıralanmak şartı ile gelişigüzel seçilebilirler.

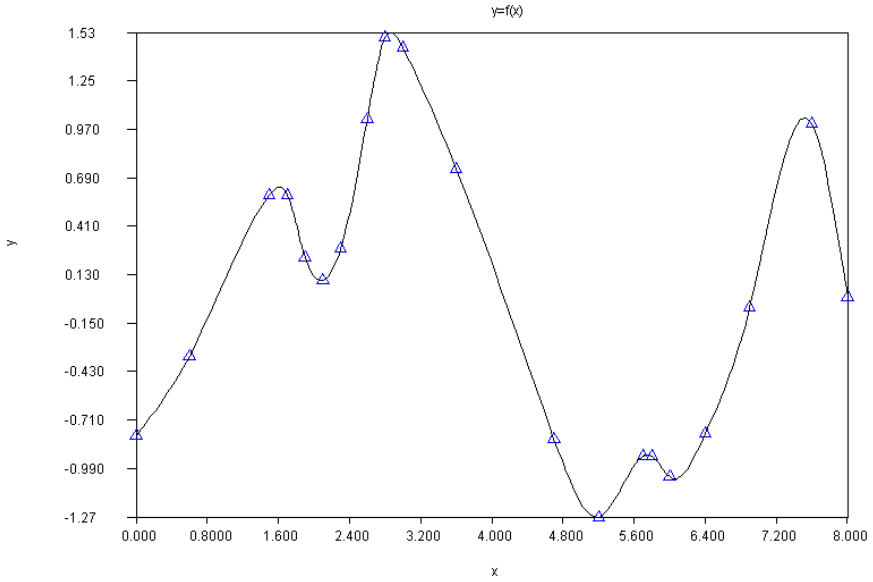
Program 6.10.1 kübik B-Şerit (spline) interpolasyonu

```
//B spline eğri uydurma
import javax.swing.*;

public class B_spline2
{ // cubic B_spline
  public double x[];
  public double f[];
  public double k[];
  int N;
  int n1;
  int m;
  double B[];
  double B2[];
  double a[];

  public B_spline2(double xi[],double yi[])
  {   N=yi.length-1;
     n1=N+7;
              x=new double[N+1];
              f=new double[N+1];
              k=new double[n1];
     for(int i=0;i<=N;i++)
     {x[i]=xi[i];f[i]=yi[i];k[i+3]=x[i];
     }
     double dx=(x[N]-x[0]);
     k[0]=x[0]-6.0*dx;
     k[1]=x[0]-4.0*dx;
     k[2]=x[0]-2.0*dx;
     k[N+4]=x[N]+2.0*dx;
     k[N+5]=x[N]+4.0*dx;
     k[N+6]=x[N]+6.0*dx;
     m=5;
     B=new double[N+7];
     B2=new double[N+7];
     A();
```

```
  }

public  double[] B(double t)
{
double b[][]=new double[m][N+7];
for(int i=0;i<k.length-1;i++)
{if(t>=k[i] && t<k[i+1]) b[1][i]=1;
 else b[1][i]=0.0;
}
for(int n=2;n<m;n++)
{
for(int i=0;i<k.length-4;i++)
{
b[n][i]=(t-k[i])/(k[i+n-1]-k[i])*b[n-1][i]+(k[i+n]-t)/(k[i+n]-k[i+1])*b[n-1][i+1];
B[i]=b[n][i];
}
}
return B;
}

public void A()
{
B2[0]=6.0/(k[4]-k[2])/(k[4]-k[1]);
B2[1]=-6.0/(k[4]-k[2])*(1.0/((k[5]-k[2])+1.0/(k[4]-k[1]) ));
B2[2]=6.0/(k[4]-k[2])/(k[5]-k[2]);
B2[N]=6.0/(k[N+4]-k[N+2])/(k[N+4]-k[N+1]);
B2[N+1]=6.0/(k[N+4]-k[N+2])*(1.0/((k[N+5]-k[N+2])+1.0/(k[N+4]-k[N+1]) ));
B2[N+2]=6.0/(k[N+4]-k[N+2])/(k[N+5]-k[N+2]);
double AA[][]=new double[N+3][N+3];
double BB[]=new double[N+3];
AA[0][0]=B2[0];
AA[0][1]=B2[1];
AA[0][2]=B2[2];
BB[0]=0;
AA[N+2][N]=B2[N];
AA[N+2][N+1]=B2[N+1];
AA[N+2][N+2]=B2[N+2];
BB[N+2]=0;
for(int j=0;j<=N;j++)
{ B=B(k[j+3]);
  for(int i=0;i<=N+2;i++)
  { AA[j+1][i]=B[i];}
  BB[j+1]=f[j];
}
a=pivotlugauss(AA,BB);
}


public double S(double x)
{ B=B(x);
  double ss=0;
  for(int i=0;i<=N+2;i++)
  {ss+=a[i]*B[i];}
  return ss;
}

 public static  double[][]  funcB_Spline(double xi[],double yi[],int aradegersayisi)
 {
    B_spline b1=new B_spline(xi,yi);
    double aa[][]=b1.funcB_Spline(aradegersayisi);
    return aa;
 }

 public  double[][]  funcB_Spline(int aradegersayisi)
 {
              //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
```

```
            // ara değer sayısı 2 dir
            int n=x.length;
            int nn=(n-1)*(aradegersayisi+1)+1;
            double z[][]=new double[2][nn];
  double dx=0;
  int k=0;
  int i;
  for(i=0;i<(n-1);i++)
    {  z[0][k]=x[i];z[1][k]=S(z[0][k]);k++;
      for(int j=0;j<aradegersayisi;j++)
        {dx=(x[i+1]-x[i])/((double)aradegersayisi+1.0);
      z[0][k]=z[0][k-1]+dx;z[1][k]=S(z[0][k]);k++;}
    }
    z[0][k]=x[i];z[1][k]=S(z[0][k]);
  return z;
}

 public static double[] pivotlugauss(double a[][],double b[])
 { //kısmi pivotlu gauss eleme yöntemi
 int n=b.length;
 double x[]=new double[n];
 double carpan=0;
 double toplam=0;
 double buyuk;
 double dummy=0;
 //gauss eleme
 int i,j,k,p,ii,jj;
 //System.out.println("Orijinal Matris :\n"+Matrix.toString(a));
 for(k=0;k<(n-1);k++)
 { //pivotlama
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
              if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
                a[k][jj]=dummy;
              }
              dummy=b[p];
              b[p]=b[k];
              b[k]=dummy;
            }
              //System.out.println("pivotlu Matris : k="+k+"\n"+Matrix.toString(a));
            //gauss elemeyi çözme
  for(i=k+1;i<n;i++)
  {  carpan=a[i][k]/a[k][k];
    a[i][k]=0;
    for(j=k+1;j<n;j++)
    {   a[i][j]-=carpan*a[k][j]; }
       b[i]   =b[i]   -carpan*b[k];
            }
 }
 //System.out.println("elimine prosesibitmiş matris : \n"+Matrix.toString(a));
 //geriye doğru yerine koyma
 x[n-1]=b[n-1]/a[n-1][n-1];
 for(i=n-2;i>=0;i--)
 {
  toplam=0;
  for(j=i+1;j<n;j++)
  {  toplam+=a[i][j]*x[j];}
  x[i]=(b[i]-toplam)/a[i][i];
 }
```

```
  return x;
  }

  public static void main(String[] arg)
  {
    double x[]={0.0,  0.6 ,1.5 ,1.7 ,1.9 ,2.1,2.3 ,2.6 ,2.8,3.0 ,3.6 ,4.7  ,5.2  ,5.7  ,5.8  , 6.0 , 6.4 ,6.9,7.6,8.0};
    double y[]={-0.8,-0.34,0.59,0.59,0.23,0.1,0.28,1.03,1.5,1.44,0.74,-0.82,-1.27,-0.92,-0.92,-1.04,-0.79,-0.06,1.0,0.0};
    int aradegersayisi=5;
    double aa[][]=funcB_Spline(x,y,10);
    Plot1 pp=new Plot1(x,y);
    pp.setPlotType(0,24);
    pp.setColor(0,0,0,255);
    pp.addData(aa[0],aa[1]);
    pp.plot();
  }
}
```



(bolum6_012.jpg,Resimaltı: **Kübik B-Şerit interpolasyonu örnek problem grafik çıktısı**)

6.11 Lineer olmayan katsayılı bir ve çok boyutlu bağımsız değişkenli fonksiyonlara eğri uydurma

Lineer olmayan katsayıları olan f(x;a) fonksiyonuna

$y = f(a;x)$ genel fonksiyonu verildiğinde, bu polinoma x_i,f_i, i=0...n verisini uydurmak istiyoruz. Burada a katsayıları fonksiyonda lineer olmayan bir formda yer almaktadır. En küçük kareler hata fonksiyonunu en genel formunda

$$H(a_0^{(m)},....,a_m^{(m)}) = \sum_{i=1}^{n} w(x_0^i, x_1^i,..., x_k^i)\left[y_i - f(a_0^{(m)},....,a_m^{(m)}, x_0^i, x_1^i,..., x_k^i)\right]^2$$

şeklinde yazabiliriz. Ancak bu durumda lineer sistemlerde olduğu gibi katsayılardan bağımsız bir türev denklemi oluşturmamız mümkün olamaz. Programimiz hata denkleminin minimizasyonu olduğu için bu denklemi optimizasyon konusunda gördüğümüz metodları kullanarak minimize edebiliriz. Örnek programda Nelder-Mead minimizasyon metodu kullanılmıştır. Önce fonksiyonlarımızı bir boyutlu (bağımsız değişkenli) olarak irdeleyelim.

Örnekte kullandığımız veri : (a.txt)

```
5.0 16.0
10.0 25.0
```

```
15.0 32.0
20.0 33.0
25.0 38.0
30.0 36.0
```

Eğri uydurmak istediğiz lineer olmıyan fonksiyon :

double pi=ai[0]*(1.0-Math.exp(-ai[1]*x)); şeklindedir. burada ai katsayılar x de bağımsız değişkendir.

Program 6.11.1 Lineer olmayan eğri uydurma örneği, Nelder-Mead Simpleks metodu

```
import java.io.*;
import javax.swing.*;

abstract class f_xj
{
  // single function  multi independent variable
  // a single value is returned  (
  // example f=x[0]+sin(x[1])
  // func(x) returns the value of f
  abstract public double func(double x[]);
}

    class yy extends f_xj
    {
              double xi[]; // independent variable data set
    double yi[]; //dependent variable data set
    double a[];  //fit function coefficient set
    int nn;
              public yy(String filename,double ia[])
    {
     //read the data to curvefit
     //get the data file and initial fit coefficient when class is defined
               xi=new double[500];
     yi=new double[500];
     int n=ia.length;
     a=new double[n];
     seta(ia);
     int i=-1;
     try{
               BufferedReader fin=new BufferedReader(new FileReader(filename));
     try {
     while(fin != null)
     {
     i++;
     xi[i]=Text.readDouble(fin);
     yi[i]=Text.readDouble(fin);
     }
     } catch(EOFException e_eof) {System.out.println("end of file"); }
     }catch(IOException e_io) {System.out.println("dosya bulunamadı"); }
    nn=i;
    a=ia;
    }

    public void seta(double ia[])
    {
                //assign new fit coefficient set
                for(int ii=0;ii<nn;ii++)
          a[ii]=ia[ii];
              }

              public double[] geta()
    {
                // return fit coefficient set
```

```
                return a;
            }

            double Ps(double x,double ai[])
    {
    //sample specific function
            // saturation pressure as a function of saturation temperature
    a=ai;
    double pi=ai[0]*(1.0-Math.exp(-ai[1]*x));
    return pi; //kPa
    }

    public double func(double ai[])
    {
    double ff=0;
    double w;
    double yy;
     for(int i=0;i<nn;i++)
            {w=Ps(xi[i],ai);
            yy=yi[i];
      //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
            w-=yy;
      ff+=w*w;
      //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
            }
    return ff;
            }
}

class zz extends yy
{   public zz(String filename,double ia[])
    {super(filename,ia);}
    public double func(double ai[])
    {
    return -super.func(ai);
            }
}

public class SCO11H
{//2. Nelder ve Mead metodu (simpleks arama metodu)
//_____
public static double[] nelder(f_xj fnelder,double a[],double da[],int maxiteration,double tolerance,int printlist)
{
double x[][]=new double[a.length+1][a.length];
for(int i=0;i<x.length;i++)
  {for(int j=0;j<x[0].length;j++)
   {if(i==j){x[i][j]=a[j]+da[j];}
    else    {x[i][j]=a[j]; }
   }
  }
  // Nelder mead çok boyutlu simplex minimizasyon metodu
  //  Nelder & Mead 1965 Computer J, v.7, 308-313.

  // Giriş değişkenleri tanımlaması
  // fnelder : abstract çok boyutlu fonksiyon f(x)
  // x : for n boyutlu n+1 simlex noktasını içerir bağımsız değişken seti
  // maxiteration : maximum iterasyon sayısı
  // tolerance :
      int NDIMS = x.length-1;
      int NPTS = x.length;
      int FUNC = NDIMS;
      int ncalls = 0;
        ////// başlangıç simplexini oluştur //////////////////
        double p[][]=new double[NPTS][NPTS]; // [row][col] = [whichvx][coord,FUNC]
        double z[]=new double[NDIMS];
        double best = 1E99;
```

```
//////////////// iilk fonksiyon değerlerini hesapla ////////////////
for (int i=0; i<NPTS; i++)
{
  for (int j=0; j<NDIMS; j++)
      {p[i][j] = x[i][j];}
  p[i][NDIMS] = fnelder.func(p[i]);
}
int iter=0;
for (iter=1; iter<maxiteration; iter++)
{
   //////////  lo, nhi, hi noktalarını tanımla //////////////
   int  ilo=0, ihi=0, inhi = -1; // -1 means missing
   double flo = p[0][FUNC];
   double fhi = flo;
   double pavg,sterr;
   for (int i=1; i<NPTS; i++)
   {
     if (p[i][FUNC] < flo)
       {flo=p[i][FUNC]; ilo=i;}
     if (p[i][FUNC] > fhi)
       {fhi=p[i][FUNC]; ihi=i;}
   }
   double fnhi = flo;
   inhi = ilo;
   for (int i=0; i<NPTS; i++)
    if ((i != ihi) && (p[i][FUNC] > fnhi))
      {fnhi=p[i][FUNC]; inhi=i;}
   ////////// çıkış kriteri //////////////
   if ((iter % 4*NDIMS) == 0)
   {
      //yi nin standart hata kriteri set değerinden (tolerance)
      // küçük olmalı
      // ortalama değeri hesapla (en büyük değer de dahil olmak üzere)
      pavg=0;
      for(int i=0;i<NPTS;i++)
        pavg+=p[i][FUNC];
      pavg/=NPTS;
      double tot=0;
      if(printlist!=0)
      {  System.out.print(iter);
         for (int j=0; j<=NDIMS; j++)
            { System.out.print(p[ilo][j]+" ");}
         System.out.println("");
      }
      for(int i=0;i<NPTS;i++)
      { tot=(p[i][FUNC]-pavg)*(p[i][FUNC]-pavg);}
      sterr=Math.sqrt(tot/NPTS);
      //if(sterr < tolerance)
        { for (int j=0; j<NDIMS; j++)
            { z[j]=p[ilo][j];}
          //break;
        }
      best = p[ilo][FUNC];
   }

   ///// ave[] vektorünü en büyük değeri hariç tutarak hesapla //////

   double ave[] = new double[NDIMS];
   for (int j=0; j<NDIMS; j++)
    ave[j] = 0;
   for (int i=0; i<NPTS; i++)
    if (i != ihi)
      for (int j=0; j<NDIMS; j++)
        ave[j] += p[i][j];
   for (int j=0; j<NDIMS; j++)
     ave[j] /= (NPTS-1);
```

```
//////// yansıt ///////////////

double r[] = new double[NDIMS];
for (int j=0; j<NDIMS; j++)
  r[j] = 2*ave[j] - p[ihi][j];
double fr = fnelder.func(r);

if ((flo <= fr) && (fr < fnhi))  // in zone: accept
{
   for (int j=0; j<NDIMS; j++)
     p[ihi][j] = r[j];
   p[ihi][FUNC] = fr;
   continue;
}

if (fr < flo)  //// genişlet, else kabul et
{
   double e[] = new double[NDIMS];
   for (int j=0; j<NDIMS; j++)
     e[j] = 3*ave[j] - 2*p[ihi][j];
   double fe = fnelder.func(e);
   if (fe < fr)
   {
     for (int j=0; j<NDIMS; j++)
       p[ihi][j] = e[j];
     p[ihi][FUNC] = fe;
     continue;
   }
   else
   {
     for (int j=0; j<NDIMS; j++)
       p[ihi][j] = r[j];
     p[ihi][FUNC] = fr;
     continue;
   }
}

////////////// daralt:

if (fr < fhi)
{
   double c[] = new double[NDIMS];
   for (int j=0; j<NDIMS; j++)
     c[j] = 1.5*ave[j] - 0.5*p[ihi][j];
   double fc = fnelder.func(c);
   if (fc <= fr)
   {
      for (int j=0; j<NDIMS; j++)
        p[ihi][j] = c[j];
      p[ihi][FUNC] = fc;
      continue;
   }
   else   /////// daralt
   {
      for (int i=0; i<NPTS; i++)
        if (i != ilo)
        {
           for (int j=0; j<NDIMS; j++)
             p[i][j] = 0.5*p[ilo][j] + 0.5*p[i][j];
           p[i][FUNC] = fnelder.func(p[i]);
        }
      continue;
   }
}
```

```
            if (fr >= fhi)   ///
            {
               double cc[] = new double[NDIMS];
               for (int j=0; j<NDIMS; j++)
                 cc[j] = 0.5*ave[j] + 0.5*p[ihi][j];
               double fcc = fnelder.func(cc);
               if (fcc < fhi)
               {
                  for (int j=0; j<NDIMS; j++)
                    p[ihi][j] = cc[j];
                  p[ihi][FUNC] = fcc;
                  continue;
               }
               else    /////////
               {
                  for (int i=0; i<NPTS; i++)
                    if (i != ilo)
                    {
                       for (int j=0; j<NDIMS; j++)
                         p[i][j] = 0.5*p[ilo][j] + 0.5*p[i][j];
                       p[i][FUNC] = fnelder.func(p[i]);
                    }
               }
            }
         }
      return z;
}
     public static double[] nelder(f_xj fnelder,double a[],double da[],double tolerance)
     {return nelder(fnelder,a,da,300,tolerance,0);}

     public static double[] nelder(f_xj fnelder,double a[],double da[])
     {return nelder(fnelder,a,da,500,1.0e-20,0);}

   public static void main(String args[]) throws IOException
   {
   //String in_name=JOptionPane.showInputDialog(" veri dosyasının isminigiriniz : ");
   String in_name="a.txt";
   double a[];
   a=new double[2];
   a[0] = 30.0;
   a[1] = 1.0;
   double da[];
   da=new double[2];
   da[0] =1.0;
   da[1] =0.2;

   yy f=new yy(in_name,a);
   int i=0;
   for(double t=-100;t<101;t+=1)
   {System.out.println("t="+t+"ti="+f.xi[i]+"Pi="+f.yi[i]+"P="+f.Ps(t,a));i++;};
   System.out.println("f giriş = "+f.func(a));
   double p[]=nelder(f,a,da,1000,1e-20,1);
   String s1=" optimizasyon değeri  : \n"+Matrix.toStringT(p)+"\n";
   s1+="orijinal fonksiyon değeri = "+f.func(a)+"\n";
   s1+="fonksiyon değeri = "+f.func(p);
   String s2="Nelder-Mead en küçük kareler lineer olmiyan eğri uydurma : ";
  System.out.println(s2+s1);    }
}
```

Lineer olmayan eğri uydurma örneği

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO11H
end of file
```

```
Nelder-Mead en küçük kareler lineer olmiyan eğri uydurma :  optimizasyon değeri  :
      38.727045551319954
       0.107456496091311

orijinal fonksiyon değeri = 328.36740341973405
fonksiyon değeri = 7.789833525493602

> Terminated with exit code 0.
```

Aynı problemi değişik yöntemler kullanarak ta çözebiliriz, Örneğin Davidon-Fletcher-Powell yöntemini kullanırsak:

Program 6.11.2 Lineer olmayan eğri uydurma , Davidon-Fletcher-Powell metodu

```
// OPO12F Davidon-Fletcher-Powell optimizasyonu
// Metod Referansı : Numerical Analysis, 8th edition Richard L. Burden, J. Douglas Faires 3
// sayfa 617
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

    class fa extends f_xj
    {
              double xi[]; // independent variable data set
    double yi[]; //dependent variable data set
    double a[];  //fit function coefficient set
    int nn;
              public fa(String filename,double ia[])
    {
      //read the data to curvefit
      //get the data file and initial fit coefficient when class is defined
                xi=new double[500];
      yi=new double[500];
      int n=ia.length;
      a=new double[n];
      seta(ia);
      int i=-1;
      try{
                BufferedReader fin=new BufferedReader(new FileReader(filename));
      try {
      while(fin != null)
      {
      i++;
      xi[i]=Text.readDouble(fin);
      yi[i]=Text.readDouble(fin);
      }
      } catch(EOFException e_eof) {System.out.println("end of file"); }
      }catch(IOException e_io) {System.out.println("dosya bulunamadı"); }
    nn=i;
    a=ia;
    }

    public void seta(double ia[])
    {
                  //assign new fit coefficient set
                  for(int ii=0;ii<nn;ii++)
          a[ii]=ia[ii];
              }

              public double[] geta()
    {
```

```
                  // return fit coefficient set
                  return a;
                }

                double Ps(double x,double ai[])
    {
    //sample specific function
                // saturation pressure as a function of saturation temperature
    a=ai;
    double pi=ai[0]*(1.0-Math.exp(-ai[1]*x));
    //pi=Math.pow(10.0,pi);
    return pi; //kPa
    }

    public double func(double ai[])
    {
    double ff=0;
    double w;
    double yy;
     for(int i=0;i<nn;i++)
                  {w=Ps(xi[i],ai);
                  yy=yi[i];
      //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
                  w-=yy;
      ff+=w*w;
      //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
                  }
    return ff;
                }
}

public class OPO12F2
{
public static double VT_V(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
  {
  tot+=left[i]*left[i];
  }
return tot;
}

public static double[][] V_VT(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
for(int i=0;i<n;i++)
  {
  for(int j=0;j<n;j++)
    {
              aa[i][j]=left[i]*left[j];
              }
  }
return aa;
}

public static double  VT_X(double [] left,double [] right)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
```

```
  {
  tot+=left[i]*right[i];
  }
return tot;
}

public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}

public static double turev(f_xj f,double x[],int denklem_ref,int x_ref)
{
//
// verilen fonksiyonun ikinci türevlerinin matrisi
// fonksiyon f in denklem_ref sayılı fonksiyonunun(sayılar 0 dan başlar)
// x[x_ref] değişkenine göre türevi (sayılar 0 dan başlar)
// df_denklem_ref(x)/d_x_ref
// bu metod newtond metodu içinde kullanılmak içindir.
double h0=0.256808;
int i,m;
int n=7;
double f1[];
f1=new double[x.length];
double f2[];
f2=new double[x.length];
double x1[];
x1=new double[x.length];
double x2[];
x2=new double[x.length];
for(i=0;i<x.length;i++)
{
x1[i]=x[i];
x2[i]=x[i];
}
//turev of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
//vector<double> h(n,0);
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
h[i]=h0*Math.pow(r,i);
}

for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i]*1.0094847;
f1=df(f,x1);
f2=df(f,x2);
T[i][0]=( f1[denklem_ref] - f2[denklem_ref])/(2.0*h[i]);
x1[x_ref]=x[x_ref];
```

```
x2[x_ref]=x[x_ref];
}
for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] - h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]
  - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}
public static double[] df(f_xj f,double xxi[])
{
//Gredyen vektörü
// f(x0,x1,x2,...xn) founksiyonunun türev vektörü
// [df/dx0, df/dx1,df/dx2,....,df/dxn]
// df/dxj  j=0...x.length
//
// bu fonksiyon bilgisayar tarafından hesaplanmaktadır
// kullanıcı tarafından tanımlanan f(x) fonksiyonunun
// sayısal olarak alınan türevidir.

//
double a[]=new double[xxi.length];
for(int i=0;i<xxi.length;i++)
   {
   a[i]=turev(f,xxi,i);
   }
return a;
}
public static double turev(f_xj f,double xxi[],int x_ref)
{
// df/dxj
double h0=0.256808;
int i,m;
int n=7;
double x1[];
x1=new double[xxi.length];
double x2[];
x2=new double[xxi.length];
for(i=0;i<xxi.length;i++)
{
x1[i]=xxi[i];
x2[i]=xxi[i];
}
//turev of a simple function
double T[][];
T=new double[n][n];
double h[];
h=new double[n];
for(i=0;i<n;i++)
{
  h[i]=0;
  for(int j=0;j<n;j++)
    T[i][j]=0;
}
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++)
{
//h[i]=h0*Math.pow(r,i);
h0/=2.0;
h[i]=h0;
}
```

```
//first turev (difference formula)
for(i=0;i<n;i++)
{
x1[x_ref]+=h[i];
x2[x_ref]-=h[i];
T[i][0]=(f.func(x1)-f.func(x2))/(2.0*h[i]);
x1[x_ref]=xxi[x_ref];
x2[x_ref]=xxi[x_ref];
}

for(m=1;m<n;m++)
{
  for(i=0;i<n-m;i++)
  {
  T[i][m]=(h[i]*h[i]*T[i+1][m-1] -
  h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i] - h[i+m]*h[i+m]);
  }
}
double xx=T[0][n-1];
return xx;
}
public static double[] turev(f_xj f_deriv,double x[])
{
// df/dxj  j=0...x.length
// This method calculates turev of a function with more than one independent variable.
// Accuracy of method can be adjusted by changing variables h0 and n
// function input should be in the form given in abstract class
// f_xj,j=0...x.length = df/dx(x_ref)
double a[]=new double[x.length];
for(int x_ref=0;x_ref<x.length;x_ref++)
   {
   a[x_ref]=turev(f_deriv,x,x_ref);
   }
return a;
}

public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}


public static double linmin(f_xj f,double p[],double xi[])
{
                double tol=2.0e-4;
    int n=p.length;
    double pcom[],xicom[];
                int j;
                double xx,xmin,bx,ax;
                f1dim f1=new f1dim(f,p,xi);
                ax=0.0;
                xx=1.0;
                bx=2.0;
                double aa[]=mnbrak(f1,ax,xx,bx);
                ax=aa[0];
                xx=aa[1];
                bx=aa[2];
                xmin=brent(f1,ax,xx,bx,tol);
                return xmin;
}

public static double[][] linminiter(f_xj f,double p[],double xi[])
{   // yeni iterasyon noktasını verir
    int n=p.length;
    double xmin=linmin(f,p,xi);
    double aa[][]=new double[2][n];
                for (int j=0;j<n;j++)
                {   xi[j] *= xmin;
```

```
                        p[j] += xi[j];
                        aa[0][j]=p[j];
                        aa[1][j]=xi[j];
                }
    return aa;
}
public  static double[] brentf(f_x f,double ax,double bx,double cx,double tol)
{
// ax,bx,cx üç adet tahmin noktası (bxax ile cx arasında yer almalıdır)
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double CGOLD=(3.0-Math.sqrt(5))/2.0; //altın oran
double ZEPS=1.0e-10;
double xmin;
double aa[]=new double[2];
// SHFT(double a,double b,double c,d) (a)=(b);(b)=(c);(c)=(d);
        int iter;
        double a,b,d,etemp,fu,fv,fw,fx,p,q,r,tol1,tol2,u,v,w,x,xm;
        double e=0.0;
        d=0.0;
        a=((ax < cx) ? ax : cx);
        b=((ax > cx) ? ax : cx);
        x=w=v=bx;
        fw=fv=fx=f.func(x);
        for (iter=1;iter<=ITMAX;iter++) {
                xm=0.5*(a+b);
                tol2=2.0*(tol1=tol*Math.abs(x)+ZEPS);
                if (Math.abs(x-xm) <= (tol2-0.5*(b-a))) {
                        xmin=x;
            aa[0]=xmin;
            aa[1]=fx;
            return aa;
                }
                if (Math.abs(e) > tol1) {
                        r=(x-w)*(fx-fv);
                        q=(x-v)*(fx-fw);
                        p=(x-v)*q-(x-w)*r;
                        q=2.0*(q-r);
                        if (q > 0.0) p = -p;
                        q=Math.abs(q);
                        etemp=e;
                        e=d;
                        if (Math.abs(p) >= Math.abs(0.5*q*etemp) || p <= q*(a-x) || p >= q*(b-x))
                                d=CGOLD*(e=(x >= xm ? a-x : b-x));
                        else {
                                d=p/q;
                                u=x+d;
                                if (u-a < tol2 || b-u < tol2)
                                        d=SIGN(tol1,xm-x);
                        }
                } else {
                        d=CGOLD*(e=(x >= xm ? a-x : b-x));
                }
                u=(Math.abs(d) >= tol1 ? x+d : x+SIGN(tol1,d));
                fu=f.func(u);
                if (fu <= fx) {
                        if (u >= x) a=x; else b=x;
                        {v=w;w=x;x=u;}
                        {fv=fw;fw=fx;fx=fu;}
                } else {
                        if (u < x) a=u; else b=u;
                        if (fu <= fw || w == x) {
                                v=w;
                                w=u;
                                fv=fw;
```

```
                                        fw=fu;
                                } else if (fu <= fv || v == x || v == w) {
                                        v=u;
                                        fv=fu;
                                }
                        }
                }
                System.out.println("BRENT metodunda maksimum iterasyon sayısı aşıldı");
                xmin=x;
                //minumum değer a[0] ile fonksiyon değeri a[1] ile geridöndürülmektedir
                aa[0]=xmin;
                aa[1]=fx;
                return aa;
}
public  static double[] mnbrak(f_x f,double ax,double dx)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçilen uzaklık
return mnbrak(f,ax,(ax-dx),(ax+dx));
}

public  static double[] mnbrak(f_x f,double ax)
{
// dx ikinci derece polinom için gerekli iki ek noktayı oluşturmak için
// ana noktadan seçiln uzaklık
return mnbrak(f,ax,(ax-0.1),(ax+0.1));
}

public  static double[] mnbrak(f_x f,double ax,double bx,double cx)
{
// ax,bx,cx üç adet tahmin noktası verildiğinde içinde minimum bulunan ax,bx,cx üç adetnoktayı bize arama
// sonucu bulur
// f : fonksiyon (f_x abstract sınıfında tanımlanmış)
// tol tolerans
int ITMAX=100;
double xmin;
double GOLD=(Math.sqrt(5.0)+1.0)/2.0;;
double GLIMIT=100.0;
double TINY=1.0e-20;
double fa,fb,fc;
fa=0;
fb=0;
fc=0;
                double ulim,u,r,q,fu,dum;
                double aa[]=new double[3];
                fa=f.func(ax);
                fb=f.func(bx);
                if (fb > fa) {
                        //SHFT(dum,*ax,*bx,dum)
                        {dum=ax;ax=bx;bx=dum;}
                        //SHFT(dum,*fb,*fa,dum)
                        {dum=fb;fb=fa;fa=dum;}
                }
                cx=(bx)+GOLD*(bx-ax);
                fc=f.func(cx);
                        aa[0]=ax;
                        aa[1]=bx;
                        aa[2]=cx;
                while (fb > fc) {
                        r=(bx-ax)*(fb-fc);
                        q=(bx-cx)*(fb-fa);
                        u=(bx)-((bx-cx)*q-(bx-ax)*r)/
                                (2.0*SIGN(MAX(Math.abs(q-r),TINY),q-r));
                        ulim=(bx)+GLIMIT*(cx-bx);
                        if ((bx-u)*(u-cx) > 0.0) {
                                fu=f.func(u);
```

```
                                        if (fu < fc) {
                                                ax=bx;
                                                bx=u;
                                                fa=fb;
                                                fb=fu;
                                                return aa;
                                        } else if (fu > fb) {
                                                cx=u;
                                                fc=fu;
                                                return aa;
                                        }
                                        u=(cx)+GOLD*(cx-bx);
                                        fu=f.func(u);
                                } else if ((cx-u)*(u-ulim) > 0.0) {
                                        fu=f.func(u);
                                        if (fu < fc) {
                                                //SHFT(bx,cx,u,cx+GOLD*(cx-bx))
                                                {bx=cx;cx=u;u=cx+GOLD*(cx-bx);}
                                                //SHFT(fb,fc,fu,f.func(u))
                                                {fb=fc;fc=fu;fu=f.func(u);}
                                        }
                                } else if ((u-ulim)*(ulim-cx) >= 0.0) {
                                        u=ulim;
                                        fu=f.func(u);
                                } else {
                                        u=cx+GOLD*(cx-bx);
                                        fu=f.func(u);
                                }
                                //SHFT(ax,bx,cx,u)
                                {ax=bx;bx=cx;cx=u;}
                                //SHFT(fa,fb,fc,fu)
                                {fa=fb;fb=fc;fc=fu;}
                                aa[0]=ax;
                                aa[1]=bx;
                                aa[2]=cx;
                }
                                return aa;
}
public  static double brent(f_x f,double ax,double bx,double cx,double tol)
{
 double aa[]=brentf(f,ax,bx,cx,tol);
 return aa[0];
}

public static double[][] I(int n)
{
//unit matrix
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
    b[i][i]=1.0;
return b;
}
// Davidon - Fletcher-Powell metodu
//_____________________________________________________________________
public static double[] davidon_fletcher_powell( f_xj f,double x[])
{
int i,j,k;
int nmax=500;
double gamma=1.0;
double tolerance=1.0e-12 ;
int n=x.length;
double alpha;
double dx[]=new double[n];
double g[] =new double[n];
double ge[]=new double[n];
```

```
double dg[]=new double[n];
double d[] =new double[n];
double r[] =new double[n];
double rdg,dxdg;
double beta;
double alpha_d;
double Q[][]=I(n);
double Q1[][]=new double[n][n];
double Q2[][]=new double[n][n];
int nn=15;
i=0;
j=0;k=0;
g=turev(f,x);
while( j++ < nmax && gamma > tolerance )
{
   for(int ii=0;ii<n;ii++)
   {
    d[ii]=0.0;
    for(int kk=0;kk<n;kk++)
                {d[ii]=d[ii]-Q[ii][kk]*g[kk];}
   }
   System.out.println("j="+j+"d="+Matrix.toString(d));
   alpha=linmin(f,x,d);
   System.out.println("alfa="+alpha);
   if(alpha==0) break;
   for(i=0;i<n;i++){dx[i]=alpha*d[i];x[i]+=dx[i];ge[i]=g[i];}
   System.out.println("j="+j+"dx=\n"+Matrix.toString(dx));
   System.out.println("j="+j+"x=\n"+Matrix.toString(x));
   g=turev(f,x);
   System.out.println("j="+j+"g=\n"+Matrix.toString(g));
   for(i=0;i<n;i++){dg[i]=g[i]-ge[i];}
   gamma=norm(g);
   for(int ii=0;ii<n;ii++)
   {r[ii]=0.0;
               for(int kk=0;kk<n;kk++)
               {r[ii]+=Q[ii][kk]*dg[kk];}
   }
   rdg=VT_X(r,dg);
   dxdg=VT_X(dx,dg);
   Q1=V_VT(r);
   Q2=V_VT(dx);
   for(int ii=0;ii<n;ii++)
     for(int jj=0;jj<n;jj++)
       Q[ii][jj]=Q[ii][jj]-Q1[ii][jj]/rdg+Q2[ii][jj]/dxdg;
   gamma=norm(dx);
   System.out.println("j="+j+"Q=\n"+Matrix.toString(Q));
j++;k++;
if(k>=nn) {k=0;Q=I(n);}
}
if(j >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
public static double[] davidon_fletcher_powell_V1( f_xj f,double x[])
{
int i,j,k;
int nmax=500;
double gamma=1.0;
double tolerance=1.0e-12 ;
int n=x.length;
double alpha;
double dx[]=new double[n];
double g[] =new double[n];
double ge[]=new double[n];
double dg[]=new double[n];
double d[] =new double[n];
```

```
double r[] =new double[n];
double rdg,dxdg;
double beta;
double alpha_d;
double Q[][]=I(n);
double Q1[][]=new double[n][n];
double Q2[][]=new double[n][n];
int nn=15;
i=0;
j=0;k=0;
g=turev(f,x);
while( j++ < nmax && gamma > tolerance )
{

   for(int ii=0;ii<n;ii++)
   {
    d[ii]=0.0;
    for(int kk=0;kk<n;kk++)
                {d[ii]=d[ii]-Q[ii][kk]*g[kk];}
   }
   alpha=linmin(f,x,d);
   if(alpha==0) break;
   for(i=0;i<n;i++){dx[i]=alpha*d[i];x[i]+=dx[i];ge[i]=g[i];}
   g=turev(f,x);
   for(i=0;i<n;i++){dg[i]=g[i]-ge[i];}
   gamma=norm(g);
   for(int ii=0;ii<n;ii++)
   {  r[ii]=0.0;
               for(int kk=0;kk<n;kk++)
               {r[ii]+=Q[ii][kk]*dg[kk];}
   }
   rdg=VT_X(r,dg);
   dxdg=VT_X(dx,dg);
   beta=dxdg/rdg;
   Q1=V_VT(r);
   Q2=V_VT(dx);
   for(int ii=0;ii<n;ii++)
     for(int jj=0;jj<n;jj++)
       Q[ii][jj]=beta*(Q[ii][jj]-Q1[ii][jj]/rdg)+Q2[ii][jj]/dxdg;
   gamma=norm(dx);
j++;k++;
if(k>=nn) {k=0;Q=I(n);}
}
if(j >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
return x;
}
            public static double[] verigir(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }
            public static void main (String args[])
    {
            String s="kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ";
    double [] x0=verigir(s);
```

```
    fa f_x=new fa("a.txt",x0);
    double [] r1= davidon_fletcher_powell(f_x,x0);
    double [] r2= davidon_fletcher_powell_V1(f_x,x0);
    s=" çözüm seti DFP : \n"+Matrix.toStringT(r1);
    s+=" çözüm seti düzeltilmiş DFP : \n"+Matrix.toStringT(r2);
    String s2="Davidon - Fletcher - Powell çok değişkenli fonksiyon optimizasyonu:";
    JOptionPane.showMessageDialog(null,s,s2,JOptionPane.PLAIN_MESSAGE);
    System.exit(0);
    }
}
```

Lineer olmayan eğri uydurma örneği

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO12F2
end of file
Davidon - Fletcher - Powell çok değişkenli fonksiyon optimizasyonu: çözüm seti DFP :
      38.727045489861894
       0.107456496563057
 çözüm seti düzeltilmiş DFP :
      38.727045489861894
       0.107456496563057

> Terminated with exit code 0.
```

Süreklilik metodu kullanarak :

Program 6.11.3 Lineer olmayan eğri uydurma , Süreklilik metodu

```
// FSCO5D Süreklilik(Continuation-homotopy) optimizasyon
// Metod Referansı : Numerical Analysis, 8th edition Richard L. Burden, J. Douglas Faires 3
// sayfa 637
// bu versiyon fonksiyonların türevini sayısal olarak almaktadır.
// kullanıcının türev fonksiyonu tanımlaması gerekmez


import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

//============= Tanımı gereken fonksiyon ================
abstract class fi_xi
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x) returns the value of f[0] and f[1]
  // as a two dimensional vector

  abstract double[] func(double x[]);
}

abstract class fij_xi
{
  // multifunction multi independent variable
  // for n independent variable n*n matrix of
  // dependent variables are returned
  // example f[0][0]=x[0]+Math.sin(x[1])   f[0][1]=x[0]-x[1]
  //         f[1][0]=x[0]*x[0]-x[1]        f[1][1]=Math.exp(x[0]+x[1]*x[1]
  // func(x) returns the value of f[0][0], f[0][1]
```

```
  //                          f[1][0], f[1][1]
  // as a two dimensional matrix
  abstract double[][] func(double x[]);
}

    class fa extends f_xj
    {
              double xi[]; // independent variable data set
    double yi[]; //dependent variable data set
    double a[];  //fit function coefficient set
    int nn;
              public fa(String filename,double ia[])
    {
     //read the data to curvefit
     //get the data file and initial fit coefficient when class is defined
               xi=new double[500];
     yi=new double[500];
     int n=ia.length;
     a=new double[n];
     seta(ia);
     int i=-1;
     try{
               BufferedReader fin=new BufferedReader(new FileReader(filename));
     try {
     while(fin != null)
     {
     i++;
     xi[i]=Text.readDouble(fin);
     yi[i]=Text.readDouble(fin);
     }
     } catch(EOFException e_eof) {System.out.println("end of file"); }
     }catch(IOException e_io) {System.out.println("dosya bulunamadı"); }
    nn=i;
    a=ia;
    }

    public void seta(double ia[])
    {
                //assign new fit coefficient set
                for(int ii=0;ii<nn;ii++)
          a[ii]=ia[ii];
              }

              public double[] geta()
    {
                // return fit coefficient set
                return a;
              }

              double Ps(double x,double ai[])
    {
    //sample specific function
              // saturation pressure as a function of saturation temperature
    a=ai;
    double pi=ai[0]*(1.0-Math.exp(-ai[1]*x));
    //pi=Math.pow(10.0,pi);
    return pi; //kPa
    }

    public double func(double ai[])
    {
    double ff=0;
    double w;
    double yy;
     for(int i=0;i<nn;i++)
               {w=Ps(xi[i],ai);
```

```
                yy=yi[i];
      //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
                w-=yy;
      ff+=w*w;
      //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
                }
    return ff;
              }
}

public class FSCO5D1
{
// Süreklilik(Continuity-homotopy) metodu kullanarak denklem sisteminin köklerini bulmak
// Bu metod denklem sistemi çözümü için Matrix.java programını
// kullanmaktadır.


public static double[] multiply(double left,double[] right)
{
//multiplying a vector with a constant
int i;
int n=right.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {b[i]=left*right[i];}
return b;
}

public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right.length;
double[] b;
b=new double[m2];
 if(n1 != m2)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
   {
    b[ii]=0;
   }
   return b;
 }
 for(i=0;i<m1;i++)
   {
    b[i]=0;
    for(k=0;k<n1;k++)
              b[i]+=left[i][k]*right[k];
   }
return b;
//end of multiply of a matrix and a vector
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
```

```
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
    b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
    b[i]=b[i]+right[i];
  }
return b;
//end of vector addition method
}

public static double[][] inv(double[][] a)
{
// INVERSION OF A MATRIX
// inversion by using gaussian elimination
// with full pivoting
int n=a.length;
int m=a[0].length;
double b[][];
b=new double[n][n];
int indxc[];
int indxr[];
double ipiv[];
indxc=new int[n];
indxr=new int[n];
ipiv=new double[n];
int i,j,k,l,ll,ii,jj;
int icol=0;
int irow=0;
double big,dum,pivinv,temp;
if(n!=m)
{
   System.out.println("Matrix must be square ");
                  for(ii=0;ii<n;ii++)
                    for(jj=0;jj<n;jj++)
                              b[ii][jj]=0.0;
                  return b;
}
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
      b[i][j]=a[i][j];
for(i=0;i<n;i++)
{
  big=0.0;
  for(j=0;j<n;j++)
  {
    if(ipiv[j] != 1)
      for(k=0;k<n;k++)
      {
                if(ipiv[k] == 0)
                {
                  if(Math.abs(b[j][k]) >= big)
                  {
                    big=Math.abs(b[j][k]);
                    irow=j;
                    icol=k;
                  }
                }
                else if(ipiv[k] > 1 )
                  {
                  System.out.println("error : inverse of the matrix : singular matrix-1");
                  for(ii=0;ii<n;ii++)
                    for(jj=0;jj<n;jj++)
```

```
                        b[ii][jj]=0.0;
                return b;
                }
    }
  }
  ++ ipiv[icol];
  if(irow != icol)
    for(l=0;l<n;l++)
    {
    temp=b[irow][l];
    b[irow][l]=b[icol][l];
    b[icol][l]=temp;
    }
  indxr[i]=irow;
  indxc[i]=icol;
  if(b[icol][icol] == 0.0)
  {
    System.out.println("error : inverse of the matrix : singular matrix-2");
                for(ii=0;ii<n;ii++)
                  for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
    return b;
  }
  pivinv=1.0/b[icol][icol];
  b[icol][icol]=1.0;
  for(l=0;l<n;l++) b[icol][l] *=pivinv;
  for(ll=0;ll<n;ll++)
  if(ll != icol)
  {
    dum=b[ll][icol];
    b[ll][icol]=0.0;
    for(l=0;l<n;l++) b[ll][l]-= b[icol][l]*dum;
  }
}
for(l=n-1;l>=0;l--)
{
  if(indxr[l] != indxc[l])
     for(k=0;k<n;k++)
     {
       temp=b[k][indxc[l]];
       b[k][indxc[l]]=b[k][indxr[l]];
       b[k][indxr[l]]=temp;
     }
}
return b;
}

public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}

public static double[] df(f_xj f,double xxi[])
{
//Gredyen vektörü
// f(x0,x1,x2,...xn) founksiyonunun türev vektörü
// [df/dx0, df/dx1,df/dx2,....,df/dxn]
// df/dxj  j=0...x.length
//
// bu fonksiyon bilgisayar tarafından hesaplanmaktadır
// kullanıcı tarafından tanımlanan f(x) fonksiyonunun
// sayısal olarak alınan türevidir.

//
double a[]=new double[xxi.length];
for(int i=0;i<xxi.length;i++)
   {
   a[i]=dfdx(f,xxi,i);
```

```
   }
return a;
}


public static double dfdx(f_xj f,double xxi[],int xref)
{ int i;
                double h=1.0e-3;
  double hh=1.0/h;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
  for(i=0;i<xxi.length;i++)
  {
  if(i!=xref)
    {xp2h[i]=xxi[i];xph[i]=xxi[i];xm2h[i]=xxi[i];xmh[i]=xxi[i];}
  else {xp2h[i]=xxi[i]+2.0*h;xph[i]=xxi[i]+h;xm2h[i]=xxi[i]-2.0*h;xmh[i]=xxi[i]-h;}
  }
  fp2h=f.func(xp2h);
  fph=f.func(xph);
  fm2h=f.func(xm2h);
  fmh=f.func(xmh);
  return (-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
}
public static double d2fdx2(f_xj f,double xxi[],int xiref,int xjref)
{ //ikincitürev matrisi
  int i;
  double h=1.0e-4;
  double hh=1.0/h;
  int n=xxi.length;
  double ff;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
     for(int j=0;j<n;j++)
     {if(j!=xjref) {xp2h[j]=xxi[j];xph[j]=xxi[j];xm2h[j]=xxi[j];xmh[j]=xxi[j];}
      else {xp2h[j]=xxi[j]+2.0*h;xph[j]=xxi[j]+h;xm2h[j]=xxi[j]-2.0*h;xmh[j]=xxi[j]-h;}
     }
     fp2h=dfdx(f,xp2h,xiref);
     fph=dfdx(f,xph,xiref);
     fm2h=dfdx(f,xm2h,xiref);
     fmh=dfdx(f,xmh,xiref);
     ff=(-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
  return ff;
}

public static double[][]H(f_xj f,double x[])
{
  int n=x.length;
  double dy[][]=new double[n][n];;
  for(int ii=0;ii<n;ii++)
  {
    for(int jj=0;jj<n;jj++)
    {   //Hessian matrisi
        dy[ii][jj]=d2fdx2(f,x,ii,jj);
    }
  }
  return dy;
}
public static double[] continuationRK4(f_xj f,double x[],int N)
{
//=====================================================
```

```
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi J de tanımlanacaktır
// Homotopy, continuation (devamlılık) metodu ile lineer olmıyan denklem sistemi çözümü
// 4.derece Runge-Kutta denklemi ile
// yi+1 = yi + (1/6)*( k1 + 2k2 +2k3+k4)h
// k1=f(xi,yi)
// k2=f(xi+0.5h,yi+0.5k1h)
// k3=f(xi+0.5h,yi+0.5k2h)
// k4=f(xi+h,yi +k3h)
//=================================================
// x  : bağımsız değişken vektörü
// y  : bağımlı değişken vektörü
// dy : bağımlı değişkenin türev fonksiyonları vektörü
// N  : Runge-Kutta diferansiyel denklem noktasayısı

int i;
int nmax=400;
double tolerance=1.0e-20;
int n=x.length;
double h=1.0/(double)N;
double b[]=new double[n];
double x1[]=new double[n];
double k[][]=new double[4][n];
double A[][]=new double[n][n];
b=multiply(-h,df(f,x));
for(i=0;i<N;i++)
{
x1=x;
A=H(f,x1);
k[0]=multiply(inv(A),b);
x1=add(x,multiply(0.5,k[0]));
A=H(f,x1);
k[1]=multiply(inv(A),b);
x1=add(x,multiply(0.5,k[1]));
A=H(f,x1);
k[2]=multiply(inv(A),b);
x1=add(x,k[2]);
A=H(f,x1);
k[3]=multiply(inv(A),b);
for(int j=0;j<n;j++) {x[j]=x[j]+1.0/6.0*(k[0][j]+2.0*k[1][j]+2.0*k[2][j]+k[3][j]);}
}
return x;
}

public static double[] continuationRK6(f_xj f,double x[],int N)
{
//=================================================
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi J de tanımlanacaktır
// Homotopy, continuation (devamlılık) metodu ile lineer olmıyan denklem sistemi çözümü
// 6. derece Runge-Kutta denklemi ile
// yi+1 = yi + (1/90)*( 7k1 + 32k3 +12k4+32k5+7k6)h
// k1=f(xi,yi)
// k2=f(xi+0.25h , yi+0.25k1h)
// k3=f(xi+0.25h , yi+0.125k1h+0.125k2h)
// k4=f(xi+0.5h , yi - 0.5k2h+k3h)
// k5=f(xi+0.75h , yi + (3/16)k1h+(9/16)k4h)
// k6=f(xi+h , yi - (3/7)k1h+(2/7)k2h+(12/7)k3h - (12/7)k4h+(8/7)k5h)
//=================================================
// x  : bağımsız değişken vektörü
// y  : bağımlı değişken vektörü
// dy : bağımlı değişkenin türev fonksiyonları vektörü
// N  : Runge-Kutta diferansiyel denklem noktasayısı
int i;
```

```
int nmax=400;
double tolerance=1.0e-20;
int n=x.length;
double h=1.0/(double)N;
double b[]=new double[n];
double x1[]=new double[n];
double k[][]=new double[6][n];
double A[][]=new double[n][n];
b=multiply(-h,df(f,x));
for(i=0;i<N;i++)
{
x1=x;
A=H(f,x1);
// k1=f(xi,yi)
k[0]=multiply(inv(A),b);
x1=add(x,multiply(0.25,k[0]));
A=H(f,x1);
// k2=f(xi+0.25h , yi+0.25k1h)
k[1]=multiply(inv(A),b);
x1=add(x,add(multiply(0.125,k[0]),multiply(0.125,k[1])));
A=H(f,x1);
// k3=f(xi+0.25h , yi+0.125k1h+0.125k2h)
k[2]=multiply(inv(A),b);
x1=add(x,add(multiply(-0.5,k[1]),k[2]));
A=H(f,x1);
// k4=f(xi+0.5h , yi - 0.5k2h+k3h)
k[3]=multiply(inv(A),b);
x1=add(x,add(multiply((3.0/16.0),k[0]),multiply((9.0/16.0),k[3])));
A=H(f,x1);
// k5=f(xi+0.75h , yi + (3/16)k1h+(9/16)k4h)
k[4]=multiply(inv(A),b);
x1=add(x,
      add(multiply((-3.0/7.0),k[0]),add(multiply((2.0/7.0),k[1]),
      add(multiply((12.0/7.0),k[2]),
         add(multiply((-12.0/7.0),k[3]),multiply((8.0/7.0),k[4])))));
A=H(f,x1);
// k6=f(xi+h , yi - (3/7)k1h+(2/7)k2h+(12/7)k3h - (12/7)k4h+(8/7)k5h)
k[5]=multiply(inv(A),b);
// yi+1 = yi + (1/90)*( 7k1 + 32k3 +12k4+32k5+7k6)h
for(int j=0;j<n;j++) {x[j]=x[j]+1.0/90.0*(7.0*k[0][j]+32.0*k[2][j]+12.0*k[3][j]+32.0*k[4][j]+7.0*k[5][j]);}
}
return x;
}

public static double[] newton_continuationRK4(f_xj f,double x[],int N)
{
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi dfj/dxi de tanımlanacaktır
// Newton-Raphson metodu ile lineer olmıyan denklem sistemi çözümü
//ti :önem katsayısı
//x bağımsız değişken vektörü
//y bağımlı değişken vektörü
//dy bağımlı değişkenin türev fonksiyonları vektörü
x=continuationRK4(f,x,4);
double ti=1.0;
int i;
int nmax=400;
double tolerance=1.0e-10;
int n=x.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
{
b[i]=1.0;
}
```

```
i=0;
while( i++ < nmax && Matrix.abs(b) > tolerance )
{ b=Matrix.multiply(Matrix.divide(df(f,x),H(f,x)),-ti);
  x=Matrix.add(x,b);
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
            " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
System.out.println();
return x;
}
public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
   Double ax=new Double(token.nextToken());
   a[j++]=ax.doubleValue();
   }
   return a;
}

    public static void main (String args[])
    {
            double [] x0=verigir();
            fa ff=new fa("a.txt",x0);
    double [] r1= continuationRK6(ff,x0,4);
    String s=" optimum süreklilik RK6 : \n"+Matrix.toStringT(r1);
    String s1="çok değişkenli süreklilik metodu ile optimizasyon : ";
     System.out.println(s1+s);
    System.exit(0);
    }
}
```

Lineer olmayan eğri uydurma örneği, Süreklilik metodu kullanılarak

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO5D1
end of file
çok değişkenli süreklilik metodu ile optimizasyon :  optimum süreklilik RK6 :
      22.098502588635580
      -1.064781834808273

> Terminated with exit code 0.
```

Genetik algoritmalar optimizasyon metodu kullanarak :

Program 6.11.4 Lineer olmayan eğri uydurma , Genetik algoritma metodu

```
import java.io.*;
import javax.swing.*;

    class fa extends f_xj
    {
            double xi[]; // independent variable data set
    double yi[]; //dependent variable data set
    double a[];  //fit function coefficient set
    int nn;
            public fa(String filename,double ia[])
    {
```

```
        //read the data to curvefit
        //get the data file and initial fit coefficient when class is defined
                    xi=new double[500];
        yi=new double[500];
        int n=ia.length;
        a=new double[n];
        seta(ia);
        int i=-1;
        try{
                    BufferedReader fin=new BufferedReader(new FileReader(filename));
        try {
        while(fin != null)
        {
        i++;
        xi[i]=Text.readDouble(fin);
        yi[i]=Text.readDouble(fin);
        }
        } catch(EOFException e_eof) {System.out.println("end of file"); }
        }catch(IOException e_io) {System.out.println("dosya bulunamadı"); }
    nn=i;
    a=ia;
    }

    public void seta(double ia[])
    {
                    //assign new fit coefficient set
                    for(int ii=0;ii<nn;ii++)
            a[ii]=ia[ii];
                }

                public double[] geta()
    {
                    // return fit coefficient set
                    return a;
                }

                double Ps(double x,double ai[])
    {
    //sample specific function
                // saturation pressure as a function of saturation temperature
    a=ai;
    double pi=ai[0]*(1.0-Math.exp(-ai[1]*x));
    //pi=Math.pow(10.0,pi);
    return pi; //kPa
    }

    public double func(double ai[])
    {
    double ff=0;
    double w;
    double yy;
     for(int i=0;i<nn;i++)
                    {w=Ps(xi[i],ai);
                    yy=yi[i];
     //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
                    w-=yy;
     ff+=w*w;
     //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
                    }
    return ff;
                }
}

class fb extends fa
{     public fb(String filename,double ia[])
    {super(filename,ia);}
```

```
    public double func(double ai[])
    {
    return -super.func(ai);
              }
}

class OPO19D
{
   public static void main(String args[]) throws IOException
   {
   int N=63;
            int POPSIZE=200;
   int MAXGENS=200;
   int NVARS=2;
   double PXOVER=0.3;
   double PMUTATION=0.02;
   String in_name="a.txt";

   Genetic1 xx=new Genetic1(POPSIZE,MAXGENS,NVARS,N,PXOVER,PMUTATION);
   double low[]=new double[NVARS];
   double high[]=new double[NVARS];
   low[0]=30.0;high[0]=50.0;
   low[1]=0.0;high[1]=1.0;

   double a[]=new double[NVARS];
   fb fbx=new fb("a.txt",a);
   xx.setPopulation(low,high,fbx);
   a=xx.calculate(fbx,false);
   double x0[]=new double[NVARS];
   for(int i=0;i<NVARS;i++) x0[i]=a[i];
   String s="genetik algoritma sonucu =\n"+Matrix.toStringT(a);
    String s2="Genetik algoritma  çok değişkenli fonksiyon optimizasyonu:";
    JOptionPane.showMessageDialog(null,s,s2,JOptionPane.PLAIN_MESSAGE);
   }
}
```

Lineer olmayan eğri uydurma , Genetik algoritma metodu

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" OPO19D
end of file
Genetik algoritma  çok değişkenli fonksiyon optimizasyonu:genetik algoritma sonucu =
      30.312499999995760
       0.999999999999994
  -413249.869777238750000



> Terminated with exit code 0.
```

Genetik algoritma metodunun maksimum bulan bir metod olduğunu, ve bu yüzden de – işaretiyle ters olarak çağrıldığını burada tekrar hatırlatmak istiyoruz.

Bu bölümde son olarak çok değişkenli ve katsayıları lineer olmıyan fonksiyonlara eğri uydurmamız gerektiğinde ne yapacağımıza değinelim. Lineer fonksiyonlarda hatırlıyacağınız gibi genel en küçük kareler metodunu metodu değiştirmeden sadece fonksiyonların oluşum prosesini kullanarak çok boyutlu fonksiyonlar için de kullanabileceğimizi anlatmıştık. Aynı şekilde lineer olmıyan metodlarımızı da çok boyutlu fonksiyonların eğri uydurmasında sadece fonksiyonu oluşturma sistemini değiştirerek olşturabiliriz. Örnek olarak

```
5.0  1.0 16.0
10.0 1.0 25.0
15.0 1.0 32.0
20.0 1.0 33.0
```

```
25.0 1.0 38.0
30.0 1.0 36.0
5.0  2.0 32.0
10.0 2.0 50.0
15.0 2.0 64.0
20.0 2.0 66.0
25.0 2.0 76.0
30.0 2.0 72.0
```

Verisini alalım. Eğri uydurmak istediğimiz fonksion da :
double pi=x[1]*ai[0]*(1.0-Math.exp(-ai[1]*x[0])); şeklinde olsun. burada ai katsayılar vektörü x[0] ve x[1] bağımsız değişkenlerdir. Veriye incelediğimizde üstteki örneklerle aynı katsayıları vermesi gerektiğini görebiliriz. Örneğimizi süreklilik optimizasyon metoduyla vereceğiz, ama daha önce de belirttiğimiz gibi tüm optimizasyon metodlarını kullanabiliriz.

Program 6.11.5 Lineer olmayan eğri uydurma , birden fazla bağımsız değişken için olan versiyon, Süreklilik optimizasyon metodu kullanılarak çözüm

```
// FSCO5D Süreklilik(Continuation-homotopy) optimizasyon
// Metod Referansı : Numerical Analysis, 8th edition Richard L. Burden, J. Douglas Faires 3
// sayfa 637
// bu versiyon fonksiyonların türevini sayısal olarak almaktadır.
// kullanıcının türev fonksiyonu tanımlaması gerekmez


import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

//============= Tanımı gereken fonksiyon ================
abstract class fi_xi
{
  // multifunction multi independent variable
  // vector of dependent variables are returned
  // example f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x) returns the value of f[0] and f[1]
  // as a two dimensional vector

  abstract double[] func(double x[]);
}

abstract class fij_xi
{
  // multifunction multi independent variable
  // for n independent variable n*n matrix of
  // dependent variables are returned
  // example f[0][0]=x[0]+Math.sin(x[1])   f[0][1]=x[0]-x[1]
  //         f[1][0]=x[0]*x[0]-x[1]        f[1][1]=Math.exp(x[0]+x[1]*x[1]
  // func(x) returns the value of f[0][0], f[0][1]
  //                              f[1][0], f[1][1]
  // as a two dimensional matrix
  abstract double[][] func(double x[]);
}

     //
     class fxa extends f_xj
     {
                double xi[][]; // m independent variable data set
     double yi[]; //dependent variable data set
```

```
    double a[];  //fit function coefficient set
    int nn;//number of data
    int m; //number of idependent variables xi
                public fxa(String filename,double ia[])
    {
      //read the data to curvefit
      //last column is treated as independent data set
      //get the data file and initial fit coefficient when class is defined
      int n=ia.length;
      a=new double[n];
      seta(ia);
      double b[][]=Text.readDoubleT(filename);
      m=b.length-1;
      yi=b[m];
      nn=yi.length;
      //System.out.println("m="+m);
      xi=new double[m][nn];
      for(int i=0;i<m;i++) {xi[i]=b[i];}
      yi=b[m];
      //System.out.println("b=\n"+Matrix.toString(Matrix.T(b)));
      //System.out.println("xi=\n"+Matrix.toString(xi));
      //System.out.println("yi=\n"+Matrix.toString(yi));
    nn=yi.length;
    a=ia;
    }

    public void seta(double ia[])
    {
                  //assign new fit coefficient set
                  for(int ii=0;ii<nn;ii++)
          a[ii]=ia[ii];
                }

                public double[] geta()
    {
                  // return fit coefficient set
                  return a;
                }               double Ps(double x[],double ai[])
    {
    // sample specific function m dimensional independent variable
    // independent variables : x[0],x[1]...x[m-1]
    a=ai;
    double pi=x[1]*ai[0]*(1.0-Math.exp(-ai[1]*x[0]));
    return pi; //kPa
    }



    public double func(double ai[])
    {
    double ff=0;
    double w;
    double yy;
    double xx[]=new double[xi.length];
     for(int i=0;i<nn;i++)
                  {   for(int j=0;j<m;j++) {xx[j]=xi[j][i];}
                                          w=Ps(xx,ai);
                  yy=yi[i];
      //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
                  w-=yy;
      ff+=w*w;
      //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
                  }
    return ff;
                }
}
```

```
public class FSCO5D2
{
// Süreklilik(Continuity-homotopy) metodu kullanarak denklem sisteminin köklerini bulmak
// Bu metod denklem sistemi çözümü için Matrix.java programını
// kullanmaktadır.


public static double[] multiply(double left,double[] right)
{
//multiplying a vector with a constant
int i;
int n=right.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
  {b[i]=left*right[i];}
return b;
}

public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right.length;
double[] b;
b=new double[m2];
 if(n1 != m2)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
   {
    b[ii]=0;
   }
  return b;
 }
 for(i=0;i<m1;i++)
   {
    b[i]=0;
    for(k=0;k<n1;k++)
              b[i]+=left[i][k]*right[k];
   }
return b;
//end of multiply of a matrix and a vector
}

public static double[] add(double[] left,double[] right)
{
//addition of two vectors
int n1=left.length;
int n2=right.length;
int nMax;
int i;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[];
b=new double[nMax];
for(i=0;i<n1;i++)
  {
   b[i]=b[i]+left[i];
  }
for(i=0;i<n2;i++)
  {
   b[i]=b[i]+right[i];
```

```
  }
return b;
//end of vector addition method
}

public static double[][] inv(double[][] a)
{
// INVERSION OF A MATRIX
// inversion by using gaussian elimination
// with full pivoting
int n=a.length;
int m=a[0].length;
double b[][];
b=new double[n][n];
int indxc[];
int indxr[];
double ipiv[];
indxc=new int[n];
indxr=new int[n];
ipiv=new double[n];
int i,j,k,l,ll,ii,jj;
int icol=0;
int irow=0;
double big,dum,pivinv,temp;
if(n!=m)
{
  System.out.println("Matrix must be square ");
                for(ii=0;ii<n;ii++)
                  for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
                return b;
}
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
     b[i][j]=a[i][j];
for(i=0;i<n;i++)
{
 big=0.0;
 for(j=0;j<n;j++)
 {
  if(ipiv[j] != 1)
    for(k=0;k<n;k++)
    {
              if(ipiv[k] == 0)
              {
                if(Math.abs(b[j][k]) >= big)
                {
                  big=Math.abs(b[j][k]);
                  irow=j;
                  icol=k;
                }
              }
              else if(ipiv[k] > 1 )
                {
                System.out.println("error : inverse of the matrix : singular matrix-1");
                for(ii=0;ii<n;ii++)
                  for(jj=0;jj<n;jj++)
                        b[ii][jj]=0.0;
                return b;
                }
    }
 }
 ++ ipiv[icol];
 if(irow != icol)
  for(l=0;l<n;l++)
  {
```

```
    temp=b[irow][l];
    b[irow][l]=b[icol][l];
    b[icol][l]=temp;
    }
  indxr[i]=irow;
  indxc[i]=icol;
  if(b[icol][icol] == 0.0)
  {
    System.out.println("error : inverse of the matrix : singular matrix-2");
                  for(ii=0;ii<n;ii++)
                    for(jj=0;jj<n;jj++)
                              b[ii][jj]=0.0;
    return b;
  }
  pivinv=1.0/b[icol][icol];
  b[icol][icol]=1.0;
  for(l=0;l<n;l++) b[icol][l] *=pivinv;
  for(ll=0;ll<n;ll++)
  if(ll != icol)
  {
    dum=b[ll][icol];
    b[ll][icol]=0.0;
    for(l=0;l<n;l++) b[ll][l]-= b[icol][l]*dum;
  }
}
for(l=n-1;l>=0;l--)
{
  if(indxr[l] != indxc[l])
     for(k=0;k<n;k++)
     {
       temp=b[k][indxc[l]];
       b[k][indxc[l]]=b[k][indxr[l]];
       b[k][indxr[l]]=temp;
     }
}
return b;
}

public static double SIGN(double a,double b) {return (b > 0.0 ? Math.abs(a) : -Math.abs(a));}
public static double MAX(double a,double b) {return (a > b ? a : b);}

public static double[] df(f_xj f,double xxi[])
{
//Gredyen vektörü
// f(x0,x1,x2,...xn) founksiyonunun türev vektörü
// [df/dx0, df/dx1,df/dx2,....,df/dxn]
// df/dxj  j=0...x.length
//
// bu fonksiyon bilgisayar tarafından hesaplanmaktadır
// kullanıcı tarafından tanımlanan f(x) fonksiyonunun
// sayısal olarak alınan türevidir.

//
double a[]=new double[xxi.length];
for(int i=0;i<xxi.length;i++)
   {
   a[i]=dfdx(f,xxi,i);
   }
return a;
}


public static double dfdx(f_xj f,double xxi[],int xref)
{ int i;
                double h=1.0e-3;
  double hh=1.0/h;
```

```
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
  for(i=0;i<xxi.length;i++)
  {
  if(i!=xref)
    {xp2h[i]=xxi[i];xph[i]=xxi[i];xm2h[i]=xxi[i];xmh[i]=xxi[i];}
  else {xp2h[i]=xxi[i]+2.0*h;xph[i]=xxi[i]+h;xm2h[i]=xxi[i]-2.0*h;xmh[i]=xxi[i]-h;}
  }
  fp2h=f.func(xp2h);
  fph=f.func(xph);
  fm2h=f.func(xm2h);
  fmh=f.func(xmh);
  return (-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
}
public static double d2fdx2(f_xj f,double xxi[],int xiref,int xjref)
{ //ikincitürev matrisi
  int i;
  double h=1.0e-4;
  double hh=1.0/h;
  int n=xxi.length;
  double ff;
  double xp2h[]=new double[xxi.length];
  double xph[]=new double[xxi.length];
  double xm2h[]=new double[xxi.length];
  double xmh[]=new double[xxi.length];
  double fp2h,fph,fm2h,fmh;
     for(int j=0;j<n;j++)
     {if(j!=xjref) {xp2h[j]=xxi[j];xph[j]=xxi[j];xm2h[j]=xxi[j];xmh[j]=xxi[j];}
      else {xp2h[j]=xxi[j]+2.0*h;xph[j]=xxi[j]+h;xm2h[j]=xxi[j]-2.0*h;xmh[j]=xxi[j]-h;}
     }
     fp2h=dfdx(f,xp2h,xiref);
     fph=dfdx(f,xph,xiref);
     fm2h=dfdx(f,xm2h,xiref);
     fmh=dfdx(f,xmh,xiref);
     ff=(-fp2h+8.0*fph-8.0*fmh+fm2h)/12.0*hh;
  return ff;
}

public static double[][]H(f_xj f,double x[])
{
  int n=x.length;
  double dy[][]=new double[n][n];;
  for(int ii=0;ii<n;ii++)
  {
    for(int jj=0;jj<n;jj++)
    {   //Hessian matrisi
        dy[ii][jj]=d2fdx2(f,x,ii,jj);
    }
  }
  return dy;
}
public static double[] continuationRK4(f_xj f,double x[],int N)
{
//=====================================================
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi J de tanımlanacaktır
// Homotopy, continuation (devamlılık) metodu ile lineer olmıyan denklem sistemi çözümü
// 4.derece Runge-Kutta denklemi ile
// yi+1 = yi + (1/6)*( k1 + 2k2 +2k3+k4)h
// k1=f(xi,yi)
// k2=f(xi+0.5h,yi+0.5k1h)
// k3=f(xi+0.5h,yi+0.5k2h)
```

```
// k4=f(xi+h,yi +k3h)
//=====================================================
// x  : bağımsız değişken vektörü
// y  : bağımlı değişken vektörü
// dy : bağımlı değişkenin türev fonksiyonları vektörü
// N  : Runge-Kutta diferansiyel denklem noktasayısı

int i;
int nmax=400;
double tolerance=1.0e-20;
int n=x.length;
double h=1.0/(double)N;
double b[]=new double[n];
double x1[]=new double[n];
double k[][]=new double[4][n];
double A[][]=new double[n][n];
b=multiply(-h,df(f,x));
for(i=0;i<N;i++)
{
x1=x;
A=H(f,x1);
k[0]=multiply(inv(A),b);
x1=add(x,multiply(0.5,k[0]));
A=H(f,x1);
k[1]=multiply(inv(A),b);
x1=add(x,multiply(0.5,k[1]));
A=H(f,x1);
k[2]=multiply(inv(A),b);
x1=add(x,k[2]);
A=H(f,x1);
k[3]=multiply(inv(A),b);
for(int j=0;j<n;j++) {x[j]=x[j]+1.0/6.0*(k[0][j]+2.0*k[1][j]+2.0*k[2][j]+k[3][j]);}
}
return x;
}

public static double[] continuationRK6(f_xj f,double x[],int N)
{
//=====================================================
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi J de tanımlanacaktır
// Homotopy, continuation (devamlılık) metodu ile lineer olmıyan denklem sistemi çözümü
// 6. derece Runge-Kutta denklemi ile
// yi+1 = yi + (1/90)*( 7k1 + 32k3 +12k4+32k5+7k6)h
// k1=f(xi,yi)
// k2=f(xi+0.25h , yi+0.25k1h)
// k3=f(xi+0.25h , yi+0.125k1h+0.125k2h)
// k4=f(xi+0.5h , yi - 0.5k2h+k3h)
// k5=f(xi+0.75h , yi + (3/16)k1h+(9/16)k4h)
// k6=f(xi+h , yi - (3/7)k1h+(2/7)k2h+(12/7)k3h - (12/7)k4h+(8/7)k5h)
//=====================================================
// x  : bağımsız değişken vektörü
// y  : bağımlı değişken vektörü
// dy : bağımlı değişkenin türev fonksiyonları vektörü
// N  : Runge-Kutta diferansiyel denklem noktasayısı
int i;
int nmax=400;
double tolerance=1.0e-20;
int n=x.length;
double h=1.0/(double)N;
double b[]=new double[n];
double x1[]=new double[n];
double k[][]=new double[6][n];
double A[][]=new double[n][n];
b=multiply(-h,df(f,x));
```

```
for(i=0;i<N;i++)
{
x1=x;
A=H(f,x1);
// k1=f(xi,yi)
k[0]=multiply(inv(A),b);
x1=add(x,multiply(0.25,k[0]));
A=H(f,x1);
// k2=f(xi+0.25h , yi+0.25k1h)
k[1]=multiply(inv(A),b);
x1=add(x,add(multiply(0.125,k[0]),multiply(0.125,k[1])));
A=H(f,x1);
// k3=f(xi+0.25h , yi+0.125k1h+0.125k2h)
k[2]=multiply(inv(A),b);
x1=add(x,add(multiply(-0.5,k[1]),k[2]));
A=H(f,x1);
// k4=f(xi+0.5h , yi - 0.5k2h+k3h)
k[3]=multiply(inv(A),b);
x1=add(x,add(multiply((3.0/16.0),k[0]),multiply((9.0/16.0),k[3])));
A=H(f,x1);
// k5=f(xi+0.75h , yi + (3/16)k1h+(9/16)k4h)
k[4]=multiply(inv(A),b);
x1=add(x,
      add(multiply((-3.0/7.0),k[0]),add(multiply((2.0/7.0),k[1]),
      add(multiply((12.0/7.0),k[2]),
         add(multiply((-12.0/7.0),k[3]),multiply((8.0/7.0),k[4])))));
A=H(f,x1);
// k6=f(xi+h , yi - (3/7)k1h+(2/7)k2h+(12/7)k3h - (12/7)k4h+(8/7)k5h)
k[5]=multiply(inv(A),b);
// yi+1 = yi + (1/90)*( 7k1 + 32k3 +12k4+32k5+7k6)h
for(int j=0;j<n;j++) {x[j]=x[j]+1.0/90.0*(7.0*k[0][j]+32.0*k[2][j]+12.0*k[3][j]+32.0*k[4][j]+7.0*k[5][j]);}
}
return x;
}

public static double[] newton_continuationRK4(f_xj f,double x[],int N)
{
// lineer olmıyan denklem siteminin kökleri
// fj(xi)=0 lineer olmıyan denklem sistemi için çözüm matrisi
// türev matrisi dfj/dxi de tanımlanacaktır
// Newton-Raphson metodu ile lineer olmıyan denklem sistemi çözümü
//ti :önem katsayısı
//x bağımsız değişken vektörü
//y bağımlı değişken vektörü
//dy bağımlı değişkenin türev fonksiyonları vektörü
x=continuationRK4(f,x,4);
double ti=1.0;
int i;
int nmax=400;
double tolerance=1.0e-10;
int n=x.length;
double b[];
b=new double[n];
for(i=0;i<n;i++)
{
b[i]=1.0;
}
i=0;
while( i++ < nmax && Matrix.abs(b) > tolerance )
{ b=Matrix.multiply(Matrix.divide(df(f,x),H(f,x)),-ti);
  x=Matrix.add(x,b);
}
if(i >= nmax) JOptionPane.showMessageDialog(null,"Uyarı maximum iterasyon sayısı aşıldı \n"+
            " çözüm geçerli olmıyabilir","MAKSİMUM ITERASYON SAYISI UYARISI",JOptionPane.WARNING_MESSAGE);
System.out.println();
return x;
```

```
}
public static double[] verigir()
{
   String s=JOptionPane.showInputDialog("kök tahmin değerlerini arada bir boşluk bırakarak giriniz: ");
   StringTokenizer token=new StringTokenizer(s);
   int n=token.countTokens()-1;
   int m=n+1;
   double a[]=new double[m];
   int j=0;
   while(token.hasMoreTokens())
   {
   Double ax=new Double(token.nextToken());
   a[j++]=ax.doubleValue();
   }
   return a;
}

    public static void main (String args[])
    {
              double [] x0=verigir();
              fxa ff=new fxa("a.txt",x0);
    double [] r1= continuationRK6(ff,x0,4);
    String s=" optimum süreklilik RK6 : \n"+Matrix.toStringT(r1);
    String s1="çok değişkenli süreklilik metodu ile optimizasyon : ";
    JOptionPane.showMessageDialog(null,s,s1,JOptionPane.PLAIN_MESSAGE);
    System.exit(0);
    }
}
```

Lineer olmayan eğri uydurma , birden fazla bağımsız değişken için olan versiyon, Süreklilik optimizasyon metodu kullanılarak çözüm

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" FSCO5D2
xi=
      5.000000000000000      10.000000000000000      15.000000000000000      20.000000000000000
25.000000000000000      30.000000000000000       5.000000000000000      10.000000000000000
15.000000000000000      20.000000000000000      25.000000000000000      30.000000000000000
      1.000000000000000       1.000000000000000       1.000000000000000       1.000000000000000
1.000000000000000       1.000000000000000       2.000000000000000       2.000000000000000
2.000000000000000       2.000000000000000       2.000000000000000       2.000000000000000

yi=
     16.000000000000000      25.000000000000000                32.000000000000000
33.000000000000000           38.000000000000000                36.000000000000000
32.000000000000000           50.000000000000000                64.000000000000000
66.000000000000000           76.000000000000000                72.000000000000000

çok değişkenli süreklilik metodu ile optimizasyon :  optimum süreklilik RK6 :
     22.098529523731138
     -1.064865262490600

> Terminated with exit code 0.
```

İkinci bir örnek olarak gerçek bir mühendislik problemini inceleyelim:

Bir araştırma problemi olarak dilim olarak kesilmiş elmanın ısı pompası kullanılarak kurutulması incelenmektedir. Verimiz t zaman(saat), h dilim kalınlığı (mm) ve P ürünün kalan nem oranı(kg nem/kg yaş elma nemi) olarak lineer olmayan aşağıdaki formda bir eğri uydurulmak istenmektedir.

$\mathbf{a=a_0+a_1h}$

$\mathbf{k=a_2+a_3h}$

$\mathbf{P=ae^{(-kt)}}$

Verimiz

0	6	1.000	3	8	0.332	4	10	0.328
0.5	6	0.736	3.5	8	0.276	4.5	10	0.287
1	6	0.603	4	8	0.229	5	10	0.251
1.5	6	0.482	4.5	8	0.192	5.5	10	0.218
2	6	0.390	5	8	0.160	6	10	0.185
2.5	6	0.307	5.5	8	0.128	6.5	10	0.161
3	6	0.245	6	8	0.102	7	10	0.138
3.5	6	0.192	6.5	8	0.083	7.5	10	0.117
4	6	0.143	7	8	0.067	8	10	0.101
4.5	6	0.108	7.5	8	0.055	8.5	10	0.086
5	6	0.081	8	8	0.046	9	10	0.071
5.5	6	0.059	0	10	1.000	9.5	10	0.063
6	6	0.044	0.5	10	0.824	10	10	0.056
0	8	1.000	1	10	0.718	10.5	10	0.051
0.5	8	0.765	1.5	10	0.618	11	10	0.045
1	8	0.649	2	10	0.542			
1.5	8	0.541	2.5	10	0.474			
2	8	0.456	3	10	0.424			
2.5	8	0.382	3.5	10	0.372			

Şeklindedir. Bu veriye üstteki denkleme uygun veri uyduralım.

Program 6.11.7 Lineer olmayan eğri uydurma , birden fazla bağımsız değişken için olan versiyon, Nelder-Mead optimizasyon metodu kullanılarak çözüm, elma kurutma problemi

```
import java.io.*;
import javax.swing.*;

    class fxa extends f_xj
    {
            double xi[][]; // m independent variable data set
    double yi[]; //dependent variable data set
    double a[];  //fit function coefficient set
    int nn;//number of data
    int m; //number of idependent variables xi
    double xmax;
            public fxa(String filename,double ia[])
    {
      //read the data to curvefit
      //last column is treated as independent data set
      //get the data file and initial fit coefficient when class is defined
      int n=ia.length;
      a=new double[n];
      seta(ia);
      xmax=-1.0e99;
      double b[][]=Text.readDoubleT(filename);
      m=b.length-1;
      yi=b[m];
      nn=yi.length;
      //System.out.println("m="+m);
      xi=new double[m][nn];
```

```
      for(int i=0;i<nn;i++) {
                xi[0][i]=b[0][i];if(xi[0][i]>xmax) {xmax=xi[0][i];}
      xi[1][i]=b[1][i];
                yi[i]=b[m][i];
      }
      //System.out.println("b=\n"+Matrix.toString(Matrix.T(b)));
      System.out.println("xi=\n"+Matrix.toString(xi));
      System.out.println("yi=\n"+Matrix.toString(yi));
    nn=yi.length;
    a=ia;
    }

    public void seta(double ia[])
    {
                //assign new fit coefficient set
                for(int ii=0;ii<nn;ii++)
          a[ii]=ia[ii];
              }

              public double[] geta()
    {
                // return fit coefficient set
                return a;
              }
     double Ps(double x[],double ai[])
    {
    // sample specific function m dimensional independent variable
    // independent variables : x[0],x[1]...x[m-1]
    // x[0] t
    // x[1] kalınlık
    // ai[0] a
    //
    a=ai;
    double t=x[0];
    double h=x[1];
    double a=ai[0]+ai[1]*h;
    double k=ai[2]+ai[3]*h;
    double pi=a*Math.exp(-k*t);
    return pi; //kPa
    }

    public double func(double ai[])
    {
    double ff=0;
    double w;
    double yy;
    double xx[]=new double[xi.length];
     for(int i=0;i<nn;i++)
                {   for(int j=0;j<m;j++) {xx[j]=xi[j][i];}
                                        w=Ps(xx,ai);
                yy=yi[i];
      //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
                w-=yy;
      ff+=w*w;
      //System.out.println("x="+xi[i]+"w="+w+"yy="+yy+"ff="+ff);
                }
    return ff;
              }
}

public class SCO11H2
{//2. Nelder ve Mead metodu (simpleks arama metodu)
//_____________________________________________________________________
public static double[] nelder(f_xj fnelder,double a[],double da[],int maxiteration,double tolerance,int printlist)
{
double x[][]=new double[a.length+1][a.length];
```

```
for(int i=0;i<x.length;i++)
  {for(int j=0;j<x[0].length;j++)
   {if(i==j){x[i][j]=a[j]+da[j];}
    else    {x[i][j]=a[j]; }
   }
  }
  // Nelder mead çok boyutlu simplex minimizasyon metodu
  //  Nelder & Mead 1965 Computer J, v.7, 308-313.

  // Giriş değişkenleri tanımlaması
  // fnelder : abstract çok boyutlu fonksiyon f(x)
  // x : for n boyutlu n+1 simlex noktasını içerir bağımsız değişken seti
  // maxiteration : maximum iterasyon sayısı
  // tolerance :
      int NDIMS = x.length-1;
      int NPTS = x.length;
      int FUNC = NDIMS;
      int ncalls = 0;
        ////// başlangıç simplexini oluştur ///////////////////
        double p[][]=new double[NPTS][NPTS]; // [row][col] = [whichvx][coord,FUNC]
        double z[]=new double[NDIMS];
        double best = 1E99;
        ///////////////// iilk fonksiyon değerlerini hesapla /////////////////
        for (int i=0; i<NPTS; i++)
        {
          for (int j=0; j<NDIMS; j++)
              {p[i][j] = x[i][j];}
          p[i][NDIMS] = fnelder.func(p[i]);
        }
        int iter=0;
        for (iter=1; iter<maxiteration; iter++)
        {
          ///////////  lo, nhi, hi noktalarını tanımla ///////////////
          int  ilo=0, ihi=0, inhi = -1; // -1 means missing
          double flo = p[0][FUNC];
          double fhi = flo;
          double pavg,sterr;
          for (int i=1; i<NPTS; i++)
          {
            if (p[i][FUNC] < flo)
              {flo=p[i][FUNC]; ilo=i;}
            if (p[i][FUNC] > fhi)
              {fhi=p[i][FUNC]; ihi=i;}
          }
          double fnhi = flo;
          inhi = ilo;
          for (int i=0; i<NPTS; i++)
            if ((i != ihi) && (p[i][FUNC] > fnhi))
              {fnhi=p[i][FUNC]; inhi=i;}
          ////////// çıkış kriteri ///////////////
          if ((iter % 4*NDIMS) == 0)
          {
            //yi nin standart hata kriteri set değerinden (tolerance)
            // küçük olmalı
            // ortalama değeri hesapla (en büyük değer de dahil olmak üzere)
            pavg=0;
            for(int i=0;i<NPTS;i++)
              pavg+=p[i][FUNC];
            pavg/=NPTS;
            double tot=0;
            if(printlist!=0)
            {  System.out.print(iter);
              for (int j=0; j<=NDIMS; j++)
                 { System.out.print(p[ilo][j]+" ");}
              System.out.println("");
            }
```

```
    for(int i=0;i<NPTS;i++)
    { tot=(p[i][FUNC]-pavg)*(p[i][FUNC]-pavg);}
    sterr=Math.sqrt(tot/NPTS);
    //if(sterr < tolerance)
      { for (int j=0; j<NDIMS; j++)
          { z[j]=p[ilo][j];}
        //break;
      }
    best = p[ilo][FUNC];
}

///// ave[] vektorünü en büyük değeri hariç tutarak hesapla //////

double ave[] = new double[NDIMS];
for (int j=0; j<NDIMS; j++)
  ave[j] = 0;
for (int i=0; i<NPTS; i++)
  if (i != ihi)
    for (int j=0; j<NDIMS; j++)
      ave[j] += p[i][j];
for (int j=0; j<NDIMS; j++)
   ave[j] /= (NPTS-1);


///////// yansıt ////////////////

double r[] = new double[NDIMS];
for (int j=0; j<NDIMS; j++)
  r[j] = 2*ave[j] - p[ihi][j];
double fr = fnelder.func(r);

if ((flo <= fr) && (fr < fnhi))  // in zone: accept
{
    for (int j=0; j<NDIMS; j++)
      p[ihi][j] = r[j];
    p[ihi][FUNC] = fr;
    continue;
}

if (fr < flo)  //// genişlet, else kabul et
{
    double e[] = new double[NDIMS];
    for (int j=0; j<NDIMS; j++)
      e[j] = 3*ave[j] - 2*p[ihi][j];
    double fe = fnelder.func(e);
    if (fe < fr)
    {
      for (int j=0; j<NDIMS; j++)
        p[ihi][j] = e[j];
      p[ihi][FUNC] = fe;
      continue;
    }
    else
    {
      for (int j=0; j<NDIMS; j++)
        p[ihi][j] = r[j];
      p[ihi][FUNC] = fr;
      continue;
    }
}

////////////// daralt:

if (fr < fhi)
{
    double c[] = new double[NDIMS];
```

```
                for (int j=0; j<NDIMS; j++)
                  c[j] = 1.5*ave[j] - 0.5*p[ihi][j];
                double fc = fnelder.func(c);
                if (fc <= fr)
                {
                   for (int j=0; j<NDIMS; j++)
                     p[ihi][j] = c[j];
                   p[ihi][FUNC] = fc;
                   continue;
                }
                else   /////// daralt
                {
                   for (int i=0; i<NPTS; i++)
                     if (i != ilo)
                     {
                        for (int j=0; j<NDIMS; j++)
                          p[i][j] = 0.5*p[ilo][j] + 0.5*p[i][j];
                        p[i][FUNC] = fnelder.func(p[i]);
                     }
                   continue;
                }
             }

             if (fr >= fhi)   ///
             {
                double cc[] = new double[NDIMS];
                for (int j=0; j<NDIMS; j++)
                  cc[j] = 0.5*ave[j] + 0.5*p[ihi][j];
                double fcc = fnelder.func(cc);
                if (fcc < fhi)
                {
                   for (int j=0; j<NDIMS; j++)
                     p[ihi][j] = cc[j];
                   p[ihi][FUNC] = fcc;
                   continue;
                }
                else    /////////
                {
                   for (int i=0; i<NPTS; i++)
                     if (i != ilo)
                     {
                        for (int j=0; j<NDIMS; j++)
                          p[i][j] = 0.5*p[ilo][j] + 0.5*p[i][j];
                        p[i][FUNC] = fnelder.func(p[i]);
                     }
                }
             }
          }
       return z;
}
    public static double[] nelder(f_xj fnelder,double a[],double da[],double tolerance)
    {return nelder(fnelder,a,da,300,tolerance,0);}

    public static double[] nelder(f_xj fnelder,double a[],double da[])
    {return nelder(fnelder,a,da,500,1.0e-20,0);}

    public static double[] nelder(f_xj fnelder,double x0[])
    {  double [] dx0=new double[x0.length];
      for(int i=0;i<x0.length;i++) dx0[i]=0.2*x0[i];
      return nelder(fnelder,x0,dx0,500,1.0e-15,0);
    }

    public static double[] nelder(f_xj fnelder,double x0[],double tolerance)
    {
     double [] dx0=new double[x0.length];
     for(int i=0;i<x0.length;i++) dx0[i]=0.2*x0[i];
```

```
            return nelder(fnelder,x0,dx0,tolerance);
  }

  public static double[] nelder(f_xj fnelder,double x0[],int maxiteration,double tolerance)
  {
    double [] dx0=new double[x0.length];
    for(int i=0;i<x0.length;i++) dx0[i]=0.2*x0[i];
    return nelder(fnelder,x0,dx0,maxiteration,tolerance,0);
  }

 public static void main(String args[]) throws IOException
 {
 //String in_name=JOptionPane.showInputDialog(" veri dosyasının isminigiriniz : ");
 String in_name="kalinlik1.txt";
 double a[];
 double x[][],y[]; //uydurulan veri
 double xu1[][],yu1[]; //uydurulan veri
 double xu2[][],yu2[];
 double xu3[][],yu3[];
 a=new double[4];
 a[0] = 1.0;
 a[1] = 1.0;
 a[2] = 1.0;
 a[3] = 1.0;
 fxa f=new fxa(in_name,a);
 int i=0,j=0;
 double p[]=nelder(f,a);
 f=new fxa(in_name,p);
 double tmax=f.xmax;
 double dt=0.1; //ZAMAN ARALIĞI
 int n=(int)(tmax/dt);
 System.out.println("length="+f.nn+"tmax="+tmax+"n="+n);
 xu1=new double[f.m][n];
 yu1=new double[n];
 xu2=new double[f.m][n];
 yu2=new double[n];
 xu3=new double[f.m][n];
 yu3=new double[n];
 double z[]=new double[2];
 for(i=0;i<n;i++)
 {
 xu1[0][i]=dt*i;
 xu1[1][i]=6;
 z[0]=xu1[0][i];
 z[1]=xu1[1][i];
 yu1[i]=f.Ps(z,p);
 System.out.println("x0="+xu1[0][i]+"x1="+xu1[1][i]+"y="+yu1[i]);
 }
 for(i=0;i<n;i++)
 {
 xu2[0][i]=dt*i;
 xu2[1][i]=8;
 z[0]=xu2[0][i];
 z[1]=xu2[1][i];
 yu2[i]=f.Ps(z,p);
 System.out.println("x0="+xu2[0][i]+"x1="+xu2[1][i]+"y="+yu2[i]);
 }
 for(i=0;i<n;i++)
 {
 xu3[0][i]=dt*i;
 xu3[1][i]=10;
 z[0]=xu3[0][i];
 z[1]=xu3[1][i];
 yu3[i]=f.Ps(z,p);
 System.out.println("x0="+xu3[0][i]+"x1="+xu3[1][i]+"y="+yu3[i]);
 }
```

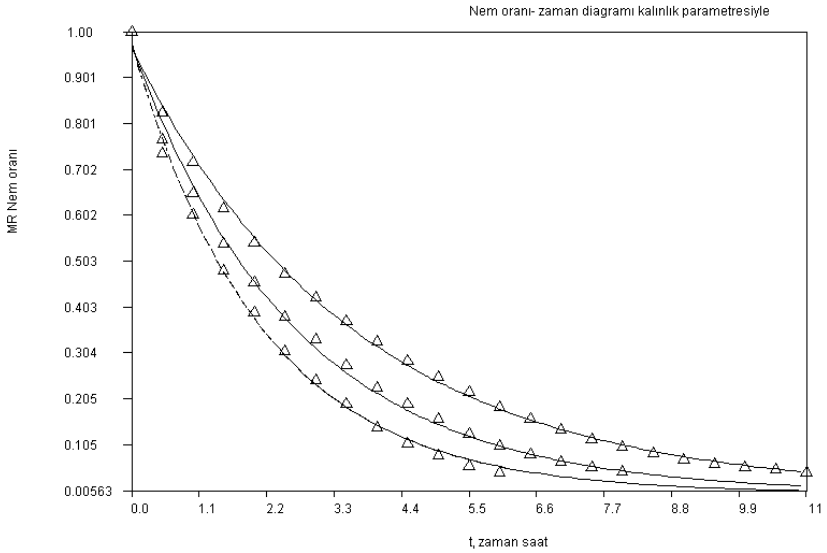
```
    Plot pp=new Plot(f.xi[0],f.yi);
    pp.setPlabel("Nem oranı- zaman diagramı kalınlık parametresiyle");
    pp.setXlabel("t, zaman saat");
    pp.setYlabel("MR Nem oranı ");
    pp.setPlotType(0,24);
    pp.addData(xu1[0],yu1);
    pp.setPlotType(1,0);
    pp.addData(xu2[0],yu2);
    pp.setPlotType(1,1);
    pp.addData(xu3[0],yu3);
    pp.setPlotType(1,2);

    //pp.addData(x,y);
    //pp.setPlotType(1,24);// üçgen olarak çiz
    pp.plot();

    String s1=" optimizasyon değeri  : \n"+Matrix.toStringT(p)+"\n";
    s1+="orijinal fonksiyon değeri = "+f.func(a)+"\n";
    s1+="fonksiyon değeri = "+f.func(p);
    s1+="standart sapma= "+Math.sqrt(f.func(p)/(f.nn-1));
    String s2="Nelder-Mead en küçük kareler lineer olmiyan eğri uydurma : ";
    System.out.println(s1);
    }
}
```

Elde edilen eğri :

a_0= 0.990155880325089
a_1= -0.002517036051923
a_2= 0.765464843543500
a_3= -0.048774936274541

veri ve eğrinin grafik gösterimi :

Çıktı 6.10-7 Lineer olmayan eğri uydurma , birden fazla bağımsız değişken için olan versiyon, Nelder-Mead optimizasyon metodu kullanılarak çözüm, elma kurutma problemi uyarlaması



(bolum6_013.jpg,Resimaltı: **Liner olmıyan eğri uydurma örnek problem grafik çıktısı**)

Bu bölümdeki metodlarımıza lineer **olmayan katsayılı genel en küçük kareler metodu** ismini de verebiliriz, temel olarak eğri uydurmak için mümkün olan en genel bir tanımı vermiş olduk. Lineer olmıyan katsayılı eğri uydurma

metodlarımızla istersek lineer katsayılı problemleri de çözebileceğimizi de hatırlatalım, ancak lineer olarak tanımlanmış bir setin direk olarak çözülmesinden dolayı tanımı daha kolaydır, lineer olmıyan metodlar yapısı icabı genelde iteratif prosesler kullanır.

6.12 Eğri uydurmada indirek (parametrik) değişken kullanımı

Tablo 6.11-1 deki x y değerleri deney verisi olarak bulunmuş olsun. Bu değerlere bir eğri uydurmaya çalıştığımızda klasik eğri uydurma metodlarının cevap vermediğini göreceğiz. Çünki aynı x noktasına karşı gelen 1 den fazla y verisi bulunmaktadır. Bu durumda bir eğri uydurmak istediğimizde sete suni bir ek parametre ilave edebiliriz. Örneğin tablodaki t parametresini ilave edelim. (t,x) ve (t,y) tekrarlanmayan iki eğri seti olduğundan bu iki sete eğri uydurmamız mümkün olabilecektir. Bunu yaptıktan sonra değerlendirme aşamasında seti tekrar (x,y) setine dönüştürebiliriz.

Tablo 6.12.1 Girdi verisi

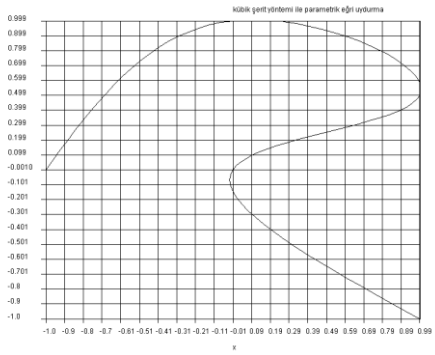
i	0	1	2	3	4
t	0	1	2	3	4
x	-1	0	1	0	1
y	0	1	0.5	0	-1

Program 6.12.1 İndirek parametrik değişken kullanımı, şerit interpolasyonu

```
İmport java.io.*;
import javax.swing.*;

class SCO11F3
{

  public static double [] thomas(double a[][],double r[])
  {
  //
  int n=a.length;
  double f[]=new double[n];
  double e[]=new double[n];
  double g[]=new double[n];
  double x[]=new double[n];
  for(int i=0;i<n;i++)     {f[i]=a[i][i];}
  for(int i=0;i<(n-1);i++) {g[i]=a[i][i+1];}
  for(int i=0;i<(n-1);i++) {e[i+1]=a[i+1][i];}
  for(int k=1;k<n;k++)
    {e[k]=e[k]/f[k-1];
     f[k]=f[k]-e[k]*g[k-1];
    }
  for(int k=1;k<n;k++)
    {r[k]=r[k]-e[k]*r[k-1];
    }
  x[n-1]=r[n-1]/f[n-1];
  for(int k=(n-2);k>=0;k--)
    {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
  return x;
  }

  public static double [] thomas(double f[],double e[],double g[],double r[])
  {
  int n=f.length;
  double x[]=new double[n];
  for(int k=1;k<n;k++)
    {e[k]=e[k]/f[k-1];
     f[k]=f[k]-e[k]*g[k-1];
    }
  for(int k=1;k<n;k++)
```

```
   {r[k]=r[k]-e[k]*r[k-1];
   }
x[n-1]=r[n-1]/f[n-1];
for(int k=(n-2);k>=0;k--)
   {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
return x;
}


public static double [][] cubic_spline(double xi[],double yi[],double c0,double cn)
{
int n=xi.length;
double h[]=new double[n];
double w[]=new double[n];
double f[]=new double[n];
double e[]=new double[n];
double g[]=new double[n];
double d[]=new double[n];
double x[]=new double[n];
double S[][]=new double[4][n];
int k;
for(k=0;k<(n-1);k++)
   {h[k]=xi[k+1]-xi[k];
    w[k]=(yi[k+1]-yi[k])/h[k];
               }
d[0]=c0;
d[n-1]=cn;
for(k=1;k<(n-1);k++)
   {d[k]=6.0*(w[k]-w[k-1]);}
f[0]=1.0;
f[n-1]=1.0;
g[0]=0.0;
g[n-1]=0.0;
e[0]=0.0;
e[n-1]=0.0;
for(k=1;k<(n-1);k++)
{f[k]=2.0*(h[k]+h[k-1]);e[k]=h[k-1];g[k]=h[k];}
S[2]=thomas(f,e,g,d);
S[3]=xi;
for(k=0;k<(n-1);k++)
  {S[0][k]=(6.*yi[k+1]-h[k]*h[k]*S[2][k+1])/(6.0*h[k]);
   S[1][k]=(6.*yi[k]-h[k]*h[k]*S[2][k])/(6.0*h[k]);
  }
return S;
}

public static double funcSpline(double S[][],double x)
{
int n=S[0].length;
double xx1=0;
double xx2=0;
double y=0;
double hk=0;
for(int k=0;k<(n-1);k++)
   {if(S[3][k]<=x && x<=S[3][k+1])
    {hk=(S[3][k+1]-S[3][k]);
     xx1=(x-S[3][k]);
     xx2=(S[3][k+1]-x);
     y=S[0][k]*xx1+S[1][k]*xx2+(xx1*xx1*xx1*S[2][k+1]+xx2*xx2*xx2*S[2][k])/(6.0*hk);
     break;
     }
   }
     if(y==0 && S[3][n-2]<=x )
     {
                int k=n-2;
                hk=(S[3][k+1]-S[3][k]);
```

```
      xx1=(x-S[3][k]);
      xx2=(S[3][k+1]-x);
      y=S[0][k]*xx1+S[1][k]*xx2+(xx1*xx1*xx1*S[2][k+1]+xx2*xx2*xx2*S[2][k])/(6.0*hk);
      }
 return y;
 }

public static double[][]  funcSpline(double xi[],double yi[],int aradegersayisi)
{
              //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
              // ara değer sayısı 2 dir
              int n=xi.length;
              int nn=(n-1)*(aradegersayisi+1)+1;
              double z[][]=new double[2][nn];
  double S[][]=cubic_spline(xi,yi,0,0);
  double dx=0;
  int k=0;
  int i;
  for(i=0;i<(n-1);i++)
    {  z[0][k]=xi[i];z[1][k]=funcSpline(S,z[0][k]);k++;
      for(int j=0;j<aradegersayisi;j++)
        {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
      z[0][k]=z[0][k-1]+dx;z[1][k]=funcSpline(S,z[0][k]);k++;}
    }
    z[0][k]=xi[i];z[1][k]=funcSpline(S,z[0][k]);
  return z;
}

 public static void main(String args[]) throws IOException
 {
 double x[];
 double y[];
 String s1="a.txt";
 JFileChooser fc=new JFileChooser();
 if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName(); }
 double a[][]=Text.readDoubleT(s1);
 x=a[0];
 y=a[1];
 int n=x.length;
 double t[]=new double [n];
 for(int i=0;i<n;i++) t[i]=i;
 double z1[][]=funcSpline(t,x,20);
 double z2[][]=funcSpline(t,y,20);
 System.out.println(" z1\n"+Matrix.toString(z1));
 System.out.println(" z2\n"+Matrix.toString(z1));
 Plot pp=new Plot(z1[1],z2[1]);
 pp.setPlabel("kübik şerit yöntemi ile parametrik eğri uydurma");
 pp.setXlabel("x");
 pp.setYlabel("y");
 pp.setPlotType(0,0);
 pp.addData(z1[1],z2[1]);
 pp.plot();
 }
 }
```



(bolum6_014.jpg,Resimaltı: **İndirek parametrik değişken kullanımı, şerit interpolasyonu**)

6.13 Hareketli en küçük kareler metodu

$\sum_{k=0}^{m} a_j^{(m)} w_i \left[\sum_{i=1}^{n} x_i^{j+k} \right] = \sum_{i=1}^{n} w_i f_i x_i^k \qquad k = 0,.....,m$ denklemini polinomlar için en küçük kareler denklemi olarak tanımlamış ve bu denklemi ağırlık fonksiyonu w_i=1=sabit olmak üzere çözmüştük. Bu çözümde polinomun üssünü artırmanın her zaman için çözüm vermediğini görebiliriz. Örneğin

x	1.47	1.83	3.02	3.56	5.86	8.75	9.45
f	2.09	1.92	2.19	2.64	3.19	3.13	3.61

Verisine en küçük kareler polinom metodu ile 3.üncü,4üncü ve 5inci dereceden eğriler uyduralım.

Program 6.13.1 3,4 ve 5inci derece polinom eğri uydurma

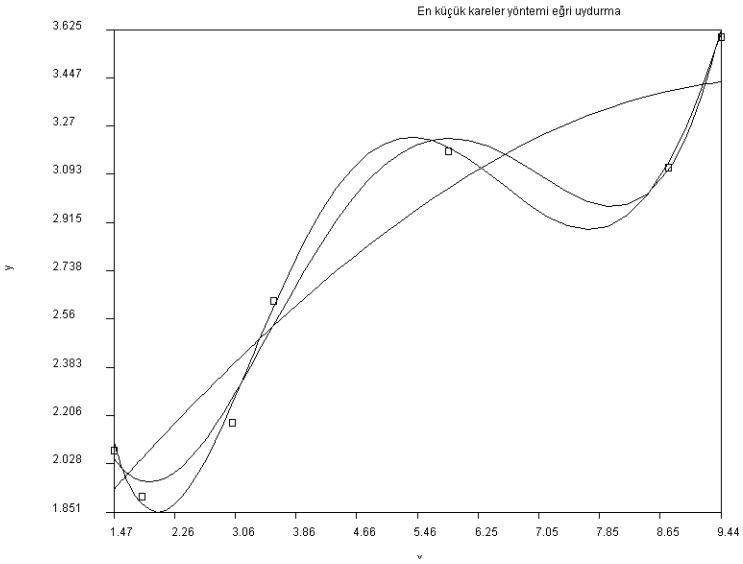
```
import java.io.*;
import javax.swing.*;

class SCO11B4
{
//en kucuk kareler metodu
  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
              p=k;
              buyuk=Math.abs(a[k][k]);
              for(ii=k+1;ii<n;ii++)
              {  dummy=Math.abs(a[ii][k]);
                if(dummy > buyuk) {buyuk=dummy;p=ii;}
              }
              if(p!=k)
              { for(jj=k;jj<n;jj++)
                { dummy=a[p][jj];
                  a[p][jj]=a[k][jj];
                  a[k][jj]=dummy;
                }
                dummy=b[p];
                b[p]=b[k];
```

```
                b[k]=dummy;
               }
              //gauss elemeyi çözme
   for(i=k+1;i<n;i++)
   {  carpan=a[i][k]/a[k][k];
     a[i][k]=0;
     for(j=k+1;j<n;j++)
     {   a[i][j]-=carpan*a[k][j]; }
        b[i]   =b[i]   -carpan*b[k];
              }
 }
 //geriye doğru yerine koyma
 x[n-1]=b[n-1]/a[n-1][n-1];
 for(i=n-2;i>=0;i--)
 {
  toplam=0;
  for(j=i+1;j<n;j++)
  {  toplam+=a[i][j]*x[j];}
  x[i]=(b[i]-toplam)/a[i][i];
 }
 return x;
}

public static double[] EKK(double xi[],double yi[],int n)
{
int l=xi.length;
int i,j,k;
int np1=n+1;
double A[][];
A=new double[np1][np1];
double B[];
B=new double[np1];
double X[];
X=new double[np1];
for(i=0;i<n+1;i++)
 {  for(j=0;j<n+1;j++)
    {if(i==0 && j==0) A[i][j]=l;
    else  for(k=0;k<l;k++) A[i][j] += Math.pow(xi[k],(i+j));
    }
    for(k=0;k<l;k++) { if(i==0)  B[i]+= yi[k];
                 else      B[i] += Math.pow(xi[k],i)*yi[k];}
 }
System.out.println(Matrix.toString(A));
System.out.println(Matrix.toStringT(B));
X=pivotlugauss(A,B);
//X=B/A;
double max=0;
for(i=0;i<n+1;i++)
 if(Math.abs(X[i]) > max) max = Math.abs(X[i]);
for(i=0;i<n+1;i++)
 if((Math.abs(X[i]/max) > 0) && (Math.abs(X[i]/max) < 1.0e-100)) X[i]=0;
return X;
}

public static double funcEKK(double e[],double x)
{
// this function calculates the value of
// least square curve fitting function
int n=e.length;
double ff;
if(n!=0.0)
 { ff=e[n-1];
 for(int i=n-2;i>=0;i--)
  { ff=ff*x+e[i]; }
 }
else
```

```
  ff=0;
return ff;
}

public static double hata(double x[],double y[],double e[])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
 double total=0;
 for(k=0;k<n;k++)
 {
 total+=(y[k]-funcEKK(e,x[k]))*(y[k]-funcEKK(e,x[k]));
 }
total=Math.sqrt(total);
return total;
}

public static double[][]  funcEKK(double xi[],double yi[],int polinomkatsayisi,int aradegersayisi)
 {
               //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
               // ara değer sayısı 2 dir
                int n=xi.length;
               int nn=(n-1)*(aradegersayisi+1)+1;
               double z[][]=new double[2][nn];
   double E[]=EKK(xi,yi,polinomkatsayisi);
   System.out.println("katsayılar :"+Matrix.toStringT(E));
   double dx=0;
   int k=0;
   int i;
   for(i=0;i<(n-1);i++)
      {z[0][k]=xi[i];z[1][k]=funcEKK(E,z[0][k]);k++;
      for(int j=0;j<aradegersayisi;j++)
      {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
      z[0][k]=z[0][k-1]+dx;z[1][k]=funcEKK(E,z[0][k]);k++;}
      }
      z[0][k]=xi[i];z[1][k]=funcEKK(E,z[0][k]);
   return z;
 }


  public static void main(String args[]) throws IOException
  {
  double x[];
  double y[];
  String s1="a.txt";
  //JFileChooser fc=new JFileChooser();
  //if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName(); }
  double a[][]=Text.readDoubleT(s1);
  x=a[0];
  y=a[1];
  int n=x.length;
  double t[]=new double [n];
  for(int i=0;i<n;i++) t[i]=i;
  double z1[][]=funcEKK(x,y,3,10);
  System.out.println("E.K.K. z1\n"+Matrix.toString(z1));
   System.out.println("E.K.K. z2\n"+Matrix.toString(z1));
  Plot pp=new Plot(z1[0],z1[1]);
  pp.addData(x,y);
  z1=funcEKK(x,y,4,10);
  pp.addData(z1[0],z1[1]);
  z1=funcEKK(x,y,5,10);
  pp.addData(z1[0],z1[1]);
  pp.setPlabel("En küçük kareler yöntemi eğri uydurma");
  pp.setXlabel("x");
  pp.setYlabel("y");
```

```
  pp.setPlotType(0,0);
  pp.setPlotType(1,18);
  pp.plot();
  }
}
```

Çıktı 6.12-1 3,4 ve 5inci derece polinom eğri uydurma



(bolum6_015.jpg,Resimaltı: en küçük kareler eğri uydurması)

Burada tüm eğriler birbirinden farklı sonuçlar vermiştir, peki hangisi daha doğrudur? Bunu kestirmemiz her zaman için çok kolay olmıyacaktır. Burada uygulayabileceğimiz değişik bir yaklaşım da sürekli değişebilir bir ağırlık fonksiyonunu uygulamaktır. Örneğin ağırlık fonksiyonu olarak x'in fonksiyonu olarak değişen

$w_i(x) = \frac{1}{|x - x_i|^k}$ gibi bir fonksiyon alabiliriz. Böyle bir fonksiyonda $x \to x_i$ olduğunda $w_i(x) \to \infty$ olduğu görülmektedir. Eğer her x için $x=x_i$ olduğunda $f=f_i$ olduğunu kabul eder, bunun dışındaki değerler için verilen x e bağlı ağırlık fonksiyonu cinsinden en küçük kareler yöntemini çözerek f değerini bulabiliriz. Yalnız bu durumda $w_i(x)$ x in fonksiyonu olarak değiştiğinden her nokta için katsayı matrisini tekrar hesaplamamız gerekecektir. Bu yüzden hareket halindeki bir en küçük kareler metodu oluşturmuş olacağız.
Bu şartlarda en küçük kareler metodumuzu oluşturduğumuzda üstte verdiğimiz sabit en küçük kareler algoritması aşağıdaki forma dönüşecektir:

Program 6.13.2 Hareket halinde en küçük kareler polinom metodu ile eğri uydurma

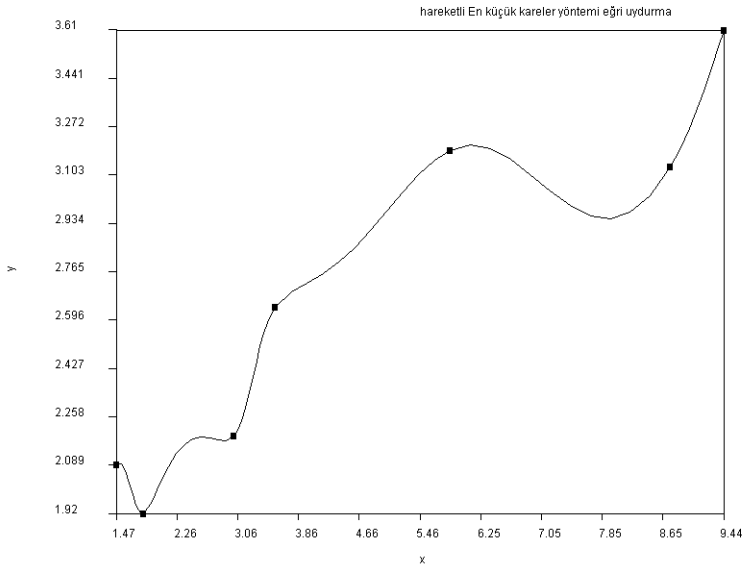
```
import java.io.*;
import javax.swing.*;

class SCO11B5
{
//en kucuk kareler metodu
  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
```

```
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
                p=k;
                buyuk=Math.abs(a[k][k]);
                for(ii=k+1;ii<n;ii++)
                {  dummy=Math.abs(a[ii][k]);
                   if(dummy > buyuk) {buyuk=dummy;p=ii;}
                }
                if(p!=k)
                { for(jj=k;jj<n;jj++)
                  { dummy=a[p][jj];
                    a[p][jj]=a[k][jj];
                    a[k][jj]=dummy;
                  }
                  dummy=b[p];
                  b[p]=b[k];
                  b[k]=dummy;
                }
                //gauss elemeyi çözme
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
           b[i]   =b[i]   -carpan*b[k];
                }
  }
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double hareketliEKK(double x,double xi[],double yi[],int n,int kk)
{
int l=xi.length;
int i,j,k;
int np1=n+1;
double A[][;
A=new double[np1][np1];
double B[;
B=new double[np1];
double X[;
X=new double[np1];
double w[]=new double[l;
for(i=0;i<l;i++) {if(x==xi[i]) return yi[i];}
for(i=0;i<l;i++) {w[i]=1.0/Math.pow(Math.abs(x-xi[i]),kk);}
for(i=0;i<np1;i++)
  {  for(j=0;j<np1;j++)
     {if(i==0 && j==0) A[i][j]=l;
     else  for(k=0;k<l;k++) A[i][j] += w[k]*Math.pow(xi[k],(i+j));
     }
     for(k=0;k<l;k++) { if(i==0)  B[i]+= w[k]*yi[k];
                        else      B[i] += w[k]*Math.pow(xi[k],i)*yi[k];}
  }
X=pivotlugauss(A,B);
double max=0;
for(i=0;i<n+1;i++)
 if(Math.abs(X[i]) > max) max = Math.abs(X[i]);
```

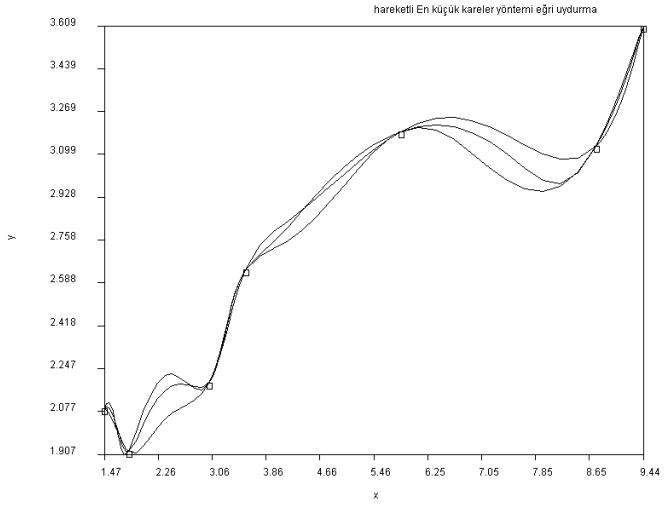
```
for(i=0;i<n+1;i++)
 if((Math.abs(X[i]/max) > 0) && (Math.abs(X[i]/max) < 1.0e-100)) X[i]=0;
return funcEKK(X,x);
}

public static double funcEKK(double e[],double x)
{
// this function calculates the value of
// least square curve fitting function
int n=e.length;
double ff;
if(n!=0.0)
  { ff=e[n-1];
  for(int i=n-2;i>=0;i--)
   { ff=ff*x+e[i]; }
  }
else
  ff=0;
return ff;
}

public static double[][]  funcEKK(double xi[],double yi[],int polinomkatsayisi,int aradegersayisi,int kk)
 {
               //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
               // ara değer sayısı 2 dir
                int n=xi.length;
               int nn=(n-1)*(aradegersayisi+1)+1;
               double z[][]=new double[2][nn];
    double dx=0;
    int k=0;
    int i;
    for(i=0;i<(n-1);i++)
      {z[0][k]=xi[i];z[1][k]=hareketliEKK(z[0][k],xi,yi,polinomkatsayisi,kk);k++;
      for(int j=0;j<aradegersayisi;j++)
      {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
      z[0][k]=z[0][k-1]+dx;z[1][k]=hareketliEKK(z[0][k],xi,yi,polinomkatsayisi,kk);k++;}
      }
      z[0][k]=xi[i];z[1][k]=hareketliEKK(z[0][k],xi,yi,polinomkatsayisi,kk);
    return z;
 }


  public static void main(String args[]) throws IOException
  {
  double x[];
  double y[];
  String s1="a.txt";
  //JFileChooser fc=new JFileChooser();
  //if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName(); }
  double a[][]=Text.readDoubleT(s1);
  x=a[0];
  y=a[1];
  int n=x.length;
  double t[]=new double [n];
  for(int i=0;i<n;i++) t[i]=i;
  double z1[][]=funcEKK(x,y,4,10,2);
  //System.out.println("E.K.K. z1\n"+Matrix.toString(z1));
  Plot pp=new Plot(z1[0],z1[1]);
  pp.setPlabel("hareketli En küçük kareler yöntemi eğri uydurma");
  pp.setXlabel("x");
  pp.setYlabel("y");
  pp.setPlotType(0,0);
  pp.addData(x,y);
   pp.setPlotType(1,21);
  pp.plot();
  }
```

```
}
```

Aynı veriye hareketli en küçük kareler metodunu uydurduğumuzda 4. dereceden polinom kullanarak ve ağırlık fonksiyonu olarak $w_i(x) = \frac{1}{|x - x_i|^2}$ fonksiyonunu alarak



(bolum6_016.jpg,Resimaltı: Hareketli en küçük kareler eğri uydurması)
Eğer yukardaki örnektekigibi 3,4 ve 5inci dereceden polinomları kullanırsak sonuç:



(bolum6_017.jpg,Resimaltı: **Hareket halinde 3,4 ve 5. dereceden en küçük kareler polinom metodu ile eğri uydurma**)

Bu metodun en büyük dezavantajı her nokta için matris sistemimizi çözmek zorunda olmamızdır. Metoddaki temel tanımdan dolayı veri tüm noktalardan tam olarak geçmektedir, bu da en küçük kareler yöntemindeki deneysel veri hatalarını düzeltme avantajını yok edebilir. Bu metod temel olarak bir interpolasyon işlemi tanımlamaktadır.

Hareketli en küçük kareler metodunu benzer şekilde birden fazla bağımsız değişkenli fonksiyonlara da uygulayabiliriz. Çok değişkenli bir fonksiyonu en küçük kareler metoduna uydurmak temel olarak tek değişkenli fonksiyonu uydurma ile aynıdır, sadece kullanılan fonksiyonun tanımını çok boyutlu olarak yapmamız gerekir.

$f(x_0, x_1,..., x_{n-1}, x_n) = \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_0, x_1,..., x_{n-1}, x_n)$ j inci derece polinomu verilmiş olsun. Bu polinoma x_0, x_1, x_2,…, x_n ,f_i, i=0...n verisini uydurmak istiyoruz. Bu veri seti için en uygun $a_j^{(m)}$ değerlerini bulmak istiyoruz. Bunun için $H(a_0^{(m)},....,a_m^{(m)}) = \sum_{i=1}^{n} w(x_0, x_1,..., x_{n-1}, x_n)\left[f_i - \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_0, x_1,..., x_{n-1}, x_n) \right]^2$ fonksiyonunun minimum değerini bulmamız gerekir. Fonksiyondaki w $(x_0, x_1,..., x_{n-1}, x_n)$ ağırlık fonksiyonu adını alır ve $w(x_0, x_1,..., x_{n-1}, x_n) \geq 0 \quad i = 1,....,n$ olmalıdır. Fonksiyonun minimum noktası türevinin 0 a eşit olduğu nokta olacaktır. Bu durumda

$$\frac{\partial H(a_0^{(m)},....,a_m^{(m)})}{\partial a_k^{(m)}} = 2\sum_{i=1}^{n} w(x_0, x_1,..., x_{n-1}, x_n)\left[f_i - \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_0, x_1,..., x_{n-1}, x_n) \right]\phi_k(x_0, x_1,..., x_{n-1}, x_n) = 0 \quad k = 0,.....,m$$

$$\left\{ \sum_{j=0}^{m} w(x_0, x_1,..., x_{n-1}, x_n)\phi_j(x_0, x_1,..., x_{n-1}, x_n)\phi_k(x_0, x_1,..., x_{n-1}, x_n) \right\}\left[a_j^{(m)}\right] = \left[\sum_{i=1}^{n} w(x_0, x_1,..., x_{n-1}, x_n)\phi_k(x_0, x_1,..., x_{n-1}, x_n) f_i \right] \quad k = 0,..,m$$

Formunu alır. Bu denklemi w=1 için bölüm 6.4 de çözmüştük. Burada ise x in fonksiyonu olarak değişecektir. Hareketli en küçük kareler metodunun ağırlık fonksiyonunun çok boyutlu fonksiyonlar için hesaplanmasında Öklid çapı kavramını kullanacağız. Eklid uzayında iki vektörel nokta

$x = (x_0, x_1,..., x_{n-1}, x_n)$

$x^i = (x_0^i, x_1^i,..., x_{n-1}^i, x_n^i)$ ve

olarak verilmişse Öklid çapı

$d(x, x^i) = \sqrt{\sum_{j=0}^{n} \left(x_j - x_j^i\right)^2}$ olarak tanımlanır. Bu durumda ağırlık fonksiyonunu Öklid çapı cinsinden yazabiliriz.

$w_i(x) = w(d(x, x^i))$. Bu tanıma bir örnek olarak

$w(d) = e^{-d^2/16}/(d^2 + 0.001)$ fonksiyonunu verebiliriz. Buradaki x vektörünü istenen nokta, x^i vektörünü ise bu noktaya en yakın verilen veri noktası olarak alacağız. Bu şartlarda en küçük kareler metodumuzu oluşturduğumuzda bölüm 6.4 de verdiğimiz çok boyutlu sabit en küçük kareler algoritması aşağıdaki forma dönüşecektir:

Program 6.13.3 Hareket halinde en küçük kareler polinom metodu ile n boyutlu yüzey uydurma (birden fazla bağımsız değişken olması durumunda)

```
import java.io.*;
import java.util.*;
import javax.swing.*;
import java.awt.event.*;

//  yüzey eğri uydurma
//  f(x,y)=a0+a1*x+a2*y+a3*x^2+a4*y^2+a5*xy+a6*x^3+a7*y^3+a8*x^2y
//  +a9*xy^2+a10*x^4+a11*y^4+a12*x^2y^2+a13x^3y+a14*xy^3

abstract class f_xir
{
  abstract double func(double x[],int equation_ref);
}

class fa extends f_xir
{
            double func(double x[],int i)
            {//genel bir yüzey modeli
   double xx=0.0;
   //density function
   if(i==0)      {xx=1.0;         }
   else if (i==1) {xx=x[0];        }
   else if (i==2) {xx=x[1];        }
```

```
    else if (i==3) {xx=x[0]*x[0];     }
    else if (i==4) {xx=x[1]*x[1];     }
    else if (i==5) {xx=x[0]*x[1];     }
    else if (i==6) {xx=x[0]*x[0]*x[0];}
    else if (i==7) {xx=x[1]*x[1]*x[1];}
    else if (i==8) {xx=x[0]*x[0]*x[1];}
    else if (i==9) {xx=x[0]*x[1]*x[1];}
    else if (i==10){xx=x[0]*x[0]*x[0]*x[0];}
    else if (i==11){xx=x[1]*x[1]*x[1]*x[1];}
    else if (i==12){xx=x[0]*x[0]*x[1]*x[1];}
    else if (i==13){xx=x[0]*x[0]*x[0]*x[1];}
    else if (i==14){xx=x[0]*x[1]*x[1]*x[1];}
    return xx;
                }
}

class SCO11HG5
{
                public static double d(double x[],double xi[])
                {//hareketli en küçük kareler metodu Öklid çap fonksiyonu
                double total=0;
                for(int i=0;i<x.length;i++)
                {total+=(x[i]-xi[i])*(x[i]-xi[i]);}
                return Math.sqrt(total);
    }

    public static double w(double x[],double xi[])
    {
                // Hareketli en küçük kareler metodu ağırlık fonksiyonu
                double d=d(x,xi);
                double eps=0.001;
                double alpha=1.0/16.0;
                double ww=Math.exp(-alpha*d*d)/(d*d+eps);
                return ww;
                }

//  Genel en küçük kareler metodu en kucuk kareler metodu
                public static double[][] Transpose(double [][] left)
    { //transpose matrix (if A=a(i,j) Transpose(A)=a(j,i)
    int i,j;
    int n=left.length;
    int m=left[0].length;
    double b[][];
    b=new double[m][n];
    for(i=0;i<n;i++)
      {for(j=0;j<m;j++)  {b[j][i]=left[i][j];} }
    return b;
    }

  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
                p=k;
                buyuk=Math.abs(a[k][k]);
                for(ii=k+1;ii<n;ii++)
                {  dummy=Math.abs(a[ii][k]);
                  if(dummy > buyuk) {buyuk=dummy;p=ii;}
```

```
                }
                if(p!=k)
                { for(jj=k;jj<n;jj++)
                  { dummy=a[p][jj];
                    a[p][jj]=a[k][jj];
                    a[k][jj]=dummy;
                  }
                  dummy=b[p];
                  b[p]=b[k];
                  b[k]=dummy;
                }
                //gauss elemeyi çözme
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
           b[i]   =b[i]   -carpan*b[k];
                }
  }
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double HEKK(double x[],double xi[][],double yi[],int n)
{ // n boyutlu yüzey eğri uydurma
  // yüzey EKK katsayılarını hesaplar
fa f=new fa();
int l=xi.length; //değişken sayısı
int i,j,k;
int np1=n+1;
double A[][];
A=new double[np1][np1];
double B[];
B=new double[np1];
double E[];
E=new double[np1];
for(i=0;i<n+1;i++)
  {  B[i]=0;
                for(j=0;j<np1;j++)
    {
                A[i][j]=0.0;
    for(k=0;k<l;k++) A[i][j]+= w(x,xi[k])*f.func(xi[k],i)*f.func(xi[k],j);
    }
    for(k=0;k<l;k++) B[i]+= w(x,xi[k])*f.func(xi[k],i)*yi[k];
  }
E=pivotlugauss(A,B);
//X=B/A;
double max=0;
for(i=0;i<n+1;i++)
 if(Math.abs(E[i]) > max) max = Math.abs(E[i]);
for(i=0;i<n+1;i++)
 if((Math.abs(E[i]/max) > 0) && (Math.abs(E[i]/max) < 1.0e-100)) E[i]=0;
double z=funcEKKgenel(E,x);
return z;
}
```

```
public static double funcEKKgenel(double e[],double x[])
{
// tek bir nokta için çok boyutlu eğri değeri hesaplar
fa f=new fa();
int n=e.length;
double ff=0;
if(n!=0.0)
  {
  for(int i=n-1;i>=0;i--)
   {ff+=e[i]*f.func(x,i);}
  }
return ff;
}

public static double[] HEKK(double x[][],double xi[][],double yi[],int n)
{
int m=x.length;
double z[]=new double[m];
for(int i=0;i<m;i++)
{
z[i]= HEKK(x[i],xi,yi,n);
}
return z;
}

public static double[][] cikti(double x[][],double xi[][],double yi[],int n)
{
int m=x.length;
int nn=x[0].length;
double c[][]=new double[m][nn+1];
double z[]=new double[m];
for(int i=0;i<m;i++)
{for(int j=0;j<nn;j++)
                {c[i][j]=x[i][j];}
}
for(int i=0;i<m;i++)
{
c[i][nn]= HEKK(x[i],xi,yi,n);
}
return c;
}

  public static void main(String[] args)
  {
  String s1=JOptionPane.showInputDialog("eğri uydurma girdi verisi dosya adı : ");
  double c[][]=Text.readDouble(s1);
  int n=c[0].length;
  int m=n-1;
  int l=c.length;
  double x[][]=new double[l][m];
  double y[]=new double[l];
  for(int i=0;i<l;i++)
  {for(int j=0;j<m;j++)
   {x[i][j]=c[i][j];}
   y[i]=c[i][m];
  }
  String s2=JOptionPane.showInputDialog("çıktı bağımsız değişken listesi dosya adı : ");
  double xy[][]=Text.readDouble(s2);
  double z[][]=cikti(xy,x,y,4);
  Text.print(z);
  }}
```

Programa örnek veri olarak girdi verisi x_{ij},y_i örnek veri $y=x_0^2+x_1^2$ fonksiyonundan oluşturulmuştur.

0 0 0	3 0 9

0	1	1	3	1	10
0	2	4	3	2	13
0	3	9	3	3	18
0	4	16	3	4	25
1	0	1	4	0	16
1	1	2	4	1	17
1	2	5	4	2	20
1	3	10	4	3	25
1	4	17	4	4	32
2	0	4			
2	1	5			
2	2	8			
2	3	13			
2	4	20			

Programa örnek veri olarak eğri uydurulacak bağımsız değişken verisi x_{ij}

0	1	1.5	0
0	1.5	1.5	0.5
0	2	1.5	1
0.5	0	1.5	1.5
0.5	0.5	1.5	2
0.5	1	2	0
0.5	1.5	2	0.5
0.5	2	2	1
1	0	2	1.5
1	0.5	2	2
1	1		
1	1.5		
1	2		

çıktı değerleri:

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO11HG5
      0.000000000000000      1.000000000000000      1.000000000000000
      0.000000000000000      1.500000000000000      2.249999999999999
      0.000000000000000      2.000000000000000      4.000000000000002
      0.500000000000000      0.000000000000000      0.249999999999999
      0.500000000000000      0.500000000000000      0.500000000000000
      0.500000000000000      1.000000000000000      1.250000000000000
      0.500000000000000      1.500000000000000      2.499999999999996
      0.500000000000000      2.000000000000000      4.250000000000001
      1.000000000000000      0.000000000000000      1.000000000000000
      1.000000000000000      0.500000000000000      1.250000000000000
      1.000000000000000      1.000000000000000      2.000000000000000
      1.000000000000000      1.500000000000000      3.249999999999996
      1.000000000000000      2.000000000000000      4.999999999999998
      1.500000000000000      0.000000000000000      2.249999999999997
      1.500000000000000      0.500000000000000      2.500000000000002
      1.500000000000000      1.000000000000000      3.249999999999998
      1.500000000000000      1.500000000000000      4.499999999999994
      1.500000000000000      2.000000000000000      6.250000000000002
      2.000000000000000      0.000000000000000      4.000000000000001
```

```
2.000000000000000   0.500000000000000   4.249999999999996
2.000000000000000   1.000000000000000   4.999999999999993
2.000000000000000   1.500000000000000   6.249999999999994
2.000000000000000   2.000000000000000   7.999999999999991

> Terminated with exit code 0.
```

Olarak bulunur.

6.14 Benstein polinomları ve bezier eğrileri

Bir fonksiyon yaklaşım formu olan Bernstein polinomları a-b aralığında

$$B_n(f;x)=\sum_{i=0}^{n}\binom{n}{i}(x-a)^i(b-x)^{n-i}\,f(x_i)$$

şeklinde tanımlanır. Buradaki x_i sabit aralıklar için $x_i=a+ih=a+(i/n)(b-a)$ i=0..n formunda yazılabilir. Aynı fonksiyon eğer 0 ile 1 aralığında tanımlanırsa

$$B_n(f;x)=\sum_{i=0}^{n}\binom{n}{i}x^i(1-x)^{n-i}\,f(i/n)=\sum_{i=0}^{n}f(i/n)B_{n,i}(x)$$

$B_{n,i}(x)=\binom{n}{i}x^i(1-x)^{n-i}$ şeklini alır.

Bernsttein polinomlarını direk olarak fonksiyon yaklaşımı olarak kullandığımızda dönüşüm oranlarının çok düşük olduğunu, tam bir yaklaşım için pratik olmıyan çok fazla iterasyon gerektiğini gözlemleriz. Bu polinomlar verilen ilk ve son noktalardan tam olarak geçerlerken orta noktalara daha yavaş yakınsarlar, ancak orta noktaları sadece çizmek istediğimiz eğri için bir referans noktası olarak düşünür isek oldukça pratik bir çizim eğri oluşturma yöntemi elde ederiz. Bu yönteme Bezier eğrileri ismi vereceğiz. Bezier eğrilerinde daha önce kullandığımız indirek referanslama yöntemi kullanılır ve böylece genel bir eğri elde etme metodu oluşturulmuş olur. Herhangi bir (x_i,y_i) veri seti için 0 ile 1 arasında değişen bir t değişkeni kabul edersek (x_0,y_0) ve (x_1,y_1) arasında

$$x(t)=\sum_{i=0}^{n}x_iB_{n,i}(t)$$

$$y(t)=\sum_{i=0}^{n}y_iB_{n,i}(t)$$

denklem setini (Bezier eğrisini) tanımlayabiliriz. Aynı kavramı kullanarak daha üst boyutlu eğrileri de oluşturmak oldukça basit bir denklem seti üzerinden mümkün olur.
Örnek problem olarak

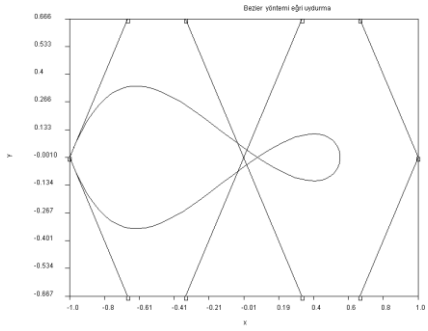
```
-1.0000000000   0.0000000000
-0.6666666667   0.6666666667
-0.3333333333   0.6666666667
0.3333333333    -0.6666666667
0.6666666667    -0.6666666667
1.0000000000    0.0000000000
0.6666666667    0.6666666667
0.3333333333    0.6666666667
-0.3333333333   -0.6666666667
-0.6666666667   -0.6666666667
-1.0000000000   0.0000000000
```

verisini kullanarak bir Bezier eğrisi oluşturalım. Bezier eğrisinde eğrinin ilk ve son noktadan geçtiğini diğer noktaların referans noktaları olduğunu tekrar hatırlatalım.

Program 6.14.1 başlangıç, bitiş ve referans noktaları kullanarak Bezier eğrisi uydurma

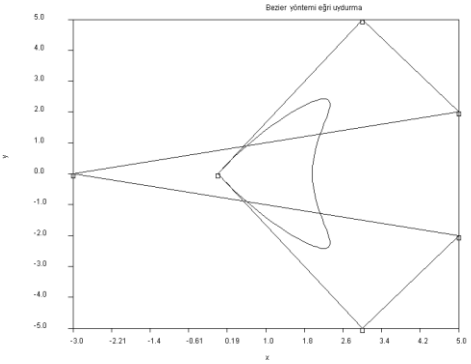
```
import java.io.*;
import javax.swing.*;

class SCO11B6
{
//Bernstein polinomları ve Bezier eğrisi (bir boyutlu)
  public static double[] B(int n,double t)
  { //  0<=t<=1
                double b[]=new double[n+1];
    b[0]=1.0;
    for(int i=1;i<=n;i++)
    {b[i]=t*b[i-1];
     for(int j=i-1;j>=1;j--)
     {b[j]=(1.0-t)*b[j]+t*b[j-1];}
     b[0]=(1.0-t)*b[0];
    }
  return b;
  }

  public static double funcB(double a[],double t)
  { double xx=0;
    int n=a.length-1;
    double b[]=B(n,t);
    for(int i=0;i<=n;i++)
    {xx+=a[i]*b[i];}
  return xx;
  }

  public static double[][] Bezier(double xi[],double yi[],int noktasayisi)
  {
                int n=xi.length;
                double z[][]=new double[2][noktasayisi];
    double dt=1.0/(double)noktasayisi;
    double t=0;
    int k=0;
    int i;
    for(i=0;i<noktasayisi;i++)
       {z[0][i]=funcB(xi,t);z[1][i]=funcB(yi,t);
        t=t+dt;
       }
    return z;
  }



  public static void main(String args[]) throws IOException
  {
  double x[];
  double y[];
  String s1="a.txt";
  //JFileChooser fc=new JFileChooser();
  //if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName(); }
  double a[][]=Text.readDoubleT(s1);
  x=a[0];
  y=a[1];
  int n=x.length;
  //for(double t=0;t<=1.0;t+=0.1) {System.out.println(funcB(4,x,t)+" "+funcB(4,y,t));}
  double z1[][]=Bezier(x,y,100);
  //System.out.println("E.K.K. z1\n"+Matrix.toString(Matrix.T(z1)));
  Plot pp=new Plot(z1[0],z1[1]);
  pp.setPlabel("Bezier  yöntemi eğri uydurma");
  pp.setXlabel("x");
  pp.setYlabel("y");
```

```
  pp.setPlotType(0,0);
  pp.addData(x,y);
  pp.addData(x,y);
  pp.setPlotType(2,18);
  pp.plot();
  }
}
```



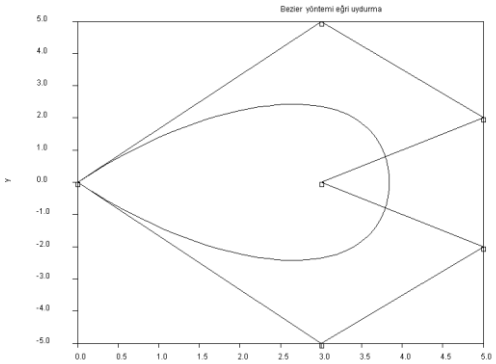
(bolum6_018.jpg,Resimaltı: **Bernstein polinomlarıyla sonsuz işareti çizdirme)**

Referans noktalarını değiştirerek kompleks eğrileri çizmek mümkündür. Örneğin
0 0
3 5
5 2
-3 0
5 -2
3 -5
0 0 verisi



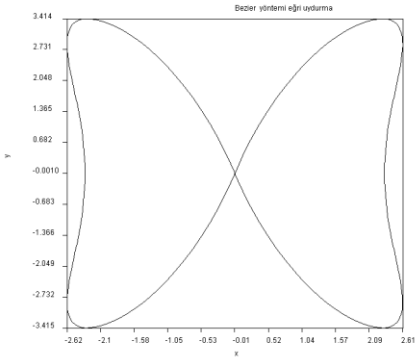
(bolum6_019.jpg,Resimaltı: **Bernstein polinomlarıyla kelebek kanadı çizdirme)**

eğrisini oluştururken
0 0
3 5
5 2
3 0
5 -2
3 -5
0 0



(bolum6_020.jpg,Resimaltı: **Bernstein polinomlarıyla damla çizdirme)**

eğrisini oluşturacaktır. Bu iki eğri arasında sadece bir referans noktasını değiştirdik.
Bu şekilde az sayıdaki referans noktası ile daha kompleks şekillerde elde edebiliriz.



(bolum6_021.jpg,Resimaltı: **Bernstein polinomlarıyla kelebek çizdirme)**

bu şekili çizmek için referans değerleri olarak

```
  double x[]={0,1.5,4,5,-3,5,4,1.5,0.0};
  double y[]={0,5,5,2,0,-2,-5,-5,0};
  double x1[]={0,-1.5,-4,-5,3,-5,-4,-1.5,0.0};
  double y1[]={0,5,5,2,0,-2,-5,-5,0};
```

x,y ve x1,y1 setleri kullanılmıştır.

6.15 Kısmi devamlı fonksiyonlar kullanarak en küçük kareler metodu ile eğri uydurma

Gerçek fonksiyonlara eğri uydurmada karşımıza çıkan temel problemlerden biri, en küçük kareler yöntemi ile eğri uydurduğumuzda eğrinin derecesi küçük olduğunda iyi bir uyum sağlıyamamak, eğrinin derecesini arttırdığımızda ise hesap hatası arttığından yine sonuç alamamak gibi bir ikilem oluşmasıdır. Eğer uydurduğumuz eğrinin hata seviyesi istediğimiz sınırlara düşemiyorsa hata seviyesini düşürmenin yollarından birisi devamlı tek bir fonksiyon yerine kısmı devamlı bir fonksiyon ailesi uydurmaktır. Her fonksiyon yeterince küçük parçalara ayrıldığında lineere yaklaşacağından hata miktarı da tek bir fonksiyon uydurmaya göre otomatik olarak azalır. Yöntem matematiksel olarak tek bir eğri uydurma ile aynıdır. Temel far burada bir eğri ailesinin olmasıdır. Bu tür bir eğri uydurmada doğal olarak her eğrinin geçerli olduğu minimum ve maksimum sınırların da tanımlanmış olması gereklidir.
Örnek olarak ideal gaz özgül ısı verisine

$Cp=a_0 + a_1x10^{-3}T + a_2x10^5/T^2 + a_3x10^{-5}T^2$ denklemine kısmi devamlı olarak uyduralım. Veri olarak Azot gazının Cp değerlerini (kJ/kg K) alalım

298.15	29.123	1800	35.598	3500	37.324
300	29.125	1900	35.794	3600	37.373
400	29.246	2000	35.969	3700	37.42
500	29.583	2100	36.126	3800	37.464
600	30.113	2200	36.267	3900	37.507
700	30.752	2300	36.395	4000	37.548
800	31.43	2400	36.511	4100	37.588
900	32.094	2500	36.616	4200	37.627
1000	32.696	2600	36.713	4300	37.665
1100	33.24	2700	36.802	4400	37.701
1200	33.723	2800	36.884	4500	37.737
		2900	36.961	4600	37.773
1300	34.148	3000	37.031	4700	37.808
1400	34.519	3100	37.097	4800	37.842
1500	34.844	3200	37.159	4900	37.877
1600	35.128	3300	37.217	5000	37.911
1700	35.378	3400	37.272		

Bilgisayar programı olarak daha önce gördüğümüz genelleştirilmiş en küçük kareler yöntemini kullanacağız, ancak bu versyonumuzda program veriyi istediğimiz alt veri parçalarına bölerek ayrı ayrı eğri uyduracaktır. Bu veriye 12 kısmi devamlı eğri uyduracak olursak :

Program 6.15.1 kısmi devamlı eğri uydurma programı Gas_GEN_Data.java

```
import java.io.*;
import java.util.*;
import javax.swing.*;

//prototip fonksiyon sınıfı
abstract class f_xr
{
  abstract double func(double x,int equation_ref);
}
/*
class fh extends f_xr
{
            double func(double t,int i)
            {// Cp(T) = xa[i]+xb[i]*1e-3*T+xc[i]*1.0e5/T^2+xd[i]*1e-6*T^3
     double T=t;
              double xx=0.0;
     if(i==0) xx=T;
     else if(i==1) xx=1.0e-3/2.0*T*T;
     else if(i==2) xx=-1.0e5/T;
     else if(i==3) xx=1.0e-6/3.0*T*T*T;
     return xx;
            }
}
*/
class fcp extends f_xr
{
            double func(double t,int i)
            { // Cp(T) = xa[i]+xb[i]*1e-3*T+xc[i]*1.0e5/T^2+
              // xd[i]*1e-6*T^2
     double T=t;
              double xx=0.0;
     if(i==0) xx=1.0;
     else if(i==1) xx=1.0e-3*T;
     else if(i==2) xx=1.0e5/(T*T);
     else if(i==3) xx=1.0e-6*T*T;
     return xx;
            }
```

```
}

class Gas_GEN_Data
{
//this class reads and store and proses ideal gas data
public double M;        // kg/kmol
public double h0;       // KJ/kmol at temperature a[0][0]=Tref
public double s0;       // KJ/kmolK at temperature a[0][0]=Tref
public double hf;       // KJ/kmol at 298 K
public double R=8.3145; // KJ/kgmolK
public double a[][];
public String GasName;
public String openName;
public String ekbilgi;
int fitoption;
f_xr f1;
f_xr f2;

public double cpl[][];  // ideal gas heat capacitiy
public double visl[][]; // saturated liquid viscosity
public double kl[][]; // saturated liquid thermal conductivity

//Genel en küçük kareler metodu en kucuk kareler metodu
   public Gas_GEN_Data(int fitoptioni,String GasNamei,String openNamei,double ai[][],int ngurup,double Mi, double h0i,double
s0i,double hfi)
   {
   //fitoption : 0   Cp
   //diğer          h
   fitoption=fitoptioni;
   GasName=GasNamei;
   a=ai;
   ekbilgi=" ";
   openName=openNamei;
   int n=a.length;
   int i,j;
   if(fitoption==0) {f1=new fcp(); }
   else            {f1=new fh(); }
   f2=new fcp();
   cpl=EKKgenel(f1,a[0],a[2],ngurup);
   visl=EKKgenel(f2,a[0],a[3],ngurup);
   kl=EKKgenel(f2,a[0],a[4],ngurup);
   M=Mi;
   h0=h0i;
   s0=s0i;
   hf=hfi;
   }

  public static double[] pivotlugauss(double a[][],double b[])
  { //kısmi pivotlu gauss eleme yöntemi
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss eleme
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivotlama
              p=k;
              buyuk=Math.abs(a[k][k]);
              for(ii=k+1;ii<n;ii++)
              {  dummy=Math.abs(a[ii][k]);
                 if(dummy > buyuk) {buyuk=dummy;p=ii;}
              }
              if(p!=k)
```

```
                { for(jj=k;jj<n;jj++)
                  { dummy=a[p][jj];
                    a[p][jj]=a[k][jj];
                    a[k][jj]=dummy;
                  }
                  dummy=b[p];
                  b[p]=b[k];
                  b[k]=dummy;
                }
                //gauss elemeyi çözme
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
          b[i]   =b[i]   -carpan*b[k];
                }
  }
  //geriye doğru yerine koyma
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double[][] EKKgenel(f_xr f, double xi[],double yi[],int ngurup)
{
                //ngurup : bir guruptaki veri sayısı
                //negri  : eğrinin derecesi
                //gurupsayisi : grupların sayısı
                int negri=3;
                int n=negri;
                double y[]=new double[xi.length];
                for(int i=0;i<xi.length;i++)
                  {y[i]=yi[i];}
if(ngurup<negri) ngurup=negri;
int Nxi=xi.length; //toplam veri sayısı
int gurupsayisi=Nxi/ngurup;
int ngurup1=ngurup+Nxi%ngurup;
//
int i,j,k;
int np1=4;
int np5=np1+2;
double A[][];
A=new double[np1][np1];
double B[];
B=new double[np1];
double X[][];
X=new double[gurupsayisi][np5];
System.out.println("gurup sayisi = "+ gurupsayisi);
double Y[]=new double[np1];
int l=0;
double max=0;
//
for(l=0;l<gurupsayisi-1;l++)
{
for(i=0;i<np1;i++)
  {  B[i]=0;
                 for(j=0;j<np1;j++)
    {
                 A[i][j]=0.0;
```

```
    for(k=l*ngurup;k<=(l+1)*ngurup;k++) {A[i][j]+=f.func(xi[k],i)*f.func(xi[k],j);}
    }
    for(k=l*ngurup;k<=(l+1)*ngurup;k++) B[i]+= f.func(xi[k],i)*y[k];
  }
Y=pivotlugauss(A,B);//pivotlugauss(A,B);

int ii;
for(ii=0;ii<Y.length;ii++) {X[l][ii]=Y[ii];}
X[l][ii]=xi[l*ngurup];
X[l][ii+1]=xi[(l+1)*ngurup];
for(i=0;i<n+1;i++)
 if(Math.abs(X[l][i]) > max) max = Math.abs(X[l][i]);
for(i=0;i<n+1;i++)
 if((Math.abs(X[l][i]/max) > 0) && (Math.abs(X[l][i]/max) < 1.0e-100))X[l][i]=0;
}
l=gurupsayisi-1;
int k1=l*ngurup;
int k2=(l+1)*ngurup+Nxi%ngurup;
for(i=0;i<np1;i++)
  {  B[i]=0;
            for(j=0;j<np1;j++)
    {
            A[i][j]=0.0;
    for(k=k1;k<k2;k++) {A[i][j]+=f.func(xi[k],i)*f.func(xi[k],j);}
    }
    for(k=k1;k<k2;k++) {B[i]+= f.func(xi[k],i)*y[k];}
  }
Y=pivotlugauss(A,B);
int ii;
for(ii=0;ii<Y.length;ii++) {X[l][ii]=Y[ii];}
X[l][ii]=xi[k1];
X[l][ii+1]=xi[k2-1];

return X;
}

public static double funcEKKgenel(f_xr f,double e[][],double x)
{
// this function calculates the value of
// least square curve fitting function
int n1=e.length;
int n2=e[0].length;
int n3=n2-2;
System.out.println("n1="+n1+"n2="+n2+"n3="+n3);
double ff=0;
double xlow,xhigh;
//for(int j=0;j<n1;j++)
//{for(int i=0;i<n3;i++){System.out.print(e[j][i]+" ");}
//System.out.println();
//}

for(int j=0;j<n1;j++)
{
xlow=e[j][n3];
xhigh=e[j][n3+1];
System.out.println("xlow="+xlow+"xhigh="+xhigh);
if((x>=xlow) && (x<=xhigh))
{
ff=0;
for(int i=0;i<n3;i++)
  {ff+=e[j][i]*f.func(x,i);System.out.println("ff="+ff+"i="+i);}
break;
}
}
return ff;
}
```

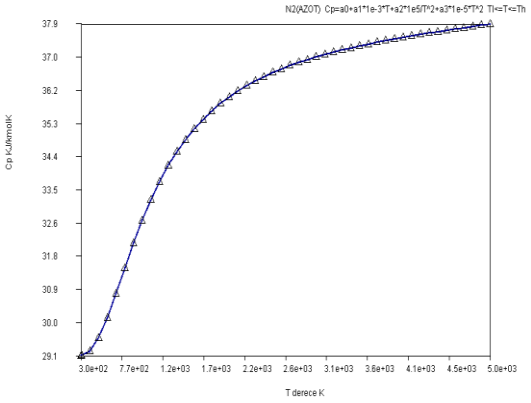
```
public static double[][] xyfit(f_xr f,double x[],double yi[],double e[][])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
 double total=0;
 double yy=0;
 double z[][]=new double[2][x.length];
 for(k=0;k<n;k++)
 {   z[0][k]=x[k];
                z[1][k]=funcEKKgenel(f,e,x[k]);
 }
return z;
}


public static double[][] dxyfit(f_xr f,double x[],double yi[],double e[][])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
 double total=0;
 double yy=0;
 double z[][]=new double[2][x.length];
 for(k=0;k<n;k++)
 {   z[0][k]=x[k];
                z[1][k]=yi[k]-funcEKKgenel(f,e,x[k]);
 }
return z;
}

public static double hata(f_xr f,double x[],double yi[],double e[][])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
 double total=0;
 double yy=0;
 for(k=0;k<n;k++)
 {
                yy=yi[k]-funcEKKgenel(f,e,x[k]);
 total+=yy*yy;
 }
total=Math.sqrt(total/(n-1));
return total;
}
 public static double[][]  hataEKKPlot(f_xr f,double xi[],double yi[],int ngurup)
 {              //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
                // ara değer sayısı 2 dir
                //ngurup bir guruptaki veri sayısı
                //polinom katsayısı : polinomun derecesi
                int n=xi.length;
                int l;
                double z[][];
    double E[][];//=EKKgenel(xi,yi,polinomkatsayisi,ngurup);
    double dx=0;
    int k=0;
    int i;
                E=EKKgenel(f,xi,yi,ngurup);
    z=dxyfit(f,xi,yi,E);
    Plot pp=new Plot(z[0],z[1]);
      pp.setPlabel("Cp=a0+a1*1e-3*T+a2*1e5/T^2+a3*1e-5*T^2  Tl<=T<=Th");
      pp.setXlabel("T derece K");
      pp.setYlabel("hata Cp KJ/kmolK");
                //System.out.println("hata="+hata+"ngurup = "+ngurup);
```

```
  //pp.addData(z[0],z[2]);
  //pp.addData(z[0],z[3]);
  pp.plot();
  return z;
}

public static double[][]  funcEKKPlot(f_xr f,double xi[],double yi[],int ngurup)
{           //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
            // ara değer sayısı 2 dir
            //ngurup bir guruptaki veri sayısı
            //polinom katsayısı : polinomun derecesi
            int n=xi.length;
            int l;
            double z[][];
  double E[][];//=EKKgenel(xi,yi,polinomkatsayisi,ngurup);
  double dx=0;
  int k=0;
  int i;
            E=EKKgenel(f,xi,yi,ngurup);
  z=xyfit(f,xi,yi,E);
  Plot pp=new Plot(z[0],z[1]);
    pp.setPlabel("Cp=a0+a1*1e-3*T+a2*1e5/T^2+a3*1e-5*T^2  Tl<=T<=Th");
    pp.setXlabel("T derece K");
    pp.setYlabel("Cp KJ/kmolK");
            //System.out.println("hata="+hata+"ngurup = "+ngurup);
   pp.addData(xi,yi,1);
  //pp.addData(z[0],z[2]);
  //pp.addData(z[0],z[3]);
  pp.plot();
  return z;
}

public static double[][]  funcEKKgenel(f_xr f,double xi[],double yi[],int ngurup,int aradegersayisi)
{
            // aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
            // ara değer sayısı 2 dir
            // ngurup bir guruptaki veri sayısı
            // polinom katsayısı : polinomun derecesi
            int n=xi.length;
            int nn=(n-1)*(aradegersayisi+1)+1;
            double z[][]=new double[2][nn];
  double E[][]=EKKgenel(f,xi,yi,ngurup);
  Text.print(E,"EfunkEKK");
  double dx=0;
  int k=0;
  int i;
  for(i=0;i<(n-1);i++)
    {z[0][k]=xi[i];z[1][k]=funcEKKgenel(f,E,z[0][k]);
    for(int j=0;j<aradegersayisi;j++)
    {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
    k++;
    z[0][k]=z[0][k-1]+dx;z[1][k]=funcEKKgenel(f,E,z[0][k]);k++;}
    }
    z[0][k]=xi[i];z[1][k]=funcEKKgenel(f,E,z[0][k]);
  return z;
}

 public double Cp_l(double t)
 { return funcEKKgenel(f2,cpl,t);}

 public double viscosity_l(double t)
 {return funcEKKgenel(f2,visl,t);}

 public double k_l(double t)
 {return funcEKKgenel(f2,kl,t);}
```

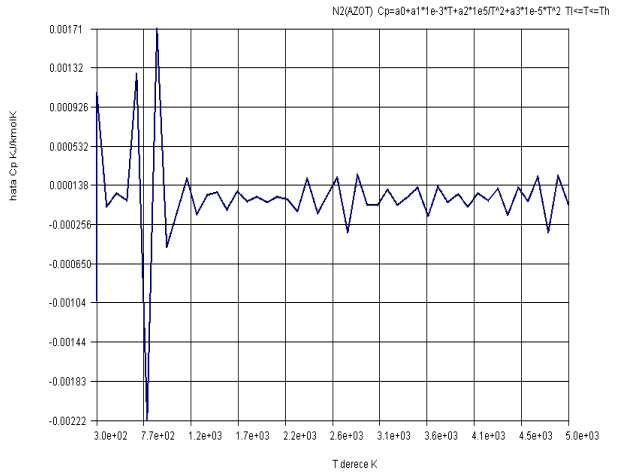
```
  public static void main(String args[]) throws IOException
  {
  //çok büyük veri setlerinde veriyi dosyadan okumamız gerekir
  String s1=JOptionPane.showInputDialog("dosya adı : ");
  double c[][]=Text.readDoubleT(s1);
  Text.print(Text.T(c));
  //int verituru=Integer.parseInt(JOptionPane.showInputDialog("veri türü 0:Cp 1:h : "));
  double d[][]=EKKgenel(new fcp(), c[0],c[1],4);
  funcEKKPlot(new fcp(), c[0],c[1],4);
  Text.print(d);
  hataEKKPlot(new fcp(), c[0],c[1],4);
  System.out.println(Matrix.toString(d));
  }

}
```

N_2 (AZOT) Cp Özgül ısı kısmi devamlı polinom katsayıları 12 denklem

A_i	B_i	C_i	D_i	T_{Li}	T_{Hi}
29.48410467	-4.39231362	0.135791768	8.962719183	298.2	600
19.58127458	18.1244325	6.04125597	-5.613346914	600	1000
25.58318122	9.820000158	-3.406728392	-2.366536137	1000	1400
31.97208934	3.568391773	-23.39157514	-0.640497282	1400	1800
33.95938165	1.99152595	-32.75504611	-0.288631316	1800	2200
37.66790853	-0.18797764	-64.90671398	0.073070085	2200	2600
32.38982337	2.448472237	-1.808705885	-0.298231241	2600	3000
37.37149772	0.104842902	-68.56025763	0.011860004	3000	3400
34.09434238	1.362910336	-2.482405204	-0.124112185	3400	3800
42.95067177	-1.76724832	-214.1109621	0.18778721	3800	4200
56.98821926	-5.88637732	-689.7268107	0.525601294	4200	4600
36.43273733	0.299350851	-15.74029559	0.001781188	4600	5000

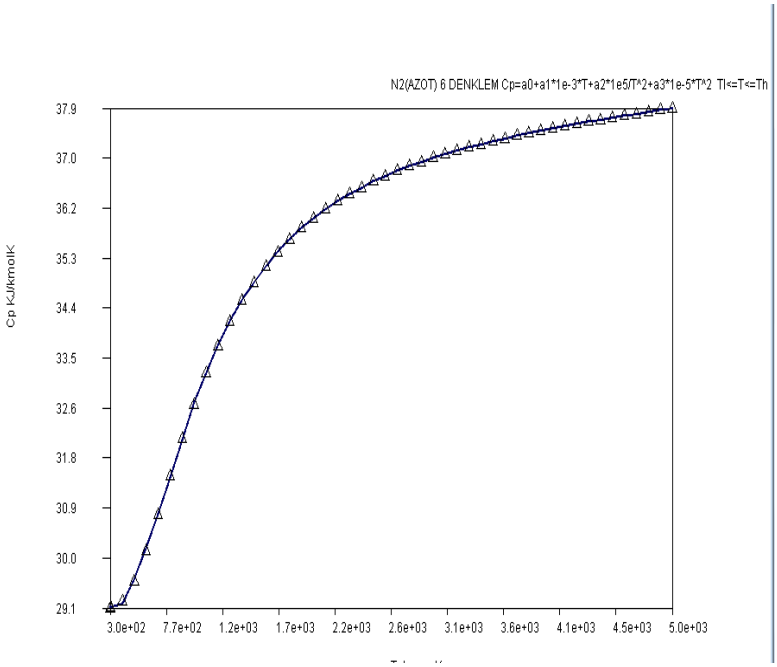


(bolum6_022.jpg,Resimaltı: **12 kısmi devamlı denklem olarak uydurulmuş eğri ve verinin görünümü)**

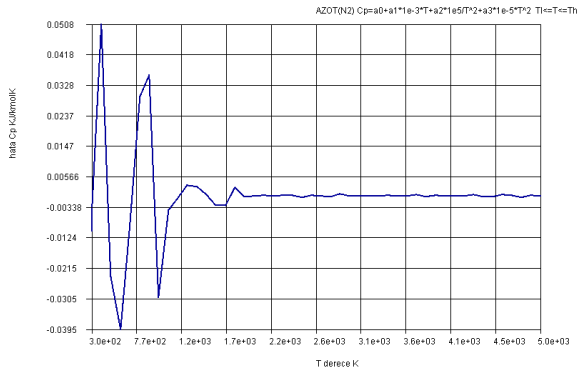


(bolum6_023.jpg,Resimaltı: 12 **kısmi devamlı denklem olarak uydurulmuş eğrinin hata miktarı)**

Aynı veri 6 kısmi devamlı eğriye uydurulacak olursa :



(bolum6_024.jpg,Resimaltı: 6 **kısmi devamlı denklem olarak uydurulmuş eğri)**



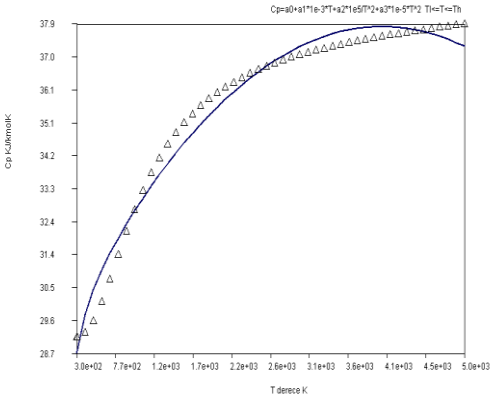
(bolum6_025.jpg,Resimaltı: **6 kısmi devamlı denklem olarak uydurulmuş verinin hata miktarının görünümü)**

Eğer sadece 1 eğri kullanırsak (sürekli fonksiyon)

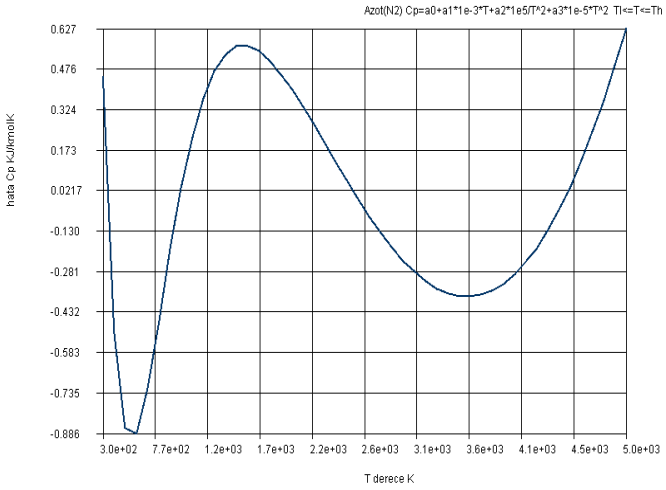
Çıktı6.14.7 N2(AZOT) Cp Özgül ısı devamlı polinom

A_i	B_i	C_i	D_i	T_{Li}	T_{Hi}
28.87568	4.485071	-1.318301	-0.5604611	298.15	5000

Çıktı6.14.8 1 denklem olarak uydurulmuş eğri ve veri grafiği



(bolum6_026.jpg,Resimaltı: **1 kısmi devamlı denklem olarak uydurulmuş eğri)**



(bolum6_027.jpg,Resimaltı: **1 kısmi devamlı denklem olarak uydurulmuş eğrinin hata grafiği)**

Grafik formdaki sonuçlar incelendiğinde kısmi devamlı fonksiyoların hata miktarı üzerindeki etkileri açık olarak görülmektedir. Hata miktar kontrollu eğri uydurma gerktiğinde kısmi devamlı fonksiyonlar önemli bir alternatif olarak karşımıza çıkmaktadır.

6.16 Bölümlü fonksiyon yaklaşımları : Pade yaklaşımı

Pay ve payda şeklinde düzenlenmiş polinom denklemleri belli veri veya fonksiyona daha az hatalı olarak yaklaşabilirler. Normal yüksek dereceli polinomların en göze batan dezavantajı salınım yapmalarıdır. Bu salınımlar verinin bir çok noktada ortalama hatanın çok üzerinde hata yapmasına yol açar. Bölümlü polinomlar ise hatayı tüm bölgede daha düzenli yayarak salınım hatalarını minimum düzeye indirir. Kesirli veya bölümlü fonksiyonlar diyebileceğimiz bu gurup fonksiyonlar en genel olarak

$r(x) = \dfrac{p(x)}{q(x)}$ formunda tanımlanır.

Buradaki p(x) ve q(x) polinom değerleridir.

$$r(x) = \frac{p(x)}{q(x)} = \frac{p_0 + p_1 x + p_2 x^2 + ... + p_n x^n}{q_0 + q_1 x + q_2 x^2 + ... + q_n x^m}$$

Pade′ bölümlü yaklaşımı temel olarak taylor Taylor denklemine dayanır.

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f^{(3)}(x_0)}{3!}(x - x_0)^3 + ... + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

İlk bölümümüzde Taylor denkleminin özel formuna ($x_0 = 0$) MacLaurin denklemi ismini verdiğimizi görmüştük.

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f^{(3)}(0)}{3!}x^3 + ... + \frac{f^{(n)}(0)}{n!}x^n$$

$$f(x) = \sum_{i=0}^{\infty} a_i x^i$$

r(x) N inci dereceden MacLaurin serisine açılırsa

$$r(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_N x^N$$

F(x) ve r(x) terimlerinin farkını yazıp sonra r(x) fonksiyonu yerine bölümlü ifadeyi yazacak olursak:

$$f(x) - r(x) = f(x) - \frac{p(x)}{q(x)} = \frac{f(x)q(x) - p(x)}{q(x)} = \frac{\sum_{i=0}^{\infty} a_i x^i \sum_{i=0}^{m} q_i x^i - \sum_{i=0}^{n} p_i x^i}{q(x)}$$

Bu serilerde $q_0=1$ kabulünü yapabiliriz. Bu denklemdeki temel gayemiz eşitliği kullanarak $q_1,q_2,\ldots,q_m$ ve $p_1,p_2,\ldots,p_n$ terimlerini bulmaktır. Polinom eşitliklerinde aynı polinom derecelerinin katsayılarıda eşit olacaktır. Burada n+m=N dir ve bu yüzden denklemimizde x'in üstü olarak olabilecek en büyük değer N'dir.

$(a_0 + a_1 x + a_2 x^2 + ...)(1 + q_1 x + q_2 x^2 + ... + q_n x^m) - (p_0 + p_1 x + p_2 x^2 + ... + p_n x^n)$ has no

Denklemi basitleştirmek için $p_{n+1}=p_{n+2}=\ldots p_N=0$ ve $q_{m+1}=q_{m+2}=\ldots p_N=0$ şeklinde tanımlandı. Bundan sonra polinomdaki x^k teriminin katsayılarını eşitlediğimizde

$\left(\sum_{i=0}^{k} a_i q_{k-i}\right) - p_k$ deklemi elde edilir.

Bunun anlamı pade yaklaşımının N+1 denklemlik bir lineer denklem sisteminin çözümünden ibaret olacağıdır.

$$\left(\sum_{i=0}^{k} a_i q_{k-i}\right) = p_k \ , k = 0,..., N$$

N+1 bilinmiyenli denklemimizdeki bilinmiyenler $q_1, q_2, ... + q_m, p_o, p_1, p2, ... + qn$ dir.

İlk örnek olarak $f[x] = e^x$ fonksiyonunu alalım ve bu fonksiyona Pade yaklaşımını uygulayalım:

Fonksiyonumuzun MacLaurin açılımı f(x)= e^x $0(x^5)$ seviyesi hata ile şu şekilde bulunur. Önce türev değerlerini x=0 civarında buluruz.

$f(x) = e^x$, f(0)=1,

$f'(x) = e^x$, $f'(0) = 1$,

$f''(x) = e^x$, $f''(0) = 1$,

$f'''(x) = e^x$, $f'''(0) = 1$,

$f''''(x) = e^x$, $f''''(0) = 1$,

Maclaurin denklemimiz:

$$f(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + 0(x^5)$$

$$p(x) = p_0 + xp_1 + x^2 p_2$$

$$q(x) = 1 + xq_1 + x^2 q_2$$

$$f(x)q(x) - p(x) = 0$$

$$(1 - p_0) + (1 - p_1 + q_1)x + (\frac{1}{2} - p_2 + q_1 + q_2)x^2 + (\frac{1}{6} + \frac{q_1}{2} + q_2)x^3 + (\frac{1}{24} + \frac{q_1}{6} + \frac{q_2}{2})x^4 + 0[x]^5 = 0$$

$$1 - p_0 = 0$$

$$1 - p_1 + q_1 = 0$$

$$\frac{1}{2} - p_2 + q_1 + q_2 = 0$$

$$\frac{1}{6} + \frac{q_1}{2} + q_2 = 0$$

$$\frac{1}{24} + \frac{q_1}{6} + \frac{q_2}{2} = 0$$

Bu denklem sistemini çözersek:

$$p_0 = 1$$

$$p_1 = \frac{1}{2}$$

$$p_2 = \frac{1}{12}$$

$$q_1 = -\frac{1}{2}$$

$$q_2 = \frac{1}{12}$$

ve $r(x) = \dfrac{1 + \frac{1}{2}x + \frac{1}{12}x^2}{1 - \frac{1}{2}x + \frac{1}{12}x^2}$

denklemi sadeleştirdiğimizde

$r(x) = \dfrac{12 + 6x + x^2}{12 - 6x + x^2}$ bulunur. Bu yaklaşımdaki maksimum hata miktarı x=1 için

f(x)-r(x)= $e^x - \dfrac{12 + 6x + x^2}{12 - 6x + x^2}$ = 0,00399611 olarak bulunur.

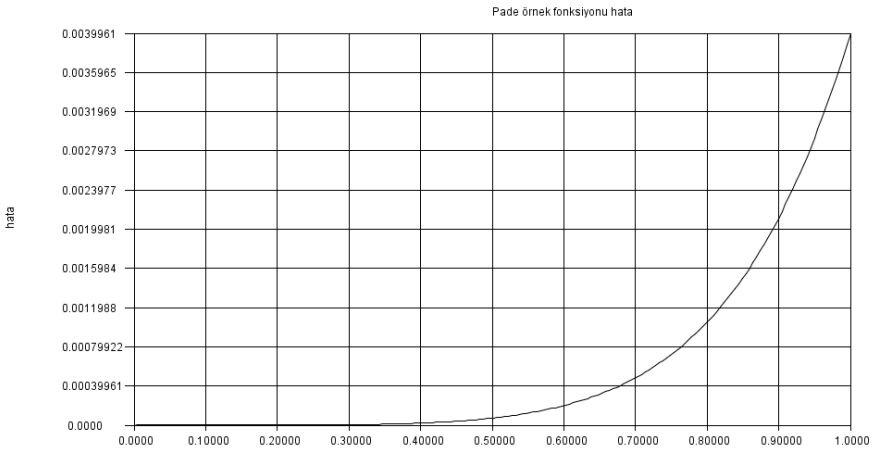
Hatanın daha detaylı irdelenmesi için küçük bir hata programı oluşturalım:

Program 6.16.1 Pade bölümlü fonksiyon yaklaşımı örnek problem hatası pade_hata.java

```
class fe extends f_x
{public double func(double x)
 {return Math.exp(x)-(12+6*x+x*x)/(12-6*x+x*x);}
}
public class pade_hata
{
public static void main(String args[])
  {
  fe f=new fe();
```

```
  String s="";
  for(double x=0;x<=2;x+=0.1)
  {s+="x = "+x+" hata = "+f.func(x)+"\n";}
  System.out.println(s);
  Plot pp=new Plot(f,0,2,200);
  pp.setPlabel("Vandermonde interpolasyonu");
  pp.setXlabel("x");
  pp.setYlabel("hata");
  pp.setGrid(1,1);
  pp.plot();
  }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" pade_hata
x = 0.0 hata = 0.0
x = 0.1 hata = 1.5358732730064162E-8
x = 0.2 hata = 5.441380297455112E-7
x = 0.30000000000000004 hata = 4.580170755108526E-6
x = 0.4 hata = 2.1418952745655417E-5
x = 0.5 hata = 7.262205147950951E-5
x = 0.6 hata = 2.0099217133084935E-4
x = 0.7 hata = 4.8370867674973894E-4
x = 0.7999999999999999 hata = 0.0010511325741000555
x = 0.8999999999999999 hata = 0.0021132326144397418
x = 0.9999999999999999 hata = 0.003996114173331122
x = 1.0999999999999999 hata = 0.007191742554601976
x = 1.2 hata = 0.01242461504423975
x = 1.3 hata = 0.020739791558123333
x = 1.4000000000000001 hata = 0.033617233031725924
x = 1.5000000000000002 hata = 0.05311764176663569
x = 1.6000000000000003 hata = 0.08206468245963094
x = 1.7000000000000004 hata = 0.12426722115150657
x = 1.8000000000000005 hata = 0.18478259954808252
x = 1.9000000000000006 hata = 0.2702174826593158
```



(bolum6_028.jpg,Resimaltı: **Pade örnek problemi hata grafiği)**

Görüldüğü gibi hata x=0 dan saptıkça artmaktadır. Şimdi aynı problemi bilgisayar programı kullanarak çözelim. Programımızda 8inci dereceye kadar MacLaurin serisi katsayıları f_x sınıfında tanımlanan fark denklemleriyle

hesaplanabilmektedir. Tabi ki fark denklemi hesabı belli bir hatayı içerecektir. İstenildiğinde bu değerler elde hesaplanarak vektör olarak da verilebilir.

Program 6.16.2 Pade bölümlü fonksiyon yaklaşımı pade.java

```
 import java.io.*;
import javax.swing.*;
class ff extends f_x
{
public double func(double x)
{return Math.exp(x);}
}

//Pade approximation
class pade
{
public double q[];
public double p[];
public double a[];

public static double[] Maclaurin(f_x f,double x,int n)
{
double a[]=new double[n+1];
double fact=1.0;
for(int i=0;i<=n;i++)
{a[i]=f.dfunc(x,i)/fact;fact*=(i+1);}
return a;
}

public pade(f_x f,int n,int m)
{
int N=m+n;
double a1[]=Maclaurin(f,0.0,N);
initilize(a1,n,m);
}

public pade(double a[],int n,int m)
{initilize(a,n,m);}

public void initilize(double ai[],int n,int m)
{
int N=m+n;
int NP1=N+1;
a=new double[NP1];
for(int i=0;i<NP1;i++)
{a[i]=ai[i];}
q=new double[m+1];
p=new double[n+1];
q[0]=1.0;
p[0]=a[0];
int i,j;
double b[][]=new double[N+1][N+1];
double bb[][]=new double[N][N];
double c[]=new double[N];
for(i=1;i<=N;i++)
{ for(j=1;j<(i-1);j++)
  {if(j<=n) b[i][j]=0;}
  if(i<=n) b[i][i]=1;
  for(j=i+1;j<=N;j++)
```

```
 {b[i][j]=0;}
 for(j=1;j<=i;j++)
 {if(j<=m)b[i][n+j]=-a[i-j];}
 for(j=n+i+1;j<=N;j++)
 {b[i][j]=0;}
 c[i-1]=a[i];
}
for(i=0;i<N;i++)
{for(j=0;j<N;j++)
 {bb[i][j]=b[i+1][j+1];}
}
double x[]=pivotlugauss(bb,c);
for(i=0;i<n;i++)
{p[i+1]=x[i];}
for(i=n;i<N;i++)
{q[i+1-n]=x[i];}
}

 public static double[] pivotlugauss(double a[][],double b[])
 { //gauss elimination with partial pivoting
 int n=b.length;
 double x[]=new double[n];
 double carpan=0;
 double toplam=0;
 double buyuk;
 double dummy=0;
 //gauss elimination
 int i,j,k,p,ii,jj;
 for(k=0;k<(n-1);k++)
 { //pivoting
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
              if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
                a[k][jj]=dummy;
              }
              dummy=b[p];
              b[p]=b[k];
              b[k]=dummy;
            }
  for(i=k+1;i<n;i++)
  {  carpan=a[i][k]/a[k][k];
    a[i][k]=0;
    for(j=k+1;j<n;j++)
    {   a[i][j]-=carpan*a[k][j]; }
       b[i]   =b[i]   -carpan*b[k];
            }
 }
 x[n-1]=b[n-1]/a[n-1][n-1];
 for(i=n-2;i>=0;i--)
 {
  toplam=0;
```

```
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
  }

public double func(double x)
{
return poly_func(p,x)/poly_func(q,x);
}

public static double poly_func(double e[],double x)
{
// polinom katsayıları
int n=e.length;
double ff;
if(n!=0.0)
  { ff=e[n-1];
  for(int i=n-2;i>=0;i--)
   { ff=ff*x+e[i]; }
  }
else
  ff=0;
return ff;
}

public String toString()
{
            String s="a : \n"+Matrix.toStringT(a,30,10)+"\n";
                s+="p : \n"+Matrix.toStringT(p,30,10)+"\n";
                s+="q : \n"+Matrix.toStringT(q,30,10)+"\n";
            return s;
}

public String toString(f_x f,double a,double b,int n)
{

return Matrix.toStringT(result(f,a,b,n));
}

public double[][] result(f_x f,double a,double b,int n)
{   double xx[][]=new double[4][n+1];
    for(int i=0;i<=n;i++)
            { xx[0][i]=a+i*(b-a)/n;
              xx[1][i]=func(xx[0][i]);
              xx[2][i]=f.func(xx[0][i]);
              xx[3][i]=xx[2][i]-xx[1][i];
            }
            return xx;
}
public void plot_func(f_x f,double a,double b)
{           double xx[]=new double[101];
    double yy[]=new double[101];
            for(int i=0;i<=100;i++)
            {xx[i]=a+i*(b-a)/100.0;yy[i]=func(xx[i]);}
            Plot pp=new Plot(xx,yy);
    pp.setPlabel("Pad"+'\u00E9'+" bölümlü yaklaşım fonksiyonu");
```

```
            pp.plot();}

            public void plot_error(f_x f,double a,double b)
   { double xx[]=new double[101];
     double yy[]=new double[101];
            for(int i=0;i<=100;i++)
            {xx[i]=a+i*(b-a)/100.0;yy[i]=f.func(xx[i])-func(xx[i]);}
            Plot pp=new Plot(xx,yy);
   pp.setPlabel("Pad"+'\u00E9'+" bölümlü yaklaşım hatası");
   pp.setYlabel("hata");
            pp.plot();
}

public static void print(String s,String s1)
{JOptionPane.showMessageDialog(null,s,s1,JOptionPane.PLAIN_MESSAGE);}

public static void main(String arg[])
{ ff f=new ff();
  // coefficients of function exp(x)
double a[]={1.00000000000000000000,
1.00000000000000000000,
0.50000000000000000000,
0.16666666666667000000,
0.04166666666666700000,
0.00833333333333330000,
0.00138888888888890000,
0.00019841269841269800,
0.00002480158730158730,
0.00000275573192239859,
0.00000027557319223985,
0.00000002505210838544,
0.00000000208767569878,
0.00000000016059043836,
0.00000000001147074559,
0.00000000000076471637,
0.00000000000004779477,
0.00000000000000281145,
0.00000000000000015619
};
   int n=10;
            int m=10;
            pade Pa=new pade(f,n,m);
            System.out.println(Pa.toString());
            System.out.println(Matrix.toStringT(Pa.result(f,0.0,1.0,10),30,10));
            Pa.plot_func(f,0.0,1.0);
            Pa.plot_error(f,0.0,1.0);
}

}
```

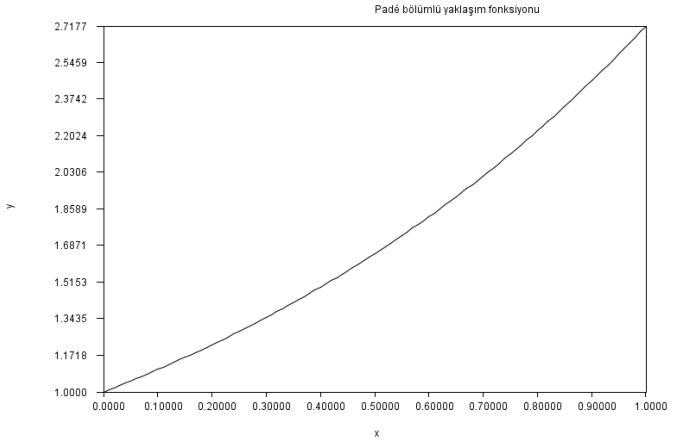
```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" pade
a :
            1.0000000000
            1.0000000000
            0.5000000000
            0.1666666667
            0.0416667485
            0.0083337990
```

0.0013638627
-0.0013654718
0.0461800517
-3.2568630254
-66.4903638085
1162.7778955521
70675.8699591349
-817052.8744371992
-41117001.1421074200
-34443972.8184028040
16472266774.8663180000
78154064880.1847200000
-739971964837.1558000000
-11822911158010.1910000000
25081771983594.7800000000

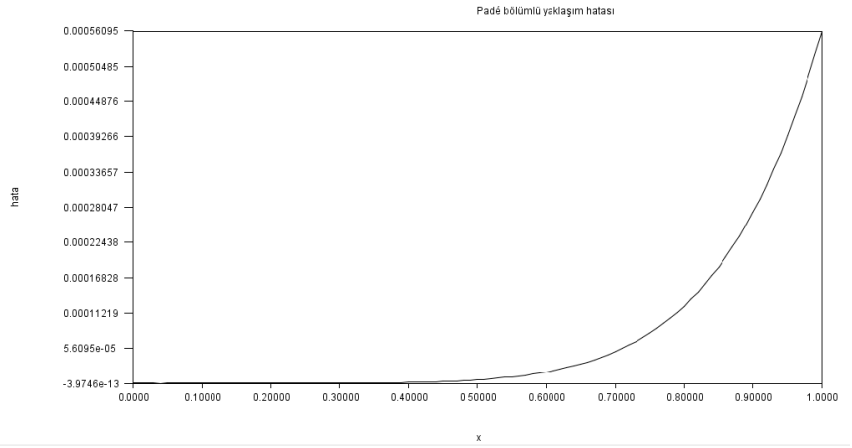
p :
1.0000000000
-54.6794767721
1003.9505573041
-25630.6734966360
76016.9707664278
7365749.0439267190
-155361366.7241304200
6908576843.9589410000
4686097094.9805120000
1301811728.0090620000
159051793.2627389700

q :
1.0000000000
-55.6794767721
1059.1300340762
-26662.1304589926
102158.7744544254
7276747.1246784420
-162684793.2154728500
7067607339.3246550000
-2301384674.9168344000
96202859.3321059600
42324278.7791172860

0.0000000000	1.0000000000	1.0000000000	0.0000000000
0.1000000000	1.1051709178	1.1051709181	0.0000000002
0.2000000000	1.2214027401	1.2214027582	0.0000000181
0.3000000000	1.3498585823	1.3498588076	0.0000002253
0.4000000000	1.4918233226	1.4918246976	0.0000013750
0.5000000000	1.6487155744	1.6487212707	0.0000056963
0.6000000000	1.8221003087	1.8221188004	0.0000184917
0.7000000000	2.0137019471	2.0137527075	0.0000507604
0.8000000000	2.2254176410	2.2255409285	0.0001232875
0.9000000000	2.4593303166	2.4596031112	0.0002727946
1.0000000000	2.7177208827	2.7182818285	0.0005609458



(bolum6_029.jpg,Resimaltı: **Pade fonksiyon grafiği)**



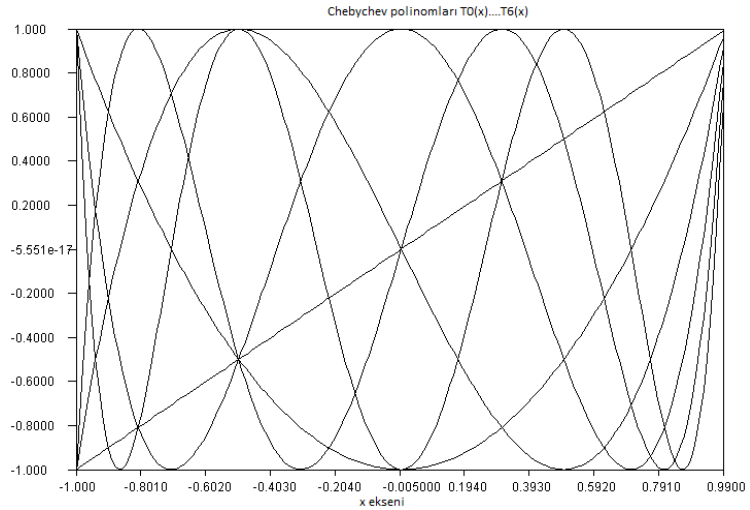
(bolum6_030.jpg,Resimaltı: **Pade hata grafiği)**

Görüldüğü gibi daha fazla katsayılarla yapılan hesapta hata azalmış, ancak hata karekteristiği aynı kalmıştır.

6.17 Bölümlü fonksiyon yaklaşımları : Chebyschev yaklaşım metodu

Pade yaklaşımının Taylor açılımına dayandığını görmüştük. Yaylor açılımını bir çok terim için yapmanın zorlukları yanı sıra hata terimlerinde de çok geniş bir yelpaze çizebilir. Daha sabit hata seviyelerine sahip bir metod istenilirse Chebyschev polinomları kullanabiliriz. Daha önce Chebyschev polinomlarını ortagonal en küçük kareler metodu olarak kullanmıştık. Temel tanımımız:

$T_n(x)=\cos(n \arccos(x))$
If this equation is written in open form
$T_0(x)=1$
$T_1(x)=x$
$T_2(x)=2x^2-x$
…..
$T_{n+1}(x)=2x\ T_n(x)- T_{n-1}(x)$



(bolum6_031.jpg,Resimaltı: Chebyschev polinomlarını ilk 6 serisinin grafik gösterimi**)**

Pade yaklaşımındaki x^k lı terimlerin yerini bu yaklaşımda $T_k(x)$ Chebyschev serileri alacaktır. Eğer N dereceli bir bölümlü fonksiyon yaklaşımı istenilirse:

$$r(x)=\frac{p(x)}{q(x)}=\frac{\sum_{k=0}^{n} p_k T_k(x)}{\sum_{k=0}^{m} q_k T_k(x)}$$ burada N=n+m ve q_0=1olarak alınmaktadır.

f(x) fonksiyonunu Chebyschev polinom serisi olarak yazarsak:

$$f(x)=\sum_{k=0}^{\infty} a_k T_k(x)$$ gives

$$f(x)-r(x)=f(x)-\frac{p(x)}{q(x)}=\frac{\sum_{k=0}^{\infty} a_k T_k(x)\sum_{k=0}^{m} q_k T_k(x)-\sum_{k=0}^{n} p_k T_k(x)}{\sum_{k=0}^{m} q_k T_k(x)}$$

Denklemin payda kısmı $q_1,q_2,\ldots,q_m$ ve $p_1,p_2,\ldots,p_n$ bilinmiyenlerini ve $T_k(x)$, k=0,1,..N. Chebychev polinomlarını içerecek şekilde açılırsa:

$$(a_0T_0(x)+a_1T_1(x)+\ldots)(T_0(x)+q_1T_1(x)+\ldots+q_mT_m(x))$$
$$-(p_0T_0(x)+p_1T_1(x)+\ldots+p_mT_m(x))$$

Bu denklemde N'inci dereceyi geçen bir terim bulunmamaktadır. Bu denklemin çözümündeki ana problemimiz Chebyschev terimlerinin çarpanlarının da gelmiş olmasıdır. Çarpan problemini

$$T_i(x)T_j(x)=0.5\left[T_{i+j}(x)+T_{|i-j|}(x)\right]$$

İlişkisini kullanarak çözeriz. Katsayıların integral denklemleri kullanılarak hesaplanması gerekmektedir.

$$a_0=\frac{1}{\pi}\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}dx=\frac{1}{\pi}\int_{0}^{\pi} f(\cos(\theta))d\theta$$

$$a_k=\frac{2}{\pi}\int_{-1}^{1}\frac{f(x)T_k(x)}{\sqrt{1-x^2}}dx=\frac{2}{\pi}\int_{0}^{\pi} f(\cos(\theta))\cos(k\theta)d\theta$$

Bu güçlük sayısal integral denklemleri kullanılarak giderilebilir. 7inci bölümde sayısal integrallerin detayını vereceğiz. Lütfen detaylar için bu bölümümüze başvurunuz. Örnek programımızda Gander ve Gautschi uyarlanabilir Gauss-Lobatto integral formülü kullanılmıştır.

Program 6.17.1 Chebyschev rational function approximation program Chebyshev.java

```
 import java.io.*;
import javax.swing.*;
class ff extends f_x
{
```

```
public double func(double x)
{return Math.exp(x);}
}

//Chebyshev  bölmeli yaklaşımı
class Chebyshev
{
public double q[];
public double p[];
public double a[];

        public static double Ti(double x,int k)
        {//Chebysev fonksiyonu
        double T[]=new double[k+1];
        T[0]=1.0;
        if(k>=1) T[1]=x;
        for(int i=2;i<=k;i++) {T[i]=2.0*x*T[i-1]-T[i-2];
   }
        return T[k];
        }

   public  double fk(f_x f,int k,double x)
   {   double a=0;
      double mult=2.0/Math.PI;
           a=mult*f.func(Math.cos(x))*Math.cos(k*x);
           return a;
        }

public double adaptive_Lobatto_integral(f_x f,double a,double b,int k)
{
        int i;
        double m=(a+b)/2.0; double h=(b-a)/2.0;
   double alpha=Math.sqrt(2.0/3.0);
   double beta=1.0/Math.sqrt(5.0);
   double x1,x2,x3;
   x1=0.94288241569547971905635175843185720232;
   x2=0.64185334234578130578123554132903188354;
   x3=0.23638319966214988028222377349205292599;
   double A,B,C,D,E,F,G;
   A=0.015827191973480183087169986733305510591;
   B=0.094273840218850045531282505077108171960;
   C=0.155071987336585396253635979802102986 80;
   D=0.188821573960182454420005339372971671 25;
   E=0.199773405226858526792068022066488402 46;
   F=0.224926465333339527016017687996395080 76;
   G=0.242611071901407733799640957903256352 33;
        double x[]={a,(m-x1*h),(m-alpha*h),(m-x2*h),(m-beta*h),(m-
x3*h),m,(m+x3*h),(m+beta*h),(m+x2*h),(m+alpha*h),(m+x1*h),b};
        int n=x.length;
        double y[]=new double[n];
        for(i=0;i<n;i++) y[i]=fk(f,k,x[i]);
        double fa=y[0];
```

```
            double fb=y[12];
    double i2=(h/6.0)*(y[0]+y[12]+5.0*(y[4]+y[8]));
    double i1=(h/1470.0)*(77.0*(y[0]+y[12])+432.0*(y[2]+y[10])+
     625.0*(y[4]+y[8])+672.0*y[6]);
    double
is=h*(A*(y[0]+y[12])+B*(y[1]+y[11])+C*(y[2]+y[10])+D*(y[3]+y[9])+E*(y[4]+y[8])+F*(y[5]+y[7])+G*y[6]);
    int s=(int)(is/Math.abs(is));
    if(s==0) s=1;
    double erri1=Math.abs(i1-is);
    double erri2=Math.abs(i2-is);
    double R;
    R=erri1/erri2;
    //if(R>0 && R<1) tol=tol/R;
    //is=s*Math.abs(is)*tol/eps;
    if(is==0) is=b-a;
  double Q=0;
  Q=Lobatto(f,a,b,fa,fb,is,k);
  return Q;
}

public  double Lobatto(f_x f,double a,double b,double fa,double fb,double is,int k)
{
  double h=(b-a)/2;
  double m=(a+b)/2;
  double alpha=Math.sqrt(2.0/3.0);
  double beta=1.0/Math.sqrt(5.0);
  double mll=m-alpha*h;
  double ml=m-beta*h;
  double mr=m+beta*h;
  double mrr=m+alpha*h;
  double x[]={mll,ml,m,mr,mrr};
  int n=x.length;
  double y[]=new double[n];
  for(int i=0;i<n;i++) y[i]=fk(f,k,x[i]);
  double fmll=y[0];
  double fml=y[1];
  double fm=y[2];
  double fmr=y[3];
  double fmrr=y[4];
  double Q;
  double i2=(h/6)*(fa+fb+5*(fml+fmr));
  double i1=(h/1470)*(77*(fa+fb)+432*(fmll+fmrr)+625*(fml+fmr)+672*fm);
  boolean c1,c2,c3;
  double w1=i1-i2;;
  double w2=is;
  double w3=w2+w1;
  c1=(w3==w2);
  c2=(mll<=a);
  c3=(b<=mrr);
  if( c1|| c2 ||c3 )
  {
   if (((m <= a) || (b<=m)))
```

```
   {
      System.out.println("istenilen toleransa ulaşılamadı");
   }
   Q=i1;
  }
  else
  {
    Q=Lobatto(f,a,mll,fa,fmll,is,k) + Lobatto(f,mll,ml,fmll,fml,is,k)+
      Lobatto(f,ml,m,fml,fm,is,k)   + Lobatto(f,m,mr,fm,fmr,is,k)+
      Lobatto(f,mr,mrr,fmr,fmrr,is,k)+Lobatto(f,mrr,b,fmrr,fb,is,k);
  }
  return Q;
}
//=============
public Chebyshev(f_x f,int n,int m)
{
int N=m+n;
int Npm=N+m;
double ai[]=new double[Npm+1];
int k;
for(k=0;k<=(N+m);k++)
{ai[k]=adaptive_Lobatto_integral(f,0.0,Math.PI,k);
}
initilize(ai,n,m);
}

public Chebyshev(double ai[],int n,int m)
{initilize(ai,n,m);}

public void initilize(double ai[],int n,int m)
{
int N=m+n;
int NP1=N+1;
int NPm=N+m;
int mp1=m+1;
int np1=n+1;
a=new double[NPm+1];
for(int i=0;i<=NPm;i++)
{a[i]=ai[i];}
q=new double[mp1];
p=new double[np1];
q[0]=1.0;
int i,j;
double b[][]=new double[NP1][NP1];
double c[]=new double[NP1];
for(i=0;i<=N;i++)
{  for(j=0;j<=i;j++)
   {if(j<=n) b[i][j]=0;}
if(i<=n) b[i][i]=1;
   for(j=i+1;j<=n;j++)
   {b[i][j]=0;}
   int k1,k2;
```

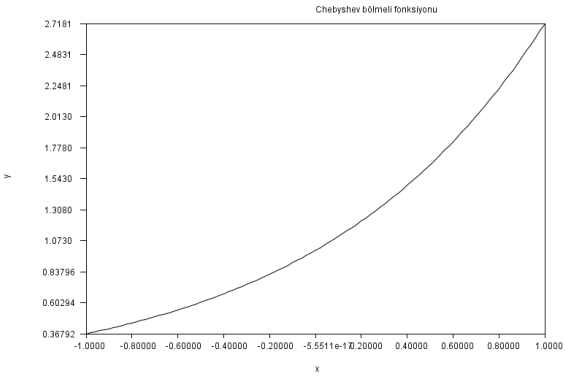
```
   for(j=n+1;j<=N;j++)
   {k1=i+j-n;
    k2=Math.abs(i-j+n);
   if(i!=0) b[i][j]=-0.5*(a[k1]+a[k2]);
   else     b[i][j]=-0.5*a[j-n];
  }
  if(i!=0) c[i]=a[i];
  else     c[i]=0.5*a[i];
}
double x[]=pivotlugauss(b,c);
for(i=0;i<=n;i++)
{p[i]=x[i];}
for(i=n+1;i<=N;i++)
{q[i-n]=x[i];}
}

  public static double[] pivotlugauss(double a[][],double b[])
  { //gauss eleme metodu
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss elimination
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //pivoting
              p=k;
              buyuk=Math.abs(a[k][k]);
              for(ii=k+1;ii<n;ii++)
              {  dummy=Math.abs(a[ii][k]);
                 if(dummy > buyuk) {buyuk=dummy;p=ii;}
              }
              if(p!=k)
              { for(jj=k;jj<n;jj++)
                { dummy=a[p][jj];
                  a[p][jj]=a[k][jj];
                  a[k][jj]=dummy;
                }
                dummy=b[p];
                b[p]=b[k];
                b[k]=dummy;
              }
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
          b[i]   =b[i]   -carpan*b[k];
              }
  }
```

```
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
  }

public double func(double x)
{
return poly_func(p,x)/poly_func(q,x);
}

public static double poly_func(double a[],double x)
{
// polinom fonksiyonu
int n=a.length;
double b=0;
for(int k=0;k<n;k++)
{b+=a[k]*Ti(x,k);}
return b;
}

public void plot_func(f_x f,double a,double b)
{           double xx[]=new double[101];
    double yy[]=new double[101];
            for(int i=0;i<=100;i++)
            {xx[i]=a+i*(b-a)/100.0;yy[i]=func(xx[i]);}
            Plot pp=new Plot(xx,yy);
    pp.setPlabel("Chebyshev bölmeli fonksiyonu");
            pp.plot();}

            public void plot_error(f_x f,double a,double b)
    { double xx[]=new double[101];
      double yy[]=new double[101];
            for(int i=0;i<=100;i++)
            {xx[i]=a+i*(b-a)/100.0;yy[i]=f.func(xx[i])-func(xx[i]);}
            Plot pp=new Plot(xx,yy);
    pp.setPlabel("Chebyshev bölmeli fonksiyon hatası");
    pp.setYlabel("hata");
            pp.plot();
}

public String toString()
{
            String s="p : \n"+Matrix.toStringT(p)+"\n";
                s+="q : \n"+Matrix.toStringT(q)+"\n";
            return s;
}
```

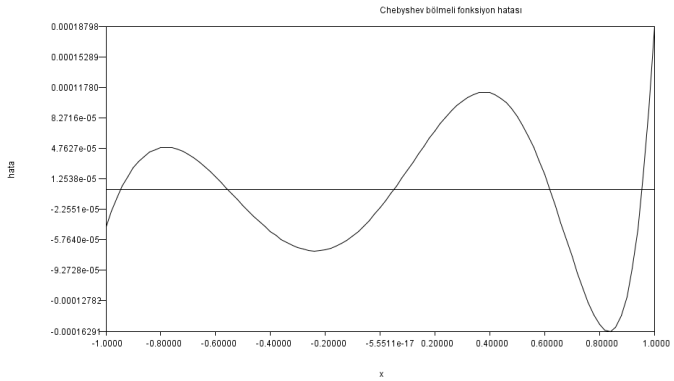
```
public String toString(f_x f,double a,double b,int n)
{

return Matrix.toStringT(result(f,a,b,n));
}

public double[][] result(f_x f,double a,double b,int n)
{   double xx[][]=new double[4][n+1];
   for(int i=0;i<=n;i++)
            { xx[0][i]=a+i*(b-a)/n;
              xx[1][i]=func(xx[0][i]);
              xx[2][i]=f.func(xx[0][i]);
              xx[3][i]=xx[2][i]-xx[1][i];
            }
            return xx;
}


public static void print(String s,String s1)
{JOptionPane.showMessageDialog(null,s,s1,JOptionPane.PLAIN_MESSAGE);}
public static void main(String arg[])
{
            ff f=new ff();
            int n=2;
            int m=2;
            Chebyshev Pa=new Chebyshev(f,n,m);
            System.out.println("Chebyshev bölmeli fonksiyon yaklaşımı katsayıları\n"+Pa.toString());
            String s1[]={"x","Chebyschev f(x)=p(x)/q(x)","f(x)","error"};
            System.out.println("Chebyshev rational polynomials n = "+n+" m =
"+m+"\n"+Matrix.toStringT(Pa.result(f,-1.0,1.0,20),30,10));
            Pa.plot_func(f,-1.0,1.0);
            Pa.plot_error(f,-1.0,1.0);
}

}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" Chebyshev
Chebyshev bölmeli fonksiyon yaklaşımı katsayıları
p :
      1.001004762421907
      0.482571445621581
      0.039718504110907

q :
      1.000000000000000
     -0.478306269991984
      0.038733833601555


Chebyshev rational polynomials n = 2 m = 2
```

-1.0000000000	0.3679215992	0.3678794412	-0.0000421580
-0.9000000000	0.4065450309	0.4065696597	0.0000246289
-0.8000000000	0.4492809903	0.4493289641	0.0000479738
-0.7000000000	0.4965447237	0.4965853038	0.0000405801
-0.6000000000	0.5487968821	0.5488116361	0.0000147540
-0.5000000000	0.6065487349	0.6065306597	-0.0000180752
-0.4000000000	0.6703679499	0.6703200460	-0.0000479039
-0.3000000000	0.7408849752	0.7408182207	-0.0000667546
-0.2000000000	0.8188000534	0.8187307531	-0.0000693004
-0.1000000000	0.9048908806	0.9048374180	-0.0000534626
0.0000000000	1.0000209015	1.0000000000	-0.0000209015
0.1000000000	1.1051481981	1.1051709181	0.0000227200
0.2000000000	1.2213348770	1.2214027582	0.0000678812
0.3000000000	1.3497567930	1.3498588076	0.0001020146
0.4000000000	1.4917133441	1.4918246976	0.0001113536
0.5000000000	1.6486369404	1.6487212707	0.0000843303
0.6000000000	1.8221015644	1.8221188004	0.0000172360
0.7000000000	2.0138295972	2.0137527075	-0.0000768897
0.8000000000	2.2256957689	2.2255409285	-0.0001548404
0.9000000000	2.4597266816	2.4596031112	-0.0001235705
1.0000000000	2.7180938467	2.7182818285	0.0001879818



(bolum6_032.jpg,Resimaltı: Chebyschev bölmeli fonksiyonu**)**



(bolum6_033.jpg,Resimaltı: Chebyschev bölmeli fonksiyonu hata grafiği**)**

6.18 Trigonometrik fonksiyon (Fourier) yaklaşımı

Sinis ve kosinüslü terimlerin serilerinden uzun süredir yararlanılmaktadır. [-□□□] aralığında bir yaklaşım polinomu olarak

$$S_n(x) = \frac{a_0}{2} + a_n \cos(nx) + \sum_{k=1}^{n-1}\left[a_k \cos(kx) + b_k \sin(kx)\right]$$

Serisini göz önüne alalım.Bu seri [-□□□] aralığında ağırlık fonksiyonu w(x)=1 olan ortagonal bir seridir. Bu yüzden seri katsayılarının bulunabilmesi için ortagonal fonksiyonların temel özelliklerinden yararlanabiliriz. [a,b] aralığında ortagonal olan bir $\Phi_n(x)$ fonksiyonu için

$\int_a^b \Phi_n(x)\Phi_m(x)dx = \begin{Bmatrix} 0 \; n \neq m \, ise \\ A \; n = m \quad ise \end{Bmatrix}$ yazılabilir. Bu özelliği kullanırsak ve seriyi eşitlemek istediğimiz fonksiyonun f(x) olduğunu varsayarsak seri katsayıları

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)dx \quad \text{for k=0,1,2,…,n}$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(kx)dx \quad \text{for k=1,2,…,n-1}$$

ifadelerinden hesaplanabilir.

$S_n(x)$ serisinin $n \to \infty$ limit değerine **Fourier serisi adı verilir.** Seriyi daha genel olarak [-l,l] bölgesi için yazarsak:

$$S(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}\left[a_k \cos(\frac{k\pi x}{l}) + b_k \sin(\frac{k\pi x}{l})\right]$$

$$a_k = \frac{1}{l}\int_{-l}^{l} f(x)\cos(\frac{k\pi x}{l})dx = \frac{1}{l}\int_{\alpha}^{\alpha+2l} f(x)\cos(\frac{k\pi x}{l})dx \quad \text{for k=0,1,2,…,n}$$

$$b_k = \frac{1}{l}\int_{-l}^{l} f(x)\sin(\frac{k\pi x}{l})dx = \frac{1}{l}\int_{\alpha}^{\alpha+2l} f(x)\sin(\frac{k\pi x}{l})dx$$

Trigonometrik polinom yaklaşımını hesaplayabilmek için bir program geliştirilmiştir. Temel olarak buradaki yaklaşımımız Chebyschev polinomundaki yaklaşımımıza çok benzemektedir. Burada da katsayıları hesaplamak için sayısal integral tekniklerini kullanacağız. İlk örnek fonksiyonumuz: f(x)=|x| olsun

Program 6.18.1 Trigonometrik fonksiyon yaklaşımı trigonometrik.java

```
 import java.io.*;
import javax.swing.*;
class ff extends f_x
{
public double func(double x)
{return Math.abs(x);}
}

// Trigonometrik  fonksiyon yaklaşımı(Fourier serisi)
class Trigonometrik
{
public double a[];
public double b[];

public  double fak(f_x f,int k,double x)
```

```
   {   double a=0;
       double mult=1.0/Math.PI;
              a=mult*f.func(x)*Math.cos(k*x);
              return a;
            }

public  double fbk(f_x f,int k,double x)
   {   double a=0;
       double mult=1.0/Math.PI;
               a=mult*f.func(x)*Math.sin(k*x);
              return a;
            }

public double adaptive_Lobatto_integral(f_x f,double a,double b,int k,int eqn_type)
{
            int i;
            double m=(a+b)/2.0; double h=(b-a)/2.0;
   double alpha=Math.sqrt(2.0/3.0);
   double beta=1.0/Math.sqrt(5.0);
   double x1,x2,x3;
   x1=0.94288241569547971905635175843185720232;
   x2=0.64185334234578130578123554132903188354;
   x3=0.23638319966214988028222377349205292599;
   double A,B,C,D,E,F,G;
   A=0.015827191973480183087169986733305510591;
   B=0.094273840218850045531282505077108171960;
   C=0.155071987336585396253635979802102986 80;
   D=0.188821573960182454420005339372971671 25;
   E=0.199773405226858526792068022066488402 46;
   F=0.224926465333339527016017687996395080 76;
   G=0.242611071901407733799640957903256352 33;
            double x[]={a,(m-x1*h),(m-alpha*h),(m-x2*h),(m-beta*h),(m-
x3*h),m,(m+x3*h),(m+beta*h),(m+x2*h),(m+alpha*h),(m+x1*h),b};
            int n=x.length;
            double y[]=new double[n];
            for(i=0;i<n;i++)
            {
            if(eqn_type==0)
            {y[i]=fak(f,k,x[i]);}
            else
            {y[i]=fbk(f,k,x[i]);}
   }
            double fa=y[0];
            double fb=y[12];
   double i2=(h/6.0)*(y[0]+y[12]+5.0*(y[4]+y[8]));
   double i1=(h/1470.0)*(77.0*(y[0]+y[12])+432.0*(y[2]+y[10])+
    625.0*(y[4]+y[8])+672.0*y[6]);
   double
is=h*(A*(y[0]+y[12])+B*(y[1]+y[11])+C*(y[2]+y[10])+D*(y[3]+y[9])+E*(y[4]+y[8])+F*(y[5]+y[7])+G*y[6]);
   int s=(int)(is/Math.abs(is));
   if(s==0) s=1;
   double erri1=Math.abs(i1-is);
```

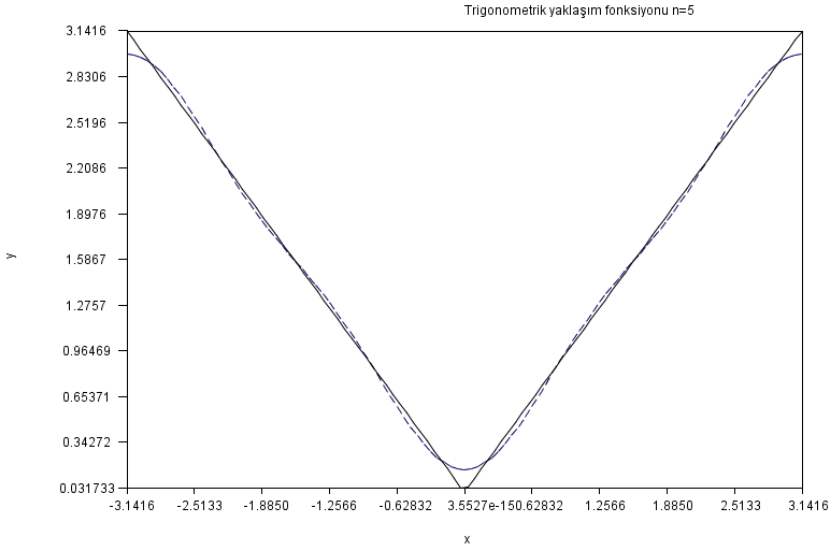
```
    double erri2=Math.abs(i2-is);
    double R;
    R=erri1/erri2;
    //if(R>0 && R<1) tol=tol/R;
    //is=s*Math.abs(is)*tol/eps;
    if(is==0) is=b-a;
  double Q=0;
  Q=Lobatto(f,a,b,fa,fb,is,k,eqn_type);
  return Q;
}

public  double Lobatto(f_x f,double a,double b,double fa,double fb,double is,int k,int eqn_type)
{
  double h=(b-a)/2;
  double m=(a+b)/2;
  double alpha=Math.sqrt(2.0/3.0);
  double beta=1.0/Math.sqrt(5.0);
  double mll=m-alpha*h;
  double ml=m-beta*h;
  double mr=m+beta*h;
  double mrr=m+alpha*h;
  double x[]={mll,ml,m,mr,mrr};
  int n=x.length;
  double y[]=new double[n];
  for(int i=0;i<n;i++)
            {
    if(eqn_type==0)
            {y[i]=fak(f,k,x[i]);}
            else
            {y[i]=fbk(f,k,x[i]);}
    }
  double fmll=y[0];
  double fml=y[1];
  double fm=y[2];
  double fmr=y[3];
  double fmrr=y[4];
  double Q;
  double i2=(h/6)*(fa+fb+5*(fml+fmr));
  double i1=(h/1470)*(77*(fa+fb)+432*(fmll+fmrr)+625*(fml+fmr)+672*fm);
  boolean c1,c2,c3;
  double w1=i1-i2;;
  double w2=is;
  double w3=w2+w1;
  c1=(w3==w2);
  c2=(mll<=a);
  c3=(b<=mrr);
  if( c1|| c2 ||c3 )
  {
   if (((m <= a) || (b<=m)))
   {
      System.out.println("istenilen toleransa ulaşılamadı");
   }
```

```
   Q=i1;
  }
  else
  {
    Q=Lobatto(f,a,mll,fa,fmll,is,k,eqn_type) + Lobatto(f,mll,ml,fmll,fml,is,k,eqn_type)+
      Lobatto(f,ml,m,fml,fm,is,k,eqn_type)   + Lobatto(f,m,mr,fm,fmr,is,k,eqn_type)+
      Lobatto(f,mr,mrr,fmr,fmrr,is,k,eqn_type)+Lobatto(f,mrr,b,fmrr,fb,is,k,eqn_type);
  }
  return Q;
}

//=============


public Trigonometrik(f_x f,int n)
{
double ai[]=new double[n+1];
double bi[]=new double[n+1];
for(int k=0;k<n;k++)
{ai[k]=adaptive_Lobatto_integral(f,-Math.PI,Math.PI,k,0);}
bi[0]=0.0;
for(int k=1;k<n;k++)
{bi[k]=adaptive_Lobatto_integral(f,-Math.PI,Math.PI,k,1);}
bi[n]=0;
initilize(ai,bi,n);
}

public Trigonometrik(double ai[],double bi[],int n)
{initilize(ai,bi,n);}

public void initilize(double ai[],double bi[],int n)
{
a=new double[n+1];
b=new double[n+1];
for(int i=0;i<=n;i++)
{a[i]=ai[i];b[i]=bi[i];}
}

public  double func(double x)
{
int n=a.length-1;
double cc=a[0]/2.0+a[n]*Math.cos(n*x);
for(int k=1;k<n;k++)
{cc+=a[k]*Math.cos(k*x)+b[k]*Math.sin(k*x);}
return cc;
}

public void plot_func(f_x f,double x0,double xn)
{           double xx[]=new double[101];
   double yy[]=new double[101];
            for(int i=0;i<=100;i++)
            {xx[i]=x0+i*(xn-x0)/100.0;yy[i]=func(xx[i]);}
```

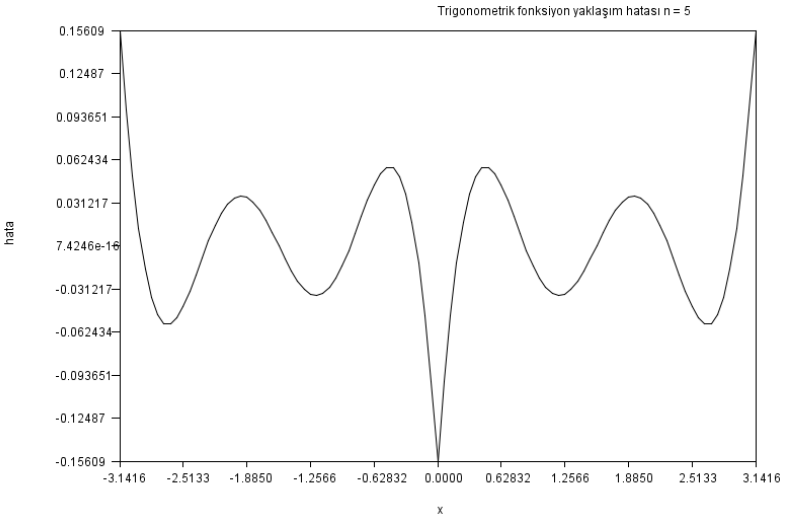
```
            Plot pp=new Plot(xx,yy);
            pp.addFunction(f,x0,xn,100);
   pp.setPlabel("Trigonometrik yaklaşım fonksiyonu n="+(a.length-1));
            pp.plot();
}

            public void plot_error(f_x f,double x0,double xn)
   { double xx[]=new double[101];
     double yy[]=new double[101];
            for(int i=0;i<=100;i++)
            {xx[i]=x0+i*(xn-x0)/100.0;yy[i]=f.func(xx[i])-func(xx[i]);}
            Plot pp=new Plot(xx,yy);
   pp.setPlabel("Trigonometrik fonksiyon yaklaşım hatası n = "+(a.length-1));
   pp.setYlabel("hata");
            pp.plot();
}

public String toString()
{
            String s="a : \n"+Matrix.toStringT(a)+"\n";
                 s+="b : \n"+Matrix.toStringT(b)+"\n";
            return s;
}

public String toString(f_x f,double a,double b,int n)
{

return Matrix.toStringT(result(f,a,b,n));
}

public double[][] result(f_x f,double a,double b,int n)
{   double xx[][]=new double[4][n+1];
   for(int i=0;i<=n;i++)
            { xx[0][i]=a+i*(b-a)/n;
              xx[1][i]=func(xx[0][i]);
              xx[2][i]=f.func(xx[0][i]);
              xx[3][i]=xx[2][i]-xx[1][i];
            }
            return xx;
}



public static void print(String s,String s1)
{JOptionPane.showMessageDialog(null,s,s1,JOptionPane.PLAIN_MESSAGE);}
public static void main(String arg[])
{
            ff f=new ff();
            int n=5;
            Trigonometrik Pa=new Trigonometrik(f,n);
            System.out.println("Trigonometrik seri katsayıları n="+n+"\n"+Pa.toString());
            String s1[]={"x","Trigonometrik f(x)","f(x)","error"};
            System.out.println("Trigonometrik polyinom n = "+n+"\n"+Matrix.toStringT(Pa.result(f,-
```

```
Math.PI,Math.PI,20),30,10));
        Pa.plot_func(f,-Math.PI,Math.PI);
        Pa.plot_error(f,-Math.PI,Math.PI);
}

}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" Trigonometrik
Trigonometrik seri katsayıları n=5
a :
     3.141592653589793
    -1.273239544735163
    -0.000000000000000
    -0.141471060526129
    -0.000000000000000
     0.000000000000000

b :
     0.000000000000000
     0.000000000000000
     0.000000000000000
     0.000000000000000
     0.000000000000000
     0.000000000000000

Trigonometrik polinom n = 5
        -3.1415926536    2.9855069321    3.1415926536    0.1560857215
        -2.8274333882    2.8648736956    2.8274333882   -0.0374403074
        -2.5132741229    2.5571517945    2.5132741229   -0.0438776716
        -2.1991148575    2.1846407798    2.1991148575    0.0144740777
        -1.8849555922    1.8497964919    1.8849555922    0.0351591003
        -1.5707963268    1.5707963268    1.5707963268   -0.0000000000
        -1.2566370614    1.2917961617    1.2566370614   -0.0351591003
        -0.9424777961    0.9569518737    0.9424777961   -0.0144740777
        -0.6283185307    0.5844408591    0.6283185307    0.0438776716
        -0.3141592654    0.2767189580    0.3141592654    0.0374403074
         0.0000000000    0.1560857215    0.0000000000   -0.1560857215
         0.3141592654    0.2767189580    0.3141592654    0.0374403074
         0.6283185307    0.5844408591    0.6283185307    0.0438776716
         0.9424777961    0.9569518737    0.9424777961   -0.0144740777
         1.2566370614    1.2917961617    1.2566370614   -0.0351591003
         1.5707963268    1.5707963268    1.5707963268   -0.0000000000
         1.8849555922    1.8497964919    1.8849555922    0.0351591003
         2.1991148575    2.1846407798    2.1991148575    0.0144740777
         2.5132741229    2.5571517945    2.5132741229   -0.0438776716
         2.8274333882    2.8648736956    2.8274333882   -0.0374403074
         3.1415926536    2.9855069321    3.1415926536    0.1560857215
```



(bolum6_034.jpg,Resimaltı: Trigonometrik fonksiyon n=5**)**



(bolum6_035.jpg,Resimaltı: Trigonometrik fonksiyon hata grafiği n=5**)**

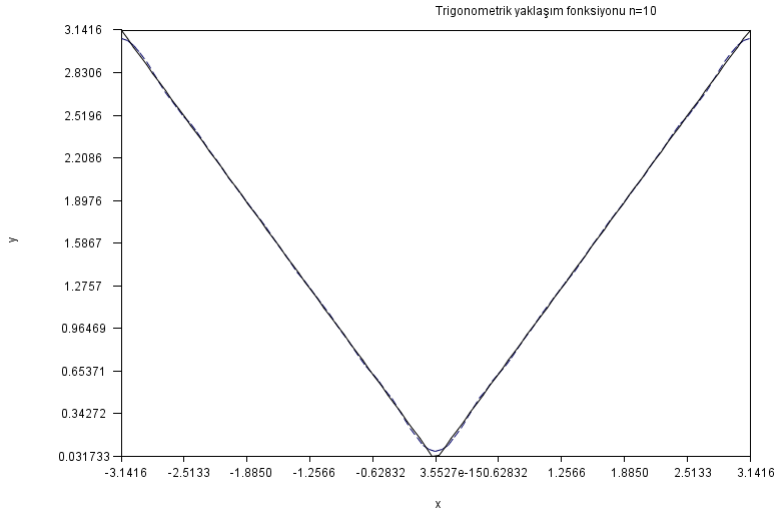
Katsayıları arttırdığımızda daha iyi bir uyum fonksiyonu elde ederiz.

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" Trigonometrik
Trigonometrik seri katsayıları n=10
a :
      3.141592653589793
     -1.273239544735163
     -0.000000000000000
     -0.141471060526129
     -0.000000000000000
     -0.050929581789406
     -0.000000000000000
     -0.025984480504799
     -0.000000000000000
     -0.015719006725125
      0.000000000000000

b :
      0.000000000000000
      0.000000000000000
```

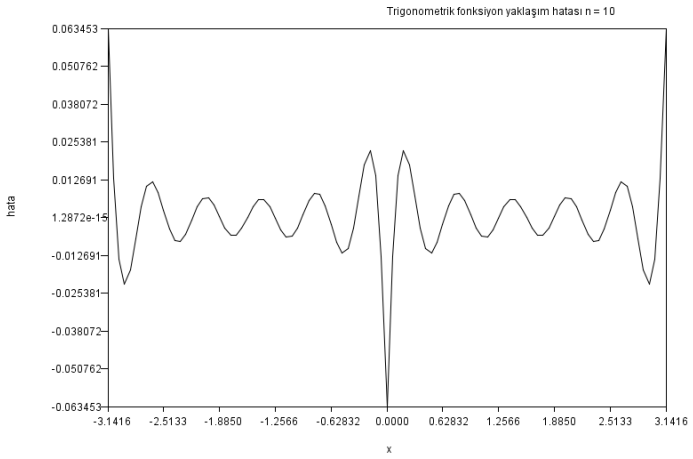
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000

Trigonometrik polinom n = 10

-3.1415926536	3.0781400011	3.1415926536	0.0634526525
-2.8274333882	2.8346507374	2.8274333882	-0.0072173492
-2.5132741229	2.5109095102	2.5132741229	0.0023646127
-2.1991148575	2.2001140890	2.1991148575	-0.0009992315
-1.8849555922	1.8845616275	1.8849555922	0.0003939646
-1.5707963268	1.5707963268	1.5707963268	-0.0000000000
-1.2566370614	1.2570310261	1.2566370614	-0.0003939646
-0.9424777961	0.9414785646	0.9424777961	0.0009992315
-0.6283185307	0.6306831434	0.6283185307	-0.0023646127
-0.3141592654	0.3069419162	0.3141592654	0.0072173492
0.0000000000	0.0634526525	0.0000000000	-0.0634526525
0.3141592654	0.3069419162	0.3141592654	0.0072173492
0.6283185307	0.6306831434	0.6283185307	-0.0023646127
0.9424777961	0.9414785646	0.9424777961	0.0009992315
1.2566370614	1.2570310261	1.2566370614	-0.0003939646
1.5707963268	1.5707963268	1.5707963268	-0.0000000000
1.8849555922	1.8845616275	1.8849555922	0.0003939646
2.1991148575	2.2001140890	2.1991148575	-0.0009992315
2.5132741229	2.5109095102	2.5132741229	0.0023646127
2.8274333882	2.8346507374	2.8274333882	-0.0072173492
3.1415926536	3.0781400011	3.1415926536	0.0634526525



(bolum6_036.jpg,Resimaltı: Trigonometrik fonksiyon n=10**)**



(bolum6_037.jpg,Resimaltı: Trigonometrik fonksiyon hata grafiği n=10**)**

Serimizin direk olarak verilere uygulanabilen ikinci bir formu daha mevcuttur. Bu formda fonksiyon yerine veri kullandığımız için yaptığımız işlemin bir eğri uydurma işlemi olduğunu söyleyebiliriz. Elimizde 2m veri seti points (x_j,y_j) j=0..2m-1 olduğunu düşünelim. Veri seti aralığı [-□□□] olsun. Şüphesiz başka bir veri aralığı da verilebilir. Bu durumda veri değişimiyle bu aralığa geçeriz. Serimizin veri için eşdeğer yazılımı:

$$S_n(x) = \frac{a_0}{2} + a_n \cos(nx) + \sum_{k=1}^{n-1}\left[a_k \cos(kx) + b_k \sin(kx)\right]$$

Şeklindedir. Serinin hata terimi:

$$E(S_n) = \sum_{j=0}^{2m-1}\left\{ y_j - \left[\frac{a_0}{2} + a_n \cos(nx_j) + \sum_{k=1}^{n-1}\left[a_k \cos(kx_j) + b_k \sin(kx_j)\right]\right]\right\}^2$$

Olacaktır. Burada katsayıları öyle seçmeliyizki hata minimum olsun. Bunu yaparken yine ckosinüs ve sinüs fonksiyonlarının ortagonal olma özelliğinden yararlanabiliriz.

$$\sum_{j=0}^{2m-1} \Phi_k(x_j)\Phi_l(x_j) = 0 \quad for\, k \neq l$$

Özelliği (daha önce yazdığımız integral olarak yazılması gereken özelliğin toplama serisine uygulanmış hali) geçerli olacaktır. Bu durumda katsayılar matrisi:

$$a_k = \frac{1}{m}\sum_{j=0}^{2m-1} y_j \cos(kx_j)$$

$$b_k = \frac{1}{m}\sum_{j=0}^{2m-1} y_j \sin(kx_j)$$

Olur. Eğer veri bölgesi [-□□□] bölgesinin dışındaysa tarnsfer fonksiyonlarıyla değer değişimine gidebiliriz. Eğer veri bölgemiz [x0,xn] ise transfer fonksiyonları

$$z = \frac{2\pi(x - x0)}{(xn - x0)} - \pi$$

$$x = \frac{(xn - x0)(z + \pi)}{2\pi} + x0$$

Olacaktır. Bu durumda serimizi

$$S_n(z) = \frac{a_0}{2} + a_n \cos(nz) + \sum_{k=1}^{n-1} \left[a_k \cos(kz) + b_k \sin(kz) \right]$$

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos(kz_j)$$

$$b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \sin(kz_j)$$

Şeklinde yazabiliriz. Bu tür set hem bir fonksiyona, hem de direk olarak veriye uydurulabilir. Örnek olarak m=5 (2m-1=9) seçip $f(x) = x^4 - 3x^3 + 2x^2 \tan(x(x-1))$ fonksiyonuna eğri uyduralım. Önce işlemi excel veriçizelgesi ortamında yapalım.

FOURIER (TRİGONOMETRİK SERİ) EĞRİ

	xj			***k***	***0***	***1***	***2***	***3***				
j	**j/5**	**z_j**		**f(x)**	**a_0**	**a_1**	**a_2**	**a_3**	**b_0**	**b_1**	**b_2**	**b_3**
0	0	-3.14159	-3.14159	0	0	0	0	0	0	0	0	0
1	0.2	-2.51327	-2.51327	0.434003	0.086801	-0.07022	0.026823	0.026823	0	-0.05102	0.0825523	-0.08255
2	0.4	-1.88496	-1.88496	0.898144	0.179629	-0.05551	-0.145323	0.145323	0	-0.170837	0.1055831	0.105583
3	0.6	-1.25664	-1.25664	1.317232	0.263447	0.081409	-0.213133	-0.213133	0	-0.250553	-0.15485	0.15485
4	0.8	-0.62832	-0.62832	1.581957	0.316392	0.255966	0.09777	-0.09777	0	-0.18597	-0.300906	-0.30091
5	1	0	0	1.557408	0.311482	0.311482	0.311482	0.311482	0	0	0	0
6	1.2	0.62832	0.62832	1.197957	0.239592	0.193834	0.074038	-0.074038	0	0.140828	0.2278651	0.227865
7	1.4	1.25664	1.25664	0.645232	0.129047	0.039878	-0.104401	-0.104401	0	0.122731	0.0758516	-0.07585
8	1.6	1.88496	1.88496	0.130144	0.026029	-0.00804	-0.021058	0.021058	0	0.024755	-0.015299	-0.0153
9	1.8	2.51327	2.51327	-0.141997	-0.0284	0.022976	-0.008776	-0.008776	0	-0.016693	0.0270095	-0.02701
10	2	3.14159	3.14159	0	0	0	0	0	0	0	0	0

7.620081	1.552416	0.771769	0.017423	0.006567	0	-0.386759	0.047806	-0.01332

x	**z**	**f(x)**	**$S_3(x)$**	**$\lvert f(x)-S_3(x) \rvert$**
0	-3.142	0	0.01529	0.015294
0.125	-2.749	0.2644	0.2548	0.009595
0.375	-1.964	0.84081	0.86573	0.024924
0.625	-1.178	1.3615	1.37668	0.015174
0.875	-0.393	1.61282	1.61826	0.005447
1.125	0.3927	1.36672	1.38986	0.023136
1.375	1.1781	0.71697	0.72965	0.012676

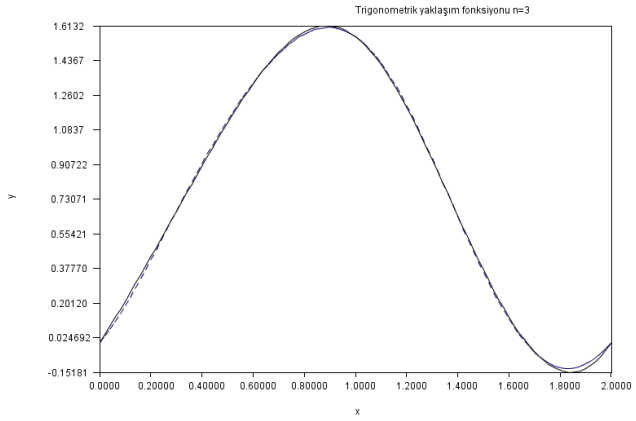
Aynı işlemi yapmak için bir java programı geliştirilmiştir. Program listesi aşağıda verilmektedir.

Program 6.18.2 fonksiyona trigonometrik dizi (Fourier) yaklaşımı Fourier.java

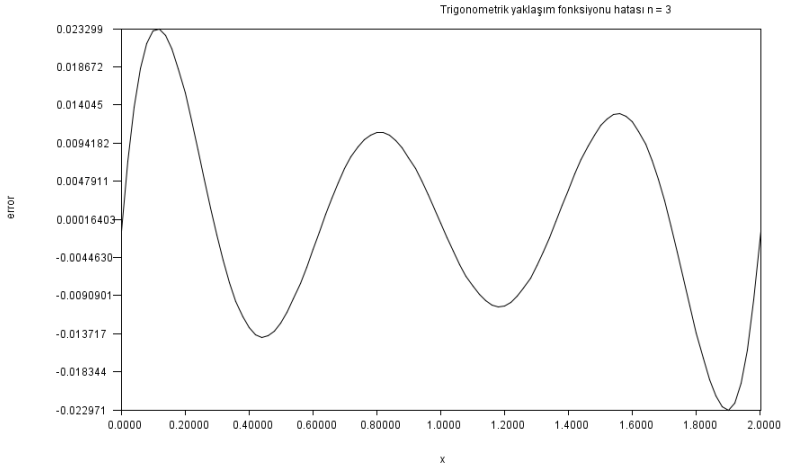
```
 import java.io.*;
import javax.swing.*;
class ff extends f_x
{
public double func(double x)
{return x*x*x*x-3.0*x*x*x+2.0*x*x-Math.tan(x*(x-2.0));}
}

// Fourier approximation
class Fourier
{
public double a[];
public double b[];
public double xj[];
public double zj[];
public double yj[];
public double x0,xn;
public int m; //(2*m+1) data points
public int n; //degree n Fourier function
```

```
public double z_x(double x)
{double zz=2.0*Math.PI*(x-x0)/(xn-x0)-Math.PI;
 return zz;
}

public double x_z(double z)
{double xx=(z+Math.PI)/(2.0*Math.PI)*(xn-x0)+x0;
 return xx;
}
public String toString()
{
                String s="a : \n"+Matrix.toStringT(a)+"\n";
                     s+="b : \n"+Matrix.toStringT(b)+"\n";
                return s;
}
public Fourier(f_x f,double x0i,double xni,int mi,int ni)
{
m=mi;
n=ni;
a=new double[n+1];
b=new double[n+1];
xj=new double[2*m+1];
zj=new double[2*m+1];
yj=new double[2*m+1];
x0=x0i;
xn=xni;
for(int k=0;k<=2*m;k++)
{xj[k]=x0+(xn-x0)/(2*m)*k;
 yj[k]=f.func(xj[k]);
 zj[k]=z_x(xj[k]);
}
  for(int k=0;k<=n;k++)
  {
  a[k]=0;
  b[k]=0;
  for(int j=0;j<2*m;j++)
    { a[k]+=1.0/m*yj[j]*Math.cos(k*zj[j]);
      b[k]+=1.0/m*yj[j]*Math.sin(k*zj[j]);
    }
  }
}

public  double func(double x)
{
double cc=a[0]/2.0+a[n]*Math.cos(n*z_x(x));
for(int k=1;k<n;k++)
{cc+=a[k]*Math.cos(k*z_x(x))+b[k]*Math.sin(k*z_x(x));}
return cc;
}

public void plot_func(f_x f,double x0i,double xni,int ni)
{
                double xx[][]=new double[4][ni+1];
                double x1=0;
   for(int i=0;i<=ni;i++)
                { x1=x0i+(xni-x0i)*i/ni;
                  xx[0][i]=x1;
                  xx[1][i]=func(x1);
                  xx[2][i]=f.func(x1);
                  xx[3][i]=xx[2][i]-xx[1][i];
                }
                Plot pp=new Plot(xx[0],xx[1]);
                pp.addData(xx[0],xx[2]);
   pp.setPlabel("Trigonometrik yaklaşım fonksiyonu n="+(a.length-1));
                pp.plot();
}
```

```
            public void plot_error(f_x f,double x0i,double xni,int ni)
   { double xx[][]=new double[4][ni+1];
              double x1=0;
     for(int i=0;i<=ni;i++)
              { x1=x0i+(xni-x0i)*i/ni;
              xx[0][i]=x1;
              xx[1][i]=func(x1);
              xx[2][i]=f.func(x1);
              xx[3][i]=xx[2][i]-xx[1][i];
              }
            Plot pp=new Plot(xx[0],xx[3]);
   pp.setPlabel("Trigonometrik yaklaşım fonksiyonu hatası n = "+(a.length-1));
   pp.setYlabel("error");
            pp.plot();
   }
public double[][] result(f_x f)
{   double xx[][]=new double[4][2*m];
   for(int i=0;i<2*m;i++)
            {
              xx[0][i]=xj[i];
              xx[1][i]=func(xj[i]);
              xx[2][i]=f.func(xj[i]);
              xx[3][i]=xx[2][i]-xx[1][i];
            }
            return xx;
}

public static void main(String arg[])
{
            ff f=new ff();
            int n=Integer.parseInt(JOptionPane.showInputDialog("trigonometrik yaklaşım fonksiyonu Sn(x) terim sayısı: n="));
            int m=Integer.parseInt(JOptionPane.showInputDialog("veri sayısı (2m-1): m="));
            double x0=0;
            double xn=2.0;
            Fourier Pa=new Fourier(f,x0,xn,m,n);
            System.out.println("Trigonometrik seri katsayıları n="+n+"\n"+Pa.toString());
            System.out.println("Trigonometrik polinom n = "+n+" m = "+m+"\n"+Matrix.toStringT(Pa.result(f),30,10));
   Pa.plot_func(f,x0,xn,100);
   Pa.plot_error(f,x0,xn,100);
}}
```



(bolum6_038.jpg,Resimaltı: Trigonometrik fonksiyon n=3**)**



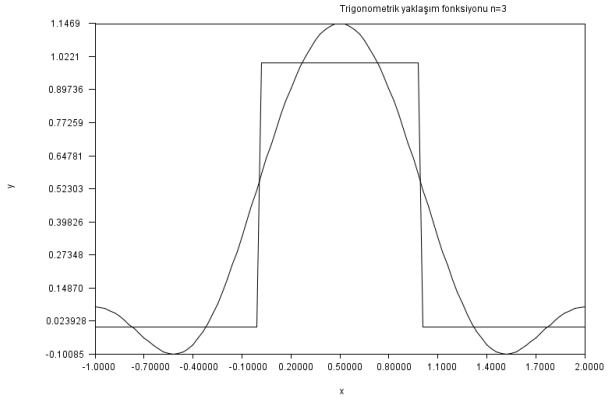
(bolum6_039.jpg,Resimaltı: Trigonometrik fonksiyon hata grafiği n=3**)**

Diğer bir örnek fonksiyonu irdeliyelim:

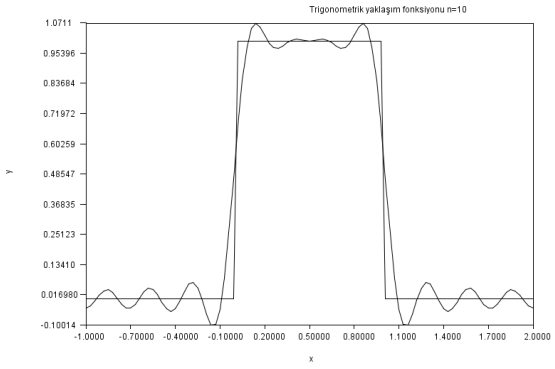
$$f(x) = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix} \quad \begin{matrix} x < 0 \ için \\ 0 <= x <= 1 \ için \\ x > 1 \ için \end{matrix}$$

```
class ff extends f_x
{
public double func(double x)
{double ff=0;
 if(x<0) ff=0;
 else if(x>=0 && x<=1) ff=1;
 else ff=0;
 return ff;}
}
```

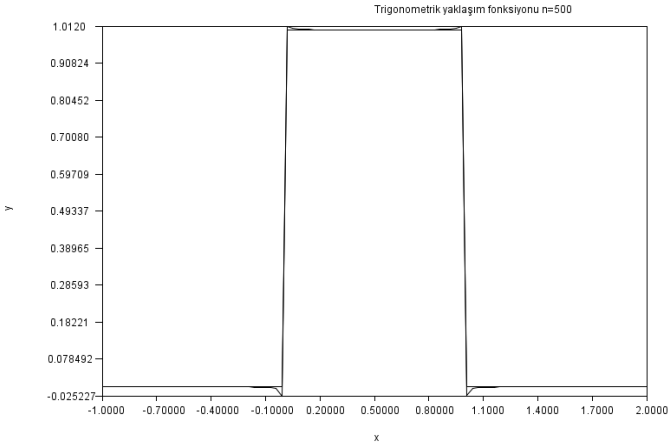
Bu fonksiyona değişik terim sayılarıyla $S_3(x)$, $S_{10}(x)$ and $S_{100}(x)$ fourier serisi eğrilerini uyduralım.



(bolum6_040.jpg,Resimaltı: Trigonometrik fonksiyon n=3**)**



(bolum6_041.jpg,Resimaltı: Trigonometrik fonksiyon n=10**)**



(bolum6_042.jpg,Resimaltı: Trigonometrik fonksiyon n=500**)**

Eğer veri digital olarak verilmişse, aynı program sadece veriyi dosyadan okuma olarak değiştirerek kullanılabilir.

Program 6.18.2 veriye trigonometrik dizi (Fourier) yaklaşımı Fourier1.java

```
 import java.io.*;
import javax.swing.*;

// Fourier approximation
class Fourier1
{
public double a[];
public double b[];
public double xj[];
public double zj[];
public double yj[];
public double x0,xn;
public int m; //(2*m+1) data points
public int n; //degree n Fourier function

public void read_data(String s1)
{
double a[][]=Text.readDoubleT(s1);
int mm=a[0].length;
m=mm/2;
x0=a[0][0];
xn=a[0][mm-1];
xj=new double[mm];
```

```
zj=new double[mm];
yj=new double[mm];
for(int k=0;k<=2*m;k++)
{xj[k]=a[0][k];
 yj[k]=a[1][k];
 zj[k]=z_x(xj[k]);
}
}

public double z_x(double x)
{double zz=2.0*Math.PI*(x-x0)/(xn-x0)-Math.PI;
 return zz;
}

public double x_z(double z)
{double xx=(z+Math.PI)/(2.0*Math.PI)*(xn-x0)+x0;
 return xx;
}

public Fourier1(String isim,int ni)
{
read_data(isim);
n=ni;
a=new double[n+1];
b=new double[n+1];
  for(int k=0;k<=n;k++)
  {
  a[k]=0;
  b[k]=0;
  for(int j=0;j<2*m;j++)
    { a[k]+=1.0/m*yj[j]*Math.cos(k*zj[j]);
      b[k]+=1.0/m*yj[j]*Math.sin(k*zj[j]);
    }
  }
}

public  double func(double x)
{
double cc=a[0]/2.0+a[n]*Math.cos(n*z_x(x));
for(int k=1;k<n;k++)
{cc+=a[k]*Math.cos(k*z_x(x))+b[k]*Math.sin(k*z_x(x));}
return cc;
}

public void plot_func(double x0i,double xni,int ni)
{
            double xx[][]=new double[2][ni+1];
            double x1=0;
   for(int i=0;i<=ni;i++)
            { x1=x0i+(xni-x0i)*i/ni;
              xx[0][i]=x1;
              xx[1][i]=func(x1);
```

```
            }
            Plot pp=new Plot(xx[0],xx[1]);
            pp.addData(xj,yj);
            pp.setPlotType(1,32);
   pp.setPlabel("Trigonometric fonksiyon(Fourier) eğri uydurma n="+(a.length-1));
            pp.plot();
}

public double[][] result()
{   double xx[][]=new double[4][2*m];
   for(int i=0;i<2*m;i++)
            {
              xx[0][i]=xj[i];
              xx[1][i]=yj[i];
              xx[2][i]=func(xj[i]);
              xx[3][i]=xx[2][i]-xx[1][i];
            }
            return xx;
}
public String toString()
{
            String s="a : \n"+Matrix.toStringT(a)+"\n";
                 s+="b : \n"+Matrix.toStringT(b)+"\n";
            return s;
}
public static void main(String arg[])
{
            int n=Integer.parseInt(JOptionPane.showInputDialog("trigonometrik yaklaşım fonksiyonu Sn(x) terim
sayısı: n="));
            String isim=JOptionPane.showInputDialog("veri dosyası ismi (a.txt)");
            Fourier1 Pa=new Fourier1(isim,n);
            System.out.println("Trigonometrik seri katsayıları n="+n+"\n"+Pa.toString());
            System.out.println("Trigonometrik polinom n = "+n+" m =
"+Pa.m+"\n"+Matrix.toStringT(Pa.result(),30,10));
   Pa.plot_func(Pa.x0,Pa.xn,100);
}
}
```

Example data a.txt (generated from the function of the previous example)

```
0       0
0.2     0.434002852
0.4     0.898143822
0.6     1.317232349
0.8     1.581957491
1       1.557407725
1.2     1.197957491
1.4     0.645232349
1.6     0.130143822
1.8     -0.141997148
2       0
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" Fourier1
```

Trigonometrik seri katsayıları n=5

a :

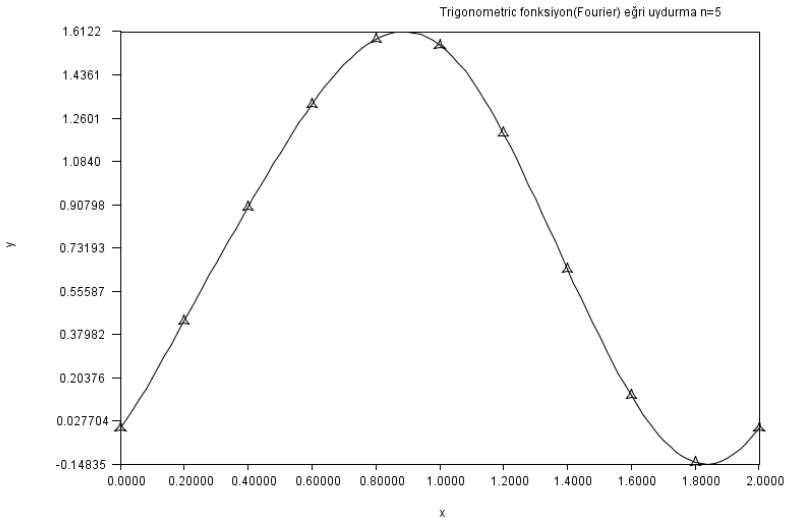
1.524016150600000
0.771769039361246
0.017422790340254
0.006567273038754
-0.000727003140254
0.000735100200000

b :

0.000000000000000
-0.386759045133159
0.047806047069749
-0.013320698468437
0.004310668575164
0.000000000000000

Trigonometrik polinom n = 5 m = 5

0.0000000000	0.0000000000	-0.0003675501	-0.0003675501
0.2000000000	0.4340028520	0.4343704021	0.0003675501
0.4000000000	0.8981438220	0.8977762719	-0.0003675501
0.6000000000	1.3172323490	1.3175998991	0.0003675501
0.8000000000	1.5819574910	1.5815899409	-0.0003675501
1.0000000000	1.5574077250	1.5577752751	0.0003675501
1.2000000000	1.1979574910	1.1975899409	-0.0003675501
1.4000000000	0.6452323490	0.6455998991	0.0003675501
1.6000000000	0.1301438220	0.1297762719	-0.0003675501
1.8000000000	-0.1419971480	-0.1416295979	0.0003675501

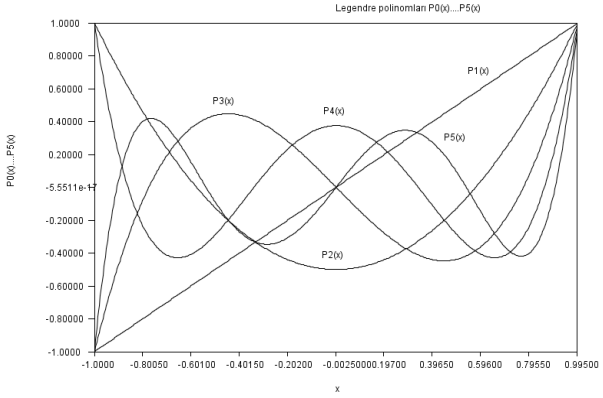


(bolum6_043.jpg,Resimaltı: Trigonometrik fonksiyon eğri uydurma n=5**)**

6.18 Legendre Polinomlarıyla fonksiyon yaklaşımı

Bir üstteki bölümdeki eğri uydurma işlemi temel olarak sinüs ve kosinüz fonsiyonlarının ortagonal fonksiyon olma özelliğinden faydalanıyordu. Diğer ortagonal fonksiyonları kullanarak ta benzer fonksiyon yaklaşımı işlemlerini oluşturabiliriz. Bu bölümde Legendre ortagonal polinomlarını kullanarak fonksiyon yaklaşımını irdeleyelim. Legendre

Polinomlarını $P_k(x) = \frac{(2k-1)}{k} xP_{k-1}(x) - \frac{k-1}{k} P_{k-2}(x), \quad k \geq 2$ denklemiyle tanımlanır. Burada $P_0(x)=1$ ve $P_1(x)=x$ olarak tanımlanmıştır. İlk 6 Legendre polinomunu grafik formda verilmiştir. Legendre polinomları Mathd sınıfında P(n,x) olarak yer almaktadır.



(bolum6_049.jpg,Legendre polinomları)

İlk 10 Legendre polinomunu açık olarak listelersek:

n	$P_n(x)$ -1<= x <=1
0	1
1	x
2	$(3x^2-1)/2$
3	$(5x^3-3x)/2$
4	$(35x^4-30x^2+3)/8$
5	$(63x^5-70x^3+15x)/8$
6	$(231x^6-315x^4+105x^2-5)/16$
7	$(429x^7-693x^5+315x^3-35x)/16$
8	$(6435x^8-12012x^6+6930x^4-1260x^2+35)/128$
9	$(12155x^9-25740x^7+18018x^5-4620x^3+315x)/128$
10	$(46189x^{10}-109395x^8+90090x^6-30030x^4+3465x^2-63)/256$

Eğer f(x) fonksiyonu yerine bir Legendre fonksiyonu serisi yazmak istersek [-1,1] aralığında $f(x) = \sum_{k=0}^{m} b_k P_k(x) + hata$ şeklinde yazabiliriz. Legendre polinomu temel olarak [-1,1] bölgesini kapsadığından bu bölgenin dışında [a,b] bölgesi için fonksiyon yaklaşımı tanımlamak istersek, değişken dönüşümü uygulayabiliriz.

$$z = \left(\frac{b-a}{2}\right)x + \left(\frac{b+a}{2}\right) = \alpha x + \beta$$

$\alpha = \left(\frac{b-a}{2}\right) \quad \beta = \left(\frac{b+a}{2}\right)$ bu durum için fonksiyonumuz

$f(z) = \sum_{k=0}^{m} b_k q_k(z) + hata$ şekinde ifade edilebilir. q_k dönüştürülmüş Legendre fonksiyonu olup

$q_k(z) = q_k(\alpha x + \beta) = P_k(x)$ şeklinde yazılabilir.

Ortogonal polinomların özelliğinden dolayı:

$(P_n(x)P_j(x)) = \int_{-1}^{1} P_n(x)P_j(x)dx$ integrali

$$(P_n(x)P_j(x)) = \begin{Bmatrix} 0 \; n \neq j \; durumunda \\ \frac{2}{2n+1} \; n = j \; durumunda \end{Bmatrix}$$ değerini alır. Bu bizim yaklaşım serisinin katsayılarını

$b_k = \frac{2k+1}{b-a}\int_a^b f(z)q_k(z)dz$ formülünden hesaplamamızı sağlar. İntegral sayısal olarak alınabilir. Sayısal integrallerin detayları için ilgili bölümü inceleyiniz. Örnek fonksiyon olarak Trigonometrik yaklaşımda incelediğimiz f(x)=|x| fonksiyonunu alalım.

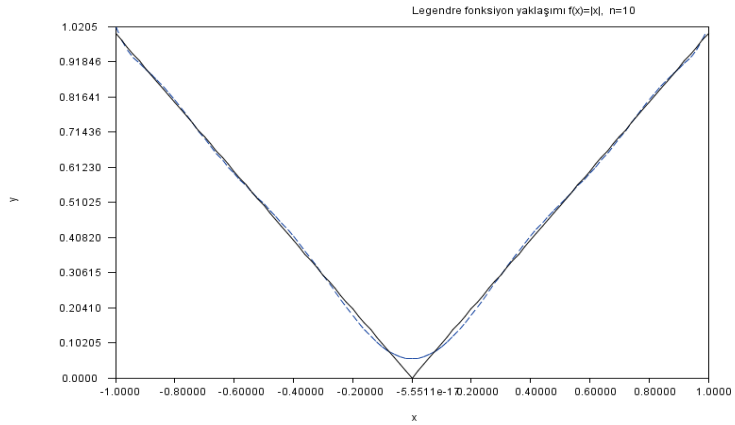
Program 6.19.1 verilen bir fonksiyona Legendre Fonksiyonu yaklaşımı legendre1.java

```
 class ff extends f_x
{
public double func(double x)
{return Math.abs(x);}
}

public class legendre1
{
public double alfa,beta,xl,xh;
public double x[];
public double z[];
public double y[];
public double b[]; //katsayılar matrisi;
public int m;   //eğri uydurma derecesi

public static double P(int k, double x)
{
 // Legendre polinomu  -1<=x<=1
if(k==0) return 1.0;
else if(k==1) return x;
else
{return ((2.0*k-1.0)*P((k-1),x)*x-P((k-2),x)*(double)(k-1))/(double)k;}
// end legendre polynomials
}

public  double q(int k, double z)
{  // dönüştürülmüş Legendre polinomu xl<= z <=xh
              double x=(z-beta)/alfa;
   return P(k,x);
}
public  double fak(f_x f,int k,double z)
    {   //integral fonksiyonu
                 double a=(2.0*k+1)/(xh-xl)*f.func(z)*q(k,z);
                 return a;
               }

public legendre1(f_x f,double xli,double xhi,int mi)
{setdata(f,xli,xhi,mi);}

public void setdata(f_x f,double xli,double xhi,int mi)
{  //z  xl, xh
     m=mi;
     xl=xli;
     xh=xhi;
     alfa=(xh-xl)/2.0;
     beta=(xh+xl)/2.0;
     b=new double[m+1];
     for(int j=0;j<=m;j++)
                {b[j]=integral(f,xl,xh,30,j);}
                System.out.println("b=\n"+Matrix.toStringT(b));
}

public  double funcLegendre(double z)
{
   double x=(z-beta)/alfa;
   double ff=0;
   double pk=0;
```

```
    for(int j=0;j<=m;j++)
      {   pk=P(j,x);
                  ff+=b[j]*pk;
                }
   return ff;
}

public static double[][] gauss_legendre_katsayilari(double x1,double x2,int n)
{
            // Gauss-Legendre integral katsayılarını hesaplar
            // Legendre polinomu kökleri
            double EPS=3.0e-15;
            int m,j,i;
            double z1,z,xm,xl,pp,p3,p2,p1;
   double a[][]=new double[2][n];//a[0][i]=x[i]   a[1][i]=w[i]
            m=(n+1)/2;
            xm=0.5*(x2+x1);
            xl=0.5*(x2-x1);
            for (i=1;i<=m;i++)  {
                        z=Math.cos(Math.PI*((i-0.25)/(n+0.5)));
                        do {
                                    p1=1.0;
                                    p2=0.0;
                                    for (j=1;j<=n;j++) {
                                                p3=p2;
                                                p2=p1;
                                                p1=((2.0*j-1.0)*z*p2-(j-1.0)*p3)/j;
                                    }
                                    pp=n*(z*p1-p2)/(z*z-1.0);
                                    z1=z;
                                    z=z1-p1/pp;
                        } while (Math.abs(z-z1) > EPS);
                        a[0][i-1]=xm-xl*z;
                        a[0][n-i]=xm+xl*z;
                        a[1][i-1]=2.0*xl/((1.0-z*z)*pp*pp);
                        a[1][n-i]=a[1][i-1];
                                    }
return a;
}

public  double integral(f_x f,double xl,double xh,int n,int k)
{
// n : integral katsayısı
// Bu program önce [x1,x2] bandı için Gauss Legender katsayılarını hesaplar
// sonra da integral hesabını yapar
double a[][]=new double[2][n];
a=gauss_legendre_katsayilari(xl,xh,n);
//System.out.println("katsayılar matrisi : \n"+Matrix.toStringT(a));
double z=0;
for(int i=0;i<n;i++)
{ z+=a[1][i]*fak(f,k,a[0][i]);}
return z;
}

  public static void main(String args[])
  {
   ff f=new ff();
   int n=200;//veri sayısı
   legendre1 l=new legendre1(f,-1,1.0,30);
   double z1[]=new double[n+1];
   double y[]=new double[n+1];
   double y1[]=new double[n+1];
   for(int i=0;i<=n;i++)
   {  z1[i]=l.xl+(l.xh-l.xl)/(double)n*i;
      y[i]=f.func(z1[i]);
      y1[i]=l.funcLegendre(z1[i]);
```

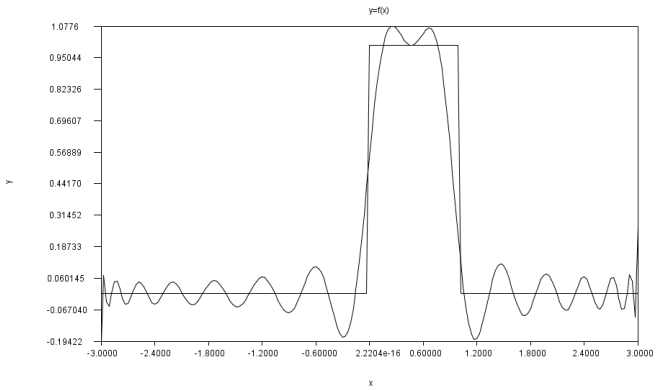
```
   }
   Plot p1=new Plot(z1,y1);
   p1.addData(z1,y);
   p1.plot();
  }
}
```

```
---------- Capture Output ----------
> "c:\java\bin\javaw.exe" legendre
b=
       0.500000000000000
       0.000000000000000
       0.625000000000000
       0.000000000000000
      -0.187500000000000
       0.000000000000000
       0.101562500000000
       0.000000000000000
      -0.066406250000000
       0.000000000000000
       0.047851562500000
> Terminated with exit code 0.
```



(bolum6_050.jpg,Legendre polinom fonksiyon yaklaşımı)

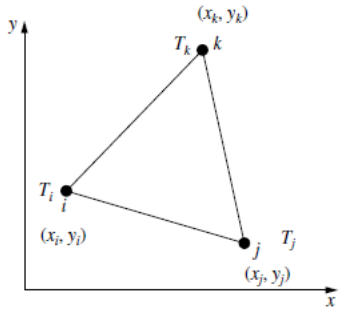
Aynı programı $f(x)=\begin{Bmatrix}0\\1\\0\end{Bmatrix}$ $\begin{matrix}x<0\ \ için\\0<=x<=1\ \ için\\x>1\ \ için\end{matrix}$ fonksiyonuna uygularsak:



6.19 Sonlu Elemanlar(finite element) metodu ile yüzey uydurma

Simpleks elemanları Nelder-Mead metodunda optimizasyon olarak kullanılmıştı. Burada da sonlu elemanlar adını verdiğimiz simpleks elemanlarını kullanarak bir yüzeye veri uydurmanın nasıl yapılacağını vereceğiz. Bir yüzeye uydurulabilecek en basit şekil lineer üçgen şeklidir. Bu eleman popüler olarak sonlu elemanlar paketlerinde de kullanılmaktadır. Üçgen lineer elemanı şu şekilde tanımlayabiliriz:

$$T(x,y)=\alpha_1+\alpha_2 x+\alpha_3 y$$



(bolum6_044.jpg,Resimaltı: Lineer üçgen yüzey tanımı)

Denklem lineer bir denklem olduğundan , denklem katsayılarını yerine koyma metodu ile kolayca bulabiliriz.

$T_i=\alpha_1+\alpha_2 x_i+\alpha_3 y_i$

$T_j=\alpha_1+\alpha_2 x_j+\alpha_3 y_j$

$T_k=\alpha_1+\alpha_2 x_k+\alpha_3 y_k$

Denklem sistemi çözüldüğünde:

$$\alpha_1=\frac{1}{2A}\left[(x_j y_k-x_k y_j)T_i+(x_k y_i-x_i y_k)T_j+(x_i y_j-x_j y_i)T_k\right]$$

$$\alpha_2=\frac{1}{2A}\left[(y_j-y_k)T_i+(y_k-y_i)T_j+(y_i-y_j)T_k\right]$$

$$\alpha_3=\frac{1}{2A}\left[(x_k-x_j)T_i+(x_i-x_k)T_j+(x_j-x_i)T_k\right]$$

Olarak bulunur. Burada '*A*' üçgenin alanıdır ve

$$2A=\det\begin{bmatrix}1 & x_i & y_i\\ 1 & x_j & y_j\\ 1 & x_k & y_k\end{bmatrix}=(x_i y_j-x_j y_i)+(x_k y_i-x_i y_k)+(x_j y_k-x_k y_j)$$

Denkleminden bulunabilir. α_1, α_2 ve α_3 değerlerini yazılımını değiştirirsek, alttaki denklem setini elde ederiz.

$$T=N_i T_i+N_j T_j+N_k T_k$$

$$N_i=\frac{1}{2A}\left[a_i+b_i x+c_i y\right]$$

$$N_j=\frac{1}{2A}\left[a_j+b_j x+c_j y\right]$$

$$N_k=\frac{1}{2A}\left[a_k+b_k x+c_k y\right]$$

$$a_i=(x_j y_k-x_k y_j) \qquad a_j=(x_k y_i-x_i y_k) \qquad a_k=(x_i y_j-x_j y_i)$$

$$b_i=(y_j-y_k) \qquad b_j=(y_k-y_i) \qquad b_k=(y_i-y_j)$$

$$c_i=(x_k-x_j) \qquad c_j=(x_i-x_k) \qquad c_k=(x_j-x_i)$$

Bu denklemleri düğüm i için (x_i,y_i) koordinatlarında çözdüğümüzde

$N_i(x_i,y_i)=1$ ve benzer olarak $N_i(x_j,y_j)=0$, $N_i(x_k,y_k)=0$ değerlerini elde ederiz.

Üçgenin içindeki herhangi bir nokta için de
$N_i+N_j+N_k=1$ olur. Lineer interpolasyon göreceli olarak basit bir yüzey interpolasyon sistemi oluşturur.

Program 6.19.1 lineer üçgen sonlu farklar simpleks elemanı

```
public class triangular_element1
{
public double a[],b[],c[];
public double dt;
public int P;
public double A;//Area
public double l[];
public double x[],y[];   //coordinates of data points m
public double xc,yc;     //center point m
public double N[],B[][]; // Shape factor and derivative
public double Ti[];      // Temperature nodes
public int n_point[];    // point numbers
public int n_element;    // element number

public triangular_element1(int n_elementi,int n_pointi[],double xi[],double yi[])
{dt=1.0;
 P=1;
 x=new double[3];
 y=new double[3];
 Ti=new double[3];
 l=new double[3];//0:lij,1:ljk,2:lik
 xc=0;yc=0;
 n_point=new int[3];
 n_element=n_elementi;
 for(int i=0;i<3;i++)
 { n_point[i]=n_pointi[i];
  x[i]=xi[i];y[i]=yi[i];
  xc+=x[i];yc+=y[i];
 }
 xc/=3.0;
 yc/=3.0;
 a=new double[3];
 b=new double[3];
 c=new double[3];
 N=new double[3];
 B=new double[2][3];
 a[0]=x[1]*y[2]-x[2]*y[1];
 a[1]=x[2]*y[0]-x[0]*y[2];
 a[2]=x[0]*y[1]-x[1]*y[0];
 b[0]=y[1]-y[2];
 b[1]=y[2]-y[0];
 b[2]=y[0]-y[1];
 c[0]=x[2]-x[1];
 c[1]=x[0]-x[2];
 c[2]=x[1]-x[0];
 A=0.5*((x[0]*y[1]-x[1]*y[0])+(x[2]*y[0]-x[0]*y[2])+(x[1]*y[2]-x[2]*y[1]));
 for(int i=0;i<3;i++)
 { B[0][i]=1.0/(2.0*A)*b[i];
   B[1][i]=1.0/(2.0*A)*c[i];
 }
 l[0]=Math.sqrt(b[2]*b[2]+c[2]*c[2]);
 l[1]=Math.sqrt(b[0]*b[0]+c[0]*c[0]);
 l[2]=Math.sqrt(b[1]*b[1]+c[1]*c[1]);
}
public triangular_element1(int n_elementi,int n_pointi[],double xi[],double yi[],double dti,int Pi)
{
 x=new double[3];
 y=new double[3];
 Ti=new double[3];
 l=new double[3];//0:lij,1:ljk,2:lik

 n_element=n_elementi;
 for(int i=0;i<3;i++)
 {n_point[i]=n_pointi[i];
  x[i]=xi[i];y[i]=yi[i];
  xc+=x[i];yc+=y[i];
 }
 xc/=3.0;
 yc/=3.0;
 a=new double[3];
```

```
 b=new double[3];
 c=new double[3];
 N=new double[3];
 B=new double[2][3];
 a[0]=x[1]*y[2]-x[2]*y[1];
 a[1]=x[2]*y[0]-x[0]*y[2];
 a[2]=x[0]*y[1]-x[1]*y[0];
 b[0]=y[1]-y[2];
 b[1]=y[2]-y[0];
 b[2]=y[0]-y[1];
 c[0]=x[2]-x[1];
 c[1]=x[0]-x[2];
 c[2]=x[1]-x[0];
 A=0.5*((x[0]*y[1]-x[1]*y[0])+(x[2]*y[0]-x[0]*y[2])+(x[1]*y[2]-x[2]*y[1]));
 for(int i=0;i<3;i++)
 { B[0][i]=1.0/(2.0*A)*b[i];
   B[1][i]=1.0/(2.0*A)*c[i];
 }
 l[0]=Math.sqrt(b[2]*b[2]+c[2]*c[2]);
 l[1]=Math.sqrt(b[0]*b[0]+c[0]*c[0]);
 l[2]=Math.sqrt(b[1]*b[1]+c[1]*c[1]);
}

public int setI(int cond,double L0,double L,double W0,double W)
{
   int i=100;
                if(cond==0) //x=0
                {if(x[0]==L0 && x[1]==L0) i=0;
                 else if(x[1]==L0 && x[2]==L0) i=1;
                 else if(x[0]==L0 && x[2]==L0) i=2;
                 else i=-1;
                }
                else if(cond==1) //x=L
                {if(x[0]==L && x[1]==L) i=0;
                 else if(x[1]==L && x[2]==L) i=1;
                 else if(x[0]==L && x[2]==L) i=2;
                 else i=-1;
                }
                else if(cond==2) //y=0
                {if(y[0]==W0 && y[1]==W0) i=0;
                 else if(y[1]==W0 && y[2]==W0) i=1;
                 else if(y[0]==W0 && y[2]==W0) i=2;
                 else i=-1;
                }
                else if(cond==3) //y=W
                {if(y[0]==W && y[1]==W) i=0;
                 else if(y[1]==W && y[2]==W) i=1;
                 else if(y[0]==W && y[2]==W) i=2;
                 else i=-1;
                }
                return i;
}


public void setN(double x,double y)
{for(int i=0;i<3;i++)
  {N[i]=1.0/(2.0*A)*(a[i]+b[i]*x+c[i]*y);}
}
public void setN(double z[])
{setN(z[0],z[1]);}

public void setT(double TT[])
{for(int i=0;i<3;i++)
  {Ti[i]=TT[i];}
}

public double[] getT()
{ return Ti;}

public double T(double x,double y)
{ //interpolation
                double TT=0;
 setN(x,y);
 for(int i=0;i<3;i++)
  {TT+=N[i]*Ti[i];}
   return TT;
```

```
}

public double T(double T[],double x,double y)
{ //set grid point temperatures and get interpolated value
                setT(T);
  setN(x,y);
  double TT=0;
 for(int i=0;i<3;i++)
  {TT+=N[i]*Ti[i];}
  return TT;
}
public double T(double T[],double z[])
{ return T(T,z[0],z[1]);}

}
```

Program 6.19.2 lineer üçgen sonlu elemanlar gridi(eğri uydurma programı)

```
 import javax.swing.*;
import org.math.plot.*;
import static java.lang.Math.*;
import static org.math.array.DoubleArray.*;
import visad.*;
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class triangular_grid1
{
public triangular_element1 te[];
public setElement se[];
public int N,M,K,K1,K2,Nm1,Mm1;
public double x[][];
public double y[][];
public double T[][];
public double Ti[];
public double xi[];
public double yi[];
public boolean bi[];
public double dx,dy;
public int Bx0[],BxM[],By0[],ByN[];
public double L,W;
public double min(double xi[][])
{ double min=1e200;
  for(int i=0;i<xi.length;i++)
   {for(int j=0;j<xi[0].length;j++)
     {if(xi[i][j]<min) min=xi[i][j];}
 }
 return min;
}
public double max(double xi[][])
{double max=-1e200;
 for(int i=0;i<xi.length;i++)
 {for(int j=0;j<xi[0].length;j++)
   {if(xi[i][j]<max) max=xi[i][j];}
 }
 return max;
}
public triangular_grid1(double xi[][],double yi[][],double zi[][])
{x=xi;
 y=yi;
 T=zi;
 N=x.length-1;
 M=x[0].length-1;
 L=max(xi);
 W=max(yi);
 setElements();
}

public void generateTriangularGrid(double Li, double Wi,int Ni, int Mi,double zi[][])
{int i,j;
 N=Ni;M=Mi;
 L=Li;
 W=Wi;
```

```
 x=new double[N+1][M+1];
 y=new double[N+1][M+1];
 T=zi;
 setElements();
}

public void setElements()
{ int i,j;
 dx=L/M;
 dy=W/N;
 K=M*N;
 K1=K*2;
 K2=K1+1;
 //grid points
 xi=new double[K2];
 yi=new double[K2];
 Ti=new double[K2];
 bi=new boolean[K2];
 se=new setElement[K2];
 for(i=0;i<K2;i++) {se[i]=new setElement();}
 te=new triangular_element1[K1];
 int k1=-1,k=0,n=0,m=0;
for(n=0;n<=N;n++)
{ for(m=0;m<=M;m++)
  {k=(N+1)*n+m;
   x[n][m]=m*dx;y[n][m]=n*dy;
   xi[k]=x[n][m];
   yi[k]=y[n][m];
   bi[k]=false;
   Ti[k]=T[n][m];
  }
}

for(n=0;n<N;n++)
{for(m=0;m<M;m++)
  { k=(N+1)*n+m;
                int kp1=(N+1)*(n+1)+m;
                int kp1p1= kp1+1;
                double xi1[]={x[n][m],x[n][m+1],x[n+1][m]};
                double yi1[]={y[n][m],y[n][m+1],y[n+1][m]};
    double zi1[]={T[n][m],T[n][m+1],T[n+1][m]};
                k1++;
                int n_point1[]={k,(k+1),kp1};
                se[k].add(k1,0);
                se[k+1].add(k1,1);
                se[kp1].add(k1,2);
                te[k1]=new triangular_element1(k1,n_point1,xi1,yi1);
                te[k1].setT(zi1);
                double xi2[]={x[n][m+1],x[n+1][m+1],x[n+1][m]};
                double yi2[]={y[n][m+1],y[n+1][m+1],y[n+1][m]};
                double zi2[]={T[n][m+1],T[n+1][m+1],T[n+1][m]};
                k1++;
                int n_point2[]={k+1,kp1p1,kp1};
    se[k+1].add(k1,0);
                se[kp1p1].add(k1,1);
                se[kp1].add(k1,2);
                te[k1]=new triangular_element1(k1,n_point2,xi2,yi2);
                te[k1].setT(zi2);
  }
}
}
public triangular_grid1(double Li, double Wi,int Ni, int Mi,double zz[][])
{generateTriangularGrid(Li,Wi,Ni,Mi,zz);}
public int find_element(double z[])
{ return find_element(z[0],z[1]);}
public int find_element(double x,double y)
{double R1=Math.sqrt(x*x+y*y);
 int kmin=0;
 double min=1.0e50;
 for(int k=0;k<K1;k++)
 {   te[k].setN(x,y);
     double r0=Math.sqrt((te[k].x[0]-x)*(te[k].x[0]-x)+(te[k].y[0]-y)*(te[k].y[0]-y));
     double r1=Math.sqrt((te[k].x[1]-x)*(te[k].x[1]-x)+(te[k].y[1]-y)*(te[k].y[1]-y));
     double r2=Math.sqrt((te[k].x[2]-x)*(te[k].x[2]-x)+(te[k].y[2]-y)*(te[k].y[2]-y));
     double dr=Math.sqrt(r0*r0+r1*r1+r2*r2);
                boolean b1=(x>=te[k].x[0]) && (x<=te[k].x[2]);
```

```
                boolean b2=(y>=te[k].y[0]) && (y<=te[k].y[1]);
                boolean b= b1 && b2;
    if(dr<min) {min=dr;kmin=k;}
 }
 return kmin;
}

public void ContourPlot()
{
ContourPlot pp=new ContourPlot( x,y,T,16,true,"Finite element","x","y");
pp.plot();
}
public void Plot3D(double L0,double L,double W0,double W) throws RemoteException, VisADException
{
Plot3D.plot3D_CD1(T,L0,L,W0,W,200,"x","y","T degree C");
}
public void GridPlot(double L0,double L,double W0,double W)
{Plot3DPanel plot = new Plot3DPanel("SOUTH");
 double dx=(L-L0)/M;
 double dy=(W-W0)/N;
 double[] xx = increment(L0, dx,L); // x = 0.0:0.1:1.0
 double[] yy = increment(W0,dy,W);
 plot.addGridPlot("Finite element Grid", xx, yy, T);
 JFrame frame = new JFrame("a plot panel");
 frame.setSize(600, 600);
 frame.setContentPane(plot);
 frame.setVisible(true);
}
public double getT(double x,double y)
{double z[]={x,y};
 int n=find_element(z);
 double T1=te[n].T(x,y);
 return T1;
}

public static void main(String arg[]) throws RemoteException, VisADException
{double L=1.0;
 double W=1.0;
 int N=2,M=2;
 double TT[][]={{80.0,90.0,100.0},{90.0,100.0,110.0},{100.0,110.0,120.0}};
 double hi=20.0;
 triangular_grid1 tg=new triangular_grid1(L,W,M,N,TT);
 double z[]={0.65,0.55};
 int n=tg.find_element(z);
 System.out.println("element = "+n);
 double T=tg.getT(z[0],z[1]);
 System.out.println("T = "+T);
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\java.exe" triangular_grid1
element = 6
T = 104.00000000000001

> Terminated with exit code 0.
```

6.20 Problemler

PROBLEM 1

Aşağıdaki verilere düz bir doğru uydurmak için en küçük kareler regresyonunu kullanınız.

X	2	3	4	7	8	9	5	5
Y	9	6	5	10	9	11	2	3

PROBLEM 2

R22 soğutucu akışkan için sıvı vizkozite değerleri gerekmektedir. Bu değerler alttaki tabloda tanımlanmıştır. Bu veriyi bilgisayara aktarabileceğimiz bir egri uydurunuz, eğrinin hata miktarını hesaplayınız, daha iyi bir eğri oluşturabilmak için (hata miktarını azaltmak için) kullanabileceğiniz diğer yöntemleri tartışınız, mümkünse uygulayınız.

-101.1111111	8.6519780084E-04	-28.88888889	3.0093836551E-04	32.22222222	1.5253606713E-04
-95.55555556	7.7466826506E-04	-26.11111111	2.9143069737E-04	35	1.4798892150E-04
-90	6.9943367368E-04	-23.33333333	2.8233640610E-04	37.77777778	1.4302839899E-04
-84.44444444	6.3536025795E-04	-20.55555556	2.7365549171E-04	40.55555556	1.3848125336E-04
-78.88888889	5.8079451036E-04	-17.77777778	2.6538795420E-04	43.33333333	1.3393410773E-04
-73.33333333	5.3325616965E-04	-15	2.5712041668E-04	46.11111111	1.2938696209E-04
-70.55555556	5.1176057211E-04	-12.22222222	2.4926625605E-04	48.88888889	1.2483981646E-04
-67.77777778	4.9150510520E-04	-9.444444444	2.4182547228E-04	51.66666667	1.2070604770E-04
-65	4.7290314580E-04	-6.666666667	2.3479806540E-04	54.44444444	1.1615890207E-04
-62.22222222	4.5512794014E-04	-3.888888889	2.2777065851E-04	57.22222222	1.1202513331E-04
-59.44444444	4.3817948824E-04	-1.111111111	2.2074325162E-04	60	1.0747798768E-04
-56.66666667	4.2205779009E-04	1.666666667	2.1412922161E-04	62.77777778	1.0334421892E-04
-53.88888889	4.0717622256E-04	4.444444444	2.0792856848E-04	65.55555556	9.9210450167E-05
-51.11111111	3.9312140879E-04	7.222222222	2.0172791534E-04	68.33333333	9.5076681410E-05
-48.33333333	3.7947997189E-04	10	1.9552726220E-04	71.11111111	9.0529535778E-05
-45.55555556	3.6666528874E-04	12.77777778	1.8973998595E-04	73.88888889	8.6395767021E-05
-42.77777778	3.5426398247E-04	15.55555556	1.8395270969E-04	76.66666667	8.1848621388E-05
-40.81111111	3.4599644496E-04	18.33333333	1.7857881030E-04	79.44444444	7.7714852631E-05
-40	3.4268942995E-04	21.11111111	1.7320491092E-04	82.22222222	7.2754330123E-05
-37.22222222	3.3152825431E-04	23.88888889	1.6783101153E-04	85	6.8207184490E-05
-34.44444444	3.2078045554E-04	26.66666667	1.6287048902E-04	87.77777778	6.2833285106E-05
-31.66666667	3.1044603365E-04	29.44444444	1.5749658964E-04	90.55555556	5.7046008846E-05
				93.33333333	5.0018601959E-05

PROBLEM 3

R134a soğutucu akışkan için doymuş akışkan sıcaklık-basınç değerleri gerekmektedir. Bu değerler tabloda tanımlanmıştır. Bu veriyi bilgisayara aktarabileceğimiz bir egri uydurunuz, eğrinin hata miktarını hesaplayınız, daha iyi bir eğri oluşturabilmak için (hata miktarını azaltmak için) kullanabileceğiniz diğer yöntemleri tartışınız, kaynakları araştırarak doyma eğrilerinde kullanılan en uygun eğri denklemini bulunuz, mümkünse bu denklemi kullanarak eğri uydurunuz.

-99	0.01	-49	2.06	0	32.73	50	212.61	100	786.93	150	2100.93
-98	0.01	-48	2.21	1	34.24	51	219.24	101	804.64	151	2137.75
-97	0.01	-47	2.37	2	35.81	52	226.02	102	822.65	152	2175.06
-96	0.01	-46	2.54	3	37.43	53	232.96	103	840.96	153	2212.87
-95	0.01	-45	2.72	4	39.12	54	240.07	104	859.56	154	2251.19
-94	0.02	-44	2.91	5	40.86	55	247.33	105	878.47	155	2290.01
-93	0.02	-43	3.11	6	42.67	56	254.77	106	897.69	156	2329.35
-92	0.02	-42	3.32	7	44.54	57	262.38	107	917.22	157	2369.21
-91	0.03	-41	3.54	8	46.48	58	270.15	108	937.07	158	2409.61
-90	0.03	-40	3.78	9	48.49	59	278.1	109	957.23	159	2450.54
-89	0.04	-39	4.03	10	50.57	60	286.23	110	977.71	160	2492.01
-88	0.04	-38	4.29	11	52.72	61	294.54	111	998.52	161	2534.04
-87	0.05	-37	4.57	12	54.94	62	303.03	112	1019.66	162	2576.63
-86	0.06	-36	4.86	13	57.23	63	311.7	113	1041.12	163	2619.8
-85	0.07	-35	5.17	14	59.61	64	320.57	114	1062.92	164	2663.54
-84	0.08	-34	5.5	15	62.06	65	329.62	115	1085.05	165	2707.88
-83	0.09	-33	5.84	16	64.59	66	338.86	116	1107.52	166	2752.81
-82	0.1	-32	6.2	17	67.2	67	348.3	117	1130.35	167	2798.36
-81	0.11	-31	6.58	18	69.9	68	357.93	118	1153.52	168	2844.54
-80	0.13	-30	6.98	19	72.68	69	367.77	119	1177.04	169	2891.36
-79	0.14	-29	7.4	20	75.55	70	377.81	120	1200.91	170	2938.83
-78	0.16	-28	7.84	21	78.52	71	388.05	121	1225.15	171	2986.97
-77	0.18	-27	8.3	22	81.57	72	398.5	122	1249.75	172	3035.81
-76	0.2	-26	8.78	23	84.71	73	409.17	123	1274.71	173	3085.29
-75	0.23	-25	9.29	24	87.95	74	420.04	124	1300.04	174	3135.59
-74	0.25	-24	9.82	25	91.29	75	431.13	125	1325.74	175	3186.64
-73	0.28	-23	10.38	26	94.73	76	442.45	126	1351.82	176	3238.47
-72	0.31	-22	10.96	27	98.27	77	453.98	127	1378.28	177	3291.12
-71	0.34	-21	11.58	28	101.92	78	465.74	128	1405.13	178	3344.62

-70	0.37	-20	12.22	29	105.66	79	477.72	129	1432.36	179	3399.02
-69	0.41	-19	12.88	30	109.52	80	489.93	130	1459.98	180	3454.38
-68	0.45	-18	13.58	31	113.49	81	502.38	131	1488	181	3510.78
-67	0.49	-17	14.32	32	117.57	82	515.06	132	1516.41	182	3568.29
-66	0.54	-16	15.08	33	121.76	83	527.98	133	1545.23	183	3627.05
-65	0.59	-15	15.88	34	126.07	84	541.15	134	1574.46		
-64	0.64	-14	16.71	35	130.49	85	554.55	135	1604.1		
-63	0.7	-13	17.58	36	135.04	86	568.21	136	1634.15		
-62	0.76	-12	18.48	37	139.71	87	582.11	137	1664.62		
-61	0.83	-11	19.43	38	144.51	88	596.27	138	1695.52		
-60	0.9	-10	20.41	39	149.43	89	610.68	139	1726.84		
-59	0.97	-9	21.44	40	154.48	90	625.35	140	1758.6		
-58	1.05	-8	22.5	41	159.66	91	640.28	141	1790.79		
-57	1.14	-7	23.61	42	164.98	92	655.47	142	1823.42		
-56	1.23	-6	24.77	43	170.43	93	670.94	143	1856.5		
-55	1.33	-5	25.97	44	176.03	94	686.67	144	1890.03		
-54	1.43	-4	27.22	45	181.76	95	702.68	145	1924.01		
-53	1.54	-3	28.52	46	187.63	96	718.96	146	1958.46		
-52	1.66	-2	29.87	47	193.65	97	735.53	147	1993.37		
-51	1.79	-1	31.27	48	199.82	98	752.38	148	2028.74		
-50	1.92			49	206.14	99	769.51	149	2064.6		

PROBLEM 4

Tablo P4 daki x y değerleri deney verisi olarak bulunmuş olsun. Bu değerlere bir eğri uydurmaya çalıştığımızda klasik eğri uydurma metodlarının cevap vermediğini göreceğiz. Çünki aynı x noktasına karşı gelen 1 den fazla y verisi bulunmaktadır. Bu durumda bir eğri uydurmaki istediğimizde sete suni bir ek parametre ilave edebiliriz. Örneğin tablodaki t parametresini ilave edelim. (t,x) ve (t,y) tekrarlanmayan iki eğri seti olduğundan bu iki sete eğri uydurmamız mümkün olabilecektir. Bunu yaptıktan sonra değerlendirme aşamasında seti tekrar (x,y) setine dönüştürebiliriz. Bu veriye eğri uyduracak bir program hazırlayınız.

Tablo P.4 Girdi verisi

İ	0	1	2	3	4
T	0	1	2	3	4
X	-1	0	1	0	1
Y	0	1	0.5	0	-1

PROBLEM 5

X	y
0	-2.20
0.1	-1.90
0.2	0
0.3	-0.1
0.4	-0.01
0.5	-0.3
0.6	-0.2
0.7	-0.09
0.8	0.4
0.9	-0.1
1.0	-1.1
1.1	-1.3
1.2	-2
1.3	-1.8
1.4	-0.02
1.5	1

Verisine

a) en küçük kareler yöntemiyle
b) interpolasyon yöntemlerini kullanarak eğri uydurunuz. Uydurduğunuz eğriyi çizerek kontrol ediniz.

PROBLEM 6

X	y
0	0
1	1
2	4
3	8

Verisine kübik şerit interpolasyonu uydurunuz.

PROBLEM 7

X	y
0	0
1	1
2	0
1	-1

Verisine kübik şerit interpolasyonu uydurunuz.

PROBLEM 8

Etilen glikol-su karışımı(antifriz) yoğunluk(kg/m^3) değerleri tabloda verilmiştir (tablo etilen_gylicol_ro.xls) olarak da mevcuttur. . Formül olarak

$\rho(x,t)= a_0+a_1*x+a_2*t+a_3*x^2+a_4*t^2+a_5*xt+a_6*x^3+a_7*t^3+a_8*x^2t$
$+a_9*xt^2+a_{10}*x^4+a_{11}*t^4+a_{12}*x^2t^2+a_{13}x^3t+a_{14}*xt^3$

kullanılacaktır. Buradaki formülden de görüldüğü gibi bu özellikler x ve t nin fonksiyonudurlar ve veri de bu şekilde verilmiştir. genel EKK yöntemi kullanarak yoğunluk fonksiyonunun katsayılarını hesaplayınız. Eğrinin uyum verimi yeterlimidir? Değilse formül ile oynıyarak daha iyi uyan bir formülü deneyiniz.

	x %hacim oranı etilen glikol konsantrasyonu									
t sıcaklık C	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
-35							1090	1105	1119	1132
-30							1089	1104	1117	1131
-25						1088	1102	1116	1129	1142
-20					1072	1087	1101	1115	1128	1140
-15					1071	1086	1100	1113	1126	1138
-10				1054	1070	1084	1098	1111	1124	1136
-5			1037	1053	1068	1083	1096	1109	1122	1134
0	999.8	1019	1036	1052	1067	1081	1095	1108	1120	1132
5	999.9	1018	1034	1050	1065	1079	1093	1105	1118	1129
10	999.7	1016	1033	1049	1063	1077	1091	1103	1115	1127
15	999.1	1015	1031	1047	1062	1075	1089	1101	1113	1124
20	998.2	1013	1030	1045	1060	1073	1086	1098	1110	1121
25	997	1012	1028	1043	1058	1071	1084	1096	1107	1118
30	995.6	1010	1026	1041	1055	1069	1081	1093	1105	1115
35	994	1008	1024	1039	1053	1066	1079	1090	1102	1112
40	992.2	1006	1022	1037	1051	1064	1076	1088	1098	1109
45	990.2	1004	1020	1034	1048	1061	1073	1084	1095	1106
50	988	1002	1017	1032	1045	1058	1070	1081	1092	1102
55	985.7	999.2	1015	1029	1043	1055	1067	1078	1088	1098
60	983.2	996.7	1012	1026	1040	1052	1064	1075	1085	1095
65	980.5	994.1	1009	1023	1037	1049	1060	1071	1081	1091
70	977.7	991.4	1006	1020	1033	1045	1057	1067	1077	1087

75	974.8	988.5	1003	1017	1030	1042	1053	1064	1073	1083
80	971.8	985.5	1000	1014	1027	1038	1049	1060	1069	1078
85	968.6	982.4	997	1011	1023	1035	1046	1056	1065	1074
90	965.3	979.2	993.6	1007	1019	1031	1042	1052	1061	1070
95	961.9	975.8	990.1	1003	1016	1027	1038	1047	1056	1065
100	958.3	972.3	986.5	999.7	1012	1023	1033	1043	1052	1060
105	954.7	968.7	982.8	995.8	1008	1019	1029	1038	1047	1055
110	950.9	965	978.9	991.8	1004	1014	1025	1034	1042	1050
115	947.1	961.2	974.9	987.7	999.3	1010	1020	1029	1038	1045
120	943.1	957.2	970.8	983.4	994.9	1005	1015	1024	1032	1040
125	939	953.1	966.6	979.1	990.4	1001	1010	1019	1027	1035

PROBLEM 9

Tabloda kürenin üzerinden bir sıvı akışındaki C_D sürtünme katsayısı Reynolds sayısının fonksiyonu olarak değişimi görülmektedir. Doğal kübik şerit interpolasyonu kullanarak C_D değerini *Re* = 5, 50, 500 and 5000 değerleri için bulunuz.

Not: Logaritmik skala kullanabilirsiniz.

Re	0.2	2	20	200	2000	20 000
c_D	103	13.9	2.72	0.800	0.401	0.433

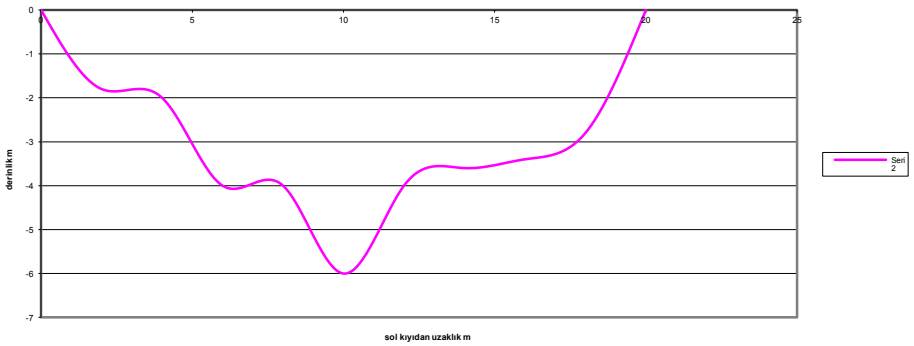
PROBLEM 10

4 komşu nokta kullanan Lagrange veya Newton interpolasyon formülü kullanarak problem 9'u çözünüz.

PROBLEM 11

Bir kanalın derinlik profili şekilde gösterilmektedir. kanalın derinlik ve su hızı verisi aşağıdaki tabloda verilmiştir.

x, m	0	2	4	6	8	10	12	14	16	18	20	
H, m	0	1.8	2	4	4	6	4	3.6	3.4	2.8	0	
U m/s	0	0.03	0.045	0.055	0.065	0.12	0.07	0.06	0.05	0.04	0	



(bolum6_045.jpg,Resimaltı: Problem 11 kanal profili**)**

H(x) ve U(x) fonksiyonlarını

a) Kübik şerit interpolasyon kullanarak hesaplayınız
b) Polinom kareler yöntemi kullanarak hesaplayınız
c) Lagrangeinterpolasyonu kullanarak hesaplayınız

PROBLEM 14

Katı bir cisimdeki sıcaklığın dağılımı

Zaman , t, s	0	5	10	15	20	25	30	35
Sıcaklık T C	90	55	40	28	24	22	21.5	20.6

Şeklinde verilmiştir.

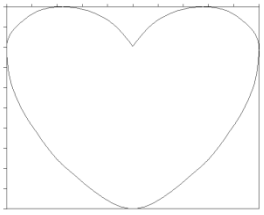
t=13 sn ve t=23 sn t=33 sn, t=43sn ve t=53 sn deki sıcaklık dağılımını bulunuz. Kullanacağınız yöntemler size bırakılmıştır . Son 2 değer için tahmin değerlerinizi ne şekilde bulacağınız konusunda özellikle dikkatli olunuz.

PROBLEM 15

Bir kalp şeklinin koordinatları tabloda verilmiştir. Verilen şekle bir eğri uydurunuz.

Xi	yi	Xi	yi	Xi	yi	xi	Yi
241	127	331	157	215	246	149	120
258	107	323	178	199	232	160	106
283	96	312	197	184	217	175	97
307	97	298	217	170	197	199	96
322	106	283	232	159	178	224	107
333	120	267	246	151	157	241	127
334	138	241	258	148	138		

Not: Aynı değerler yinelenmektedir.



(bolum6_046.jpg,Resimaltı: Problem 15 kalp şekli**)**

PROBLEM 16

Bir aracın durması için gerekli uzunluk, hızının fonksiyonudur. Bu ilişki aşağıdaki tabloda verilmiştir.

Hız (km/h)	**24**	**32**	**40**	**48**
Durma mesafesi (m)	**4.8**	**6.0**	**10**	**12**

38 km/h hızla giden aracın durma mesafesini(m). Newton interpolasyon polinomunu kullanarak hesaplayınız.

PROBLEM 17

Suyun sıcaklıkla ısıl iletkenlik katsayısının değişimi için aşağıdaki tablodaki veriler bilinmektedir.

Sıcaklık ***T*** . [K]	275	290	305	320	340	360
Isıl iletkenlik katsayısı ***k***. [W/(m·K)]	0.574	0.598	0.620	0.640	0.660	0.674

a-) Isıl iletkenlik katsayısının sıcaklıkla değişimi için:

$k = A \cdot T + B$ doğrusunu uydurunuz.

b-) 300 K sıcaklığı için, suyun ısıl iletkenlik katsayısını tahmin ediniz.

PROBLEM 18

Etilen glikol için Prandtl (Pr) sayısının sıcaklıkla değişimi aşağıda verilmiştir. Bu verileri kullanarak Pr sayısı ile sıcaklık arasındaki polinom eşitliğini türetiniz ve bu eşitlikten yararlanarak 42 °C sıcaklıktaki Pr sayısını hesaplayınız.

Sıcaklık (K)	280	300	320	340
Pr sayısı	400	151	73.5	42.8

PROBLEM 19

Suyun doymuş sıvı entalpisinin sıcaklıkla değişimini belirlemek için, t oC biriminde sıcaklık olmak üzere

$$h_f = A + Bt + Ct^2 + Dt^3$$

biçiminde bir kübik eşitlik uydurulmak istenmektedir. Aşağıdaki veriler termodinamik tablolardan alınmıştır. Bu verilerle belirtilen eşitlikteki sabitleri, en küçük kareler eğri uydurma yöntemi ile

t, ^oC	10	30	50	70	90
h_f , kJ/kg	41.99	125.66	209.26	292.97	376.94

PROBLEM 20

Hooke yasasına göre bir lineer yaya uygulanan kuvvet F=kx dir. Burada k yay katsayısı, x ise yaydaki uzamadır. Aşağıdaki tabloda, bir yaya uygulanan kuvvet ve uzama miktarları deney sonuçları verilmiştir. Bu verilere göre yay sabitini tahmin ediniz?

X, m	0.02	0.04	0.06	0.08	0.1
F, kN	3.1	6.1	9.2	12	15.1

PROBLEM 21

Aşağıdaki verilere $f(x) = (Ax+B)^{-2}$ eğrisini uydurunuz.

X	-1	0	1	2
Y	13.46	3.02	0.68	0.14

PROBLEM 22

Kalınlığı 5 mm olan levha içinde bir ısı kaynağı tarafından 2.7 x 10^7 W/m^3 ısı üretilmektedir. Levha malzemesinin ısı iletim katsayısı k=25 W/m K'dir. Levhanın bir yüzeyi (x=0) 161.5 °C, diğer yüzeyi (x=5 mm) ise 166.9 °C sıcaklıklarında bulunmaktadır. Levha içindeki farklı konumlarda sıcaklık değerleri tabloda verilmiştir. Bu verileri kullanarak Newton interpolasyon polinomunu kullarak levha içindeki sıcaklık dağılımını polinom şeklinde ifade ediniz. X=4 mm düzlemindeki sıcaklık değerini hesaplayınız.

x (mm)	0	0.5	2.5	4.5	5.0
T (°C)	161.5	166.9	177.7	171.22	166.9

PROBLEM 23

Dikey bir levha üzerinden serbest taşınım, sınır tabakası içinden değişik noktalarda gerçekleştirilen ölçümlerde belirlenen hızlar (boyutsuz formda) tablodaki gibidir. Bu verilere y hız x yüzeyden mesafe olmak üzere:
$y=a_0+a_1x+a_2x^2$
polinomunu en küçük kareler yöntemi ile uydurunuz.

k	**1**	**2**	**3**	**4**	**5**	**6**
x_k	0.0	0.2	0.4	0.6	0.8	1.0
y_k	0..00	1.05	0.85	0.35	0.10	1.00

PROBLEM 24

Çok düşük basınçlarda (P<10 kPa) kızgın su buharının entalpisinin belirlenmesi için aşağıdaki verilerden yararlanarak

h=A+BT

biçiminde bir eşitlik öneriniz. Burada A ve B sabitler , h ,(kJ/kg) olarak entalpi ve T , (oC) olarak sıcaklıktır.

h (kJ/kg)	2510.6	2519.8	2528.9	2538.1	2547.2	2556.3	2565.3
T (^oC)	5	10	15	20	25	30	35

PROBLEM 25

Bir doğru akım elektrik motorunda akımla (I) , moment (F) arasında

$I=A\ (F^B)$

İlişkisi mevcuttur. Motorla ilgili deneysel ölçmeler aşağıdaki tablodaki gibi verilmiştir.

F (Nm)	100	200	300	400
I (Amper)	8	11.31	13.85	16

Bu ilişkiyi en küçük kareler yöntemi ile belirleyiniz.

PROBLEM 26

Lagrange enterpolasyon yöntemini kullanarak sıcaklığın (t °C) fonksiyonu olarak atmosferik basınçtaki kuru havanın yoğunluğunu ρ (kg/ m^3) veren eşitliği türetiniz. 22,5 °C'da kuru havanın yoğunluğunu hesaplayınız.

t (°C)	0	20	40	60
ρ (kg/ m^3)	1,2930	1,2045	1,1267	1,0595

PROBLEM 27

z değişkeni için araştırılan fonksiyon,

z=ax+by+cxy

biçimindedir. En küçük kareler yöntemini kullanarak aşağıdaki veriler için, a,b,c katsayılarının en uygun değerlerini belirleyiniz.

z	**x**	**Y**
0.1	1	1
-0.9	1	2
2.0	2	2
-1.8	3	1

PROBLEM 29

Su buharının doyma basıncı sıcaklıkla bağlantılı olarak değişmektedir. *P=f(t)* ilişkisi için önerilen eşitliklerden birisi de "*Dupperet Eşitliği*" olarak bilinir. Bu eşitlik:

$$P_{doyma} = A\left(\frac{t_{doyma}}{100}\right)^4$$

biçimindedir. Burada t_{doyma} :^{o}C biriminde alındığında , P_{doyma}, kPa biriminde hesaplanmaktadır. Sıcaklıkla su buharının doyma basıncının değişimi ile ilgili aşağıdaki tablodaki değerler verilmektedir. En küçük kareler tekniği ile "A "katsayısını belirleyiniz.

t, ^{o}C	10	30	60	90	110	140	180	200	250	300	340	370
P kPa	1.2276	4.246	19.940	70.14	143.27	361.3	1002.1	1553.8	3973	8581	14586	21030

PROBLEM 30

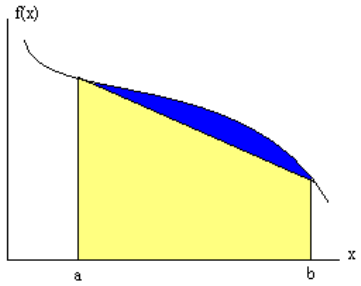
Kalınlığı 5 mm olan levha içinde bir ısı kaynağı tarafından 2,7* 10^7 W/m^3 ısı üretilmektedir.Levha malzemesinin ısı iletim katsayısı k=25 W/m K'dir. Levhanın bir yüzeyi (x=0) 161,5 °C, diğer yüzeyi (x=5 mm) ise 166,9 °C sıcaklıklarında bulunmaktadır. Levha içindeki faklı konumlarda sıcaklık değerleri tabloda verilmiştir. Bu verileri kullanarak Newton enterpolasyon yönteminden yararlanarak levha içindeki sıcaklık dağılımını polinom şeklinde ifade ediniz.X= 4 mm düzlemindeki sıcaklık değerini hesaplayınız.

X(mm)	0	0,5	2,5	4,5	5
T (°C)	161,5	166, 9	177, 7	171, 22	166,9

BÖLÜM 7 İntegral ve türev

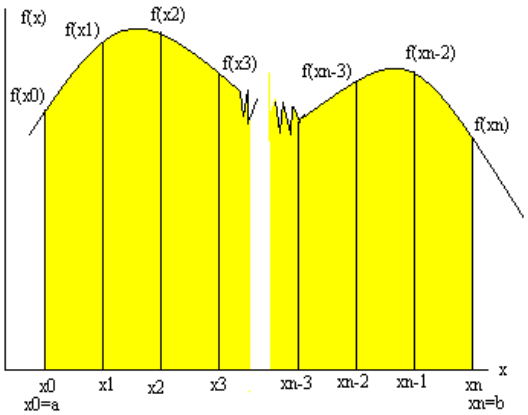
7.1 Newton-cotes formulleri : trapez(yamuk), Bole, Simpson 1/3 ve 3/8 metodları

f(x) fonksiyonu verilmiş olsun. a ve b noktaları arasındaki integralini sayısal olarak hesaplamak istiyelim. Integral fonksiyonun altındaki alan olduğundan en basit yaklaşım iki noktayı bir doğru ile birleştirip altındaki alanı hesaplamaktır. Bu yaklaşım bize trapez(yamuk) denklemini verir.



(bolum7_000.jpg,Resimaltı: **trapez-yamuk yaklaşımı düz bir doğru altındaki alan yaklaşımı**)

$$I = \int_a^b f(x)dx = (b-a)\frac{f(b)+f(a)}{2}$$



(bolum7_001.jpg,Resimaltı: **çoklu trapez -çoklu yamuk integral yaklaşımı**)

Eğer toplam alanı küçük parçacıklara bölersek hata miktarı azalacaktır. Küçük parçacıklar için eğri formülü doğru formülüne yaklaşacaktır. Bu durumda integral formülü

$$I = \int_a^b f(x)dx = (b-a)\frac{f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)}{2n}$$

şeklinde yazılabilir. Örnek fonksiyon olarak

$I = \int_0^1 \frac{4}{\pi}\sqrt{1-x^2}\,dx = 1.0$ fonksiyonunu alalım. Fonksiyonun değerini bildiğimiz için karşılaştırma yapabiliriz.

Program 7.1.1 Trapezoidal integral kuralı

```
//=====================================================
// Numerical Analysis package in java
// Newton-Cotes integral formula 4 points , n=3
// trapezoidal rule
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class f1 extends f_x
{
  double func(double x)
  {return 4.0/Math.PI*Math.sqrt(1.0-x*x);}
}

class integ6
{

public static double newton_cotes2(f_x ff,double a,double b, int n)
{       //Newton-cotes integral 2 points
        //trapezoidal rule
        // ff integrating function
        // a,b integral limits
        // n : number of sub division h=(b-a)/n;
        double h=(b-a)/n;
        double h1=h;
        double sum=0;
        int p,i;
        double x[]=new double[3];
        double f[]=new double[3];
        for(p=0;p<n;p++)
        {  {for(i=0;i<2;i++) {x[i]=a+i*h1+p*h;f[i]=ff.func(x[i]);}}
           sum+=h*(f[0]+f[1])/2.0;
        }
        return sum;
}

  public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  double i1= newton_cotes2(b1,0.0,1.0,1);
  double hata=1-i1;
  System.out.println("integral of class         f1 : "+i1+"hata = "+hata);
}
}
```

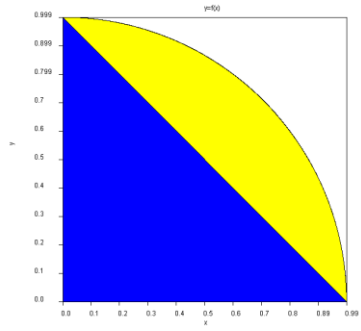
Trapez (yamuk) integral kuralı (0 ile 1 arası tek parça olarak)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ6A
integral of class        f1 : 0.6366197723675814  hata = 0.3633802276324186
```

Burada oluşan hatayı görebilmemiz için grafik çıktısını aldık. Grafik çıktıyı alan trapezoidal integral programı :

Program 7.1.2 Trapezoidal integral kuralı, grafik çizmeli versiyonu

```
//=====================================================
// Numerical Analysis package in java
// Newton-Cotes integral formula 4 points , n=3
// trapezoidal rule
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class f1 extends f_x
{
  double func(double x)
  {return 4.0/Math.PI*Math.sqrt(1.0-x*x);}
}
class integ6A
{
public static double newton_cotes2(f_x ff,double a,double b, int n)
{       //Newton-cotes integral 2 points
        //trapezoidal rule
        double h=(b-a)/n;
        double h1=h;
        double sum=0;
        int p,i;
        int nn=2*n;
        double x[]=new double[2];
        double f[]=new double[2];
        double xi[]=new double[nn];
        double fi[]=new double[nn];
        int j=0;
        for(p=0;p<n;p++)
        {  {for(i=0;i<2;i++) {x[i]=a+i*h1+p*h;xi[j]=x[i];f[i]=ff.func(x[i]);fi[j]=f[i];j++;}}
           sum+=h*(f[0]+f[1])/2.0;
        }
        Plot pp=new Plot(xi,fi);
        pp.addFunction(ff,0.0,1.0,2000,2);
        pp.plot();
        return sum;
}

  public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  double i1= newton_cotes2(b1,0.0,1.0,1);
  double hata=1-i1;
  System.out.println("integral of class        f1 : "+i1+"hata = "+hata);

  }
}
```

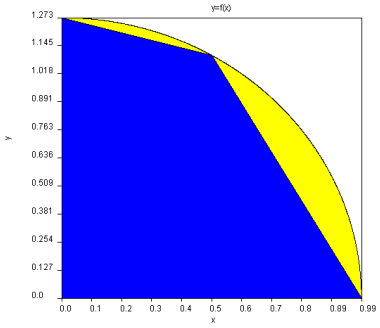


(bolum7_002.jpg,Resimaltı: **Trapez -yamuk integral kuralı 0 ile 1 arası tek parça olarak grafik gösterim**)

İntegral bölgesini 2 eşit bölgeye bölerek trapez kuralını uygularsak :

Trapez (yamuk) integral kuralı (0 ile 1 arası 2 parça olarak)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ6A
integral of class        f1 : 0.8696387816055828hata = 0.13036121839441717
```

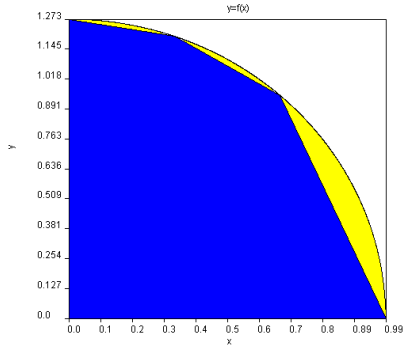


(bolum7_003.jpg,Resimaltı: **Trapez (yamuk) integral kuralı (0 ile 1 arası 2 parça olarak) grafik gösterim**)

Trapezi 3 parça olarak hesaplarsak

Trapez (yamuk) integral kuralı (0 ile 1 arası 3 parça olarak)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ6A
integral of class        f1 : 0.9286860839333196hata = 0.07131391606668036
> Terminated with exit code 1.
```



(bolum7_004.jpg,Resimaltı: **Trapez (yamuk) integral kuralı (0 ile 1 arası 3 parça olarak) grafik gösterim**)

Trapezi 10 parça için hesapladığımızda

Trapez (yamuk) integral kuralı (0 ile 1 arası 10 parça olarak)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ6A
integral of class        f1 : 0.9881988750835945hata = 0.011801124916405503
```

Trapezi 100 parça olarak hesaplarsak :

Trapez (yamuk) integral kuralı (0 ile 1 arası 100 parça olarak)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ6A
integral of class       f1 : 0.9996257879552262hata = 3.742120447738362E-4
```

Görüldüğü gibi step sayısı arttıkça trapez integral tahmini gerçek sayıya yaklaşıyor. Ancak yeterince iyi bir tahmin için step sayısını oldukça arttırmamız gerekebilir. Daha az step sayısıyla daha iyi integral tahmini yapabilmek için birinci derece polinom(doğru) yerine daha yüksek dereceden polinom formülleri kullanılabilir.

h= (b – a)/n

dersek **ikinci derece polinom formülü(Simpson 1/3 kuralı)** n=2, 3 noktalı Newton-Cotes formülü

$$I=\int_a^b f(x)dx=\frac{(b-a)}{6}[f(x_0)+4f(x_1)+f(x_2)]$$

veya n=2 için

$$h=\frac{(b-a)}{n}=\frac{(b-a)}{2}\qquad I=\int_a^b f(x)dx=\frac{h}{3}[f(x_0)+4f(x_1)+f(x_2)]$$

ikinci derece polinom formülü eşit terimler kullanımıyla:

$$I=\int_a^b f(x)dx=(b-a)\frac{f(x_0)+4\sum_{i=1,3,5}^{n-1}f(x_i)+\sum_{j=2,4,6}^{n-2}f(x_j)+f(x_n)}{6}$$

üçüncü derece polinom formülü(Simpson 3/8 kuralı) n=3, 4 noktalı Newton-Cotes formülü

$$I=\int_a^b f(x)dx=\frac{(b-a)}{8}[f(x_0)+3f(x_1)+3f(x_2)+f(x_3)]$$

veya n=3 için

$$h=\frac{(b-a)}{n}=\frac{(b-a)}{3}\qquad I=\int_a^b f(x)dx=\frac{3h}{8}[f(x_0)+3f(x_1)+3f(x_2)+f(x_2)]$$

dördüncü derece polinom formülü (Bode formülü) n=4, 5 noktalı Newton-Cotes formülü

$$I=\int_a^b f(x)dx=\frac{(b-a)}{90}[7f(x_0)+32f(x_1)+12f(x_2)+32f(x_3)+7f(x_4)]$$

veya n=4 için

$$h=\frac{(b-a)}{n}=\frac{(b-a)}{4}\qquad I=\int_a^b f(x)dx=\frac{2h}{45}[7f(x_0)+32f(x_1)++12f(x_2)+32f(x_3)+7f(x_4)]$$

beşinci derece polinom formülü n=5, 6 noktalı Newton-Cotes formülü

$$I=\int_a^b f(x)dx=\frac{(b-a)}{288}[19f(x_0)+75f(x_1)+50f(x_2)+50f(x_3)+75f(x_4)+19f(x_5)]$$

veya n=5 için

$$h=\frac{(b-a)}{n}=\frac{(b-a)}{5}\qquad I=\int_a^b f(x)dx=\frac{5h}{288}[19f(x_0)+75f(x_1)+50f(x_2)+50f(x_3)+75f(x_4)+19f(x_5)]$$

sekizinci derece polinom formülü n=8, 9 noktalı Newton-Cotes formülü

$$I=\int_a^b f(x)dx=\frac{(b-a)}{28350}[989f(x_0)+5888f(x_1)-928f(x_2)+10496f(x_3)-4540f(x_4)$$
$$+10496f(x_5)-928f(x_6)+5888f(x_7)+989f(x_8)]$$

veya n=8 için

$$h=\frac{(b-a)}{n}=\frac{(b-a)}{8} \qquad I=\int_a^b f(x)dx=\frac{4h}{14175}[989f(x_0)+5888f(x_1)-928f(x_2)+10496f(x_3)-4540f(x_4)$$
$$+10496f(x_5)-928f(x_6)+5888f(x_7)+989f(x_8)]$$

Bu denklemlerin uygulandığı programlar altta verilmiştir.

Program 7.1.3 2. derece polinom integral (Simpson 1/3) kuralı

```
//=======================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =======================================================
import java.io.*;


class f1 extends f_x
{
  double func(double x)
  {return 1.0/(1.0+x*x);}
}

class integ5
{
public static double newton_cotes3(f_x ff,double a,double b, int n)
{      //Newton-cotes integral 3 points
       //Simpson 1/3 rule

       double h=(b-a)/n;
       double h1=h/2.0;
       double sum=0;
       int p,i;
       double x[]=new double[3];
       double f[]=new double[3];
       for(p=0;p<n;p++)
       {  {for(i=0;i<3;i++) {x[i]=a+i*h1+p*h;f[i]=ff.func(x[i]);}}
          sum+=h*(f[0]+4.0*f[1]+f[2])/6.0;
       }
       return sum;
}

  public static void main(String args[]) throws IOException
  {
   f1 b1=new f1();
  double i1=  newton_cotes3(b1,0.0,1.0,1);
  double i2 = Math.atan(1.0)-Math.atan(0);
  double hata =i2-i1;
  System.out.println("integral of class        f1 : "+i1+"hata="+hata);
}
}
```

2. derece polinom integral kuralı (simpson 1/3 kuralı 1 parça olarak)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ5
integral of class        f1 : 0.9473117846849166 hata = 0.052688215315083387
> Terminated with exit code 0.
```

2. derece polinom integral kuralı (simpson 1/3 kuralı 2 parça olarak)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ5
integral of class        f1 : 0.9815388228080559hata = 0.018461177191944067
> Terminated with exit code 0.
```

Program 7.1.4 3. derece polinom integral kuralı(simpson 3/8 kuralı)

```
//=====================================================
// Numerical Analysis package in java
// Newton-Cotes integral formula 4 points , n=3
// Simpson 3/8 rule
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class f1 extends f_x
{
  double func(double x)
  {
  return 4.0/Math.PI*Math.sqrt(1.0-x*x);
  }
}

class integ4
{
public static double newton_cotes4(f_x ff,double a,double b, int n)
{       //Newton-cotes integral 4 points
        //Simpson 3/8 rule
        // ff integrating function
        // a,b integral limits
        // n : number of sub division h=(b-a)/n;
                    double h=(b-a)/n;
        double h1=h/3.0;
        double sum=0;
        int p,i;
        double x[]=new double[4];
        double f[]=new double[4];
        for(p=0;p<n;p++)
        {  {for(i=0;i<4;i++) {x[i]=a+i*h1+p*h;f[i]=ff.func(x[i]);}}
           sum+=h*(f[0]+3.0*f[1]+3.0*f[2]+f[3])/8.0;
        }
        return sum;
}

public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  double i1= newton_cotes4(b1,0.0,1.0,2);
  double hata=1-i1;
  System.out.println("integral of class         f1 : "+i1+"hata = "+hata);
}}
```

3. derece polinom integral kuralı(Simpson 3/8 kuralı, 1 parça)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ4
integral of class         f1 : 0.9877870530380952hata = 0.01221294696190478

> Terminated with exit code 0.
```

Program 7.1.5 4. derece polinom(Boole) integral kuralı, 1 parça

```
//=====================================================
// Numerical Analysis package in java
// Newton-Cotes integral formula 5 points , n=4
// Boole formula
// Dr. Turhan Coban
// =====================================================
import java.io.*;
```

```
class f1 extends f_x
{
  double func(double x)
  {
  return 4.0/Math.PI*Math.sqrt(1.0-x*x);
  }
}

class integ3
{

public static double boole(f_x ff,double a,double b, int n)
{  return newton_cotes5(ff,a,b,n);}

public static double newton_cotes5(f_x ff,double a,double b, int n)
{       //Newton-cotes integral 5 points
        //Boole rule
        // ff integrating function
        // a,b integral limits
        // n : number of sub division h=(b-a)/n;
                    double h=(b-a)/n;
        double h1=h/4.0;
        double sum=0;
        int p,i;
        double x[]=new double[5];
        double f[]=new double[5];
        for(p=0;p<n;p++)
        {  {for(i=0;i<5;i++) {x[i]=a+i*h1+p*h;f[i]=ff.func(x[i]);}}
           sum+=h*(7.0*f[0]+32.0*f[1]+12.0*f[2]+32.0*f[3]+7.0*f[4])/90.0;
        }
        return sum;
}

   public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  double i1=  newton_cotes5(b1,0.0,1.0,1);
  double i2 = 1.0;
  double hata =i2-i1;
  System.out.println("integral of class         f1 : "+i1+" hata="+hata);
 }
}
```

4. derece polinom(boole) integral kuralı (1 parça)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ3
integral of class         f1 : 0.9838206253495985  hata=0.016179374650401535

> Terminated with exit code 0.
```

Program 7.1.6 5. derece polinom integral kuralı

```
//=====================================================
// Numerical Analysis package in java
// Newton-Cotes integral formula 6 points, n=5
//
// Dr. Turhan Coban
// =====================================================
import java.io.*;

class f1 extends f_x
{
  double func(double x)
  {
  return 4.0/Math.PI*Math.sqrt(1.0-x*x);
```

```
  }
}

class integ2
{

public static double newton_cotes6(f_x ff,double a,double b, int n)
{       //Newton-cotes integral 6 points, n=5
        // ff integrating function
        // a,b integral limits
        // n : number of sub division h=(b-a)/n;
                    double h=(b-a)/n;
        double h1=h/5.0;
        double sum=0;
        int p,i;
        double x[]=new double[6];
        double f[]=new double[6];
        for(p=0;p<n;p++)
        {  {for(i=0;i<6;i++) {x[i]=a+i*h1+p*h;f[i]=ff.func(x[i]);}}
           sum+=h*(19.0*f[0]+75.0*f[1]+50.0*f[2]+50.0*f[3]+75.0*f[4]+19.0*f[5])/288.0;
        }
        return sum;
}

        public static void main(String args[]) throws IOException
        { f1 b1=new f1();
          double i1=  newton_cotes6(b1,0.0,1.0,1);
          double i2 = 1.0;
          double hata =i2-i1;
          System.out.println("integral of class         f1 : "+i1+" hata="+hata);
        }
}
```

5. derece polinom integral kuralı

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ2
integral of class         f1 : 0.9872489399864514 hata=0.012751060013548643

> Terminated with exit code 0.
```

Program 7.1.7 8. derece polinom integral kuralı

```
//=====================================================
// Numerical Analysis package in java
// Newton-Cotes integral formula 9 points, n=8
//
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class f1 extends f_x
{
  double func(double x)
  {
  return 4.0/Math.PI*Math.sqrt(1.0-x*x);
  }
}



class integ2A
{

public static double newton_cotes9(f_x ff,double a,double b, int n)
{       //Newton-cotes integral 9 points, n=8
```

```
      // ff integrating function
      // a,b integral limits
      // n : number of sub division h=(b-a)/n;
                double h=(b-a)/n;
      double h1=h/8.0;
      double sum=0;
      int p,i;
      double x[]=new double[9];
      double f[]=new double[9];
      for(p=0;p<n;p++)
      {  {for(i=0;i<9;i++) {x[i]=a+i*h1+p*h;f[i]=ff.func(x[i]);}}
        sum+=h/28350.0*(989.0*f[0]+5888.0*f[1]-928.0*f[2]+10496.0*f[3]-4540.0*f[4]+
        10496.0*f[5]-928.0*f[6]+5888.0*f[7]+989.0*f[8]);
      }
      return sum;
}

      public static void main(String args[]) throws IOException
      { f1 b1=new f1();
        double i1=  newton_cotes9(b1,0.0,1.0,1);
        double i2 = 1.0;
        double hata =i2-i1;
        System.out.println("integral of class         f1 : "+i1+" hata="+hata);
      }
}
```

8. derece polinom integral kuralı

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ2A
integral of class         f1 : 0.9952701383291108 hata=0.004729861670889179

> Terminated with exit code 0.
```

İntegral sonuçlarını bir tablo halinde özetlersek

Tablo 7.1-1 örnek fonksiyon Newton-Cotes formülleri sonuçlarının karşılaştırılması

$$I = \int_0^1 \frac{4}{\pi}\sqrt{1-x^2}\,dx = 1.0$$

N	Integral yaklaşımı	Hata miktarı ($I_{yaklaşım}$-I_{tam})
Bölgeyi 1 parça (0-1) alarak integrali hesaplarsak		
1 (Trapezoidal)	0.6366197723675814	0.3633802276324186
2 (Simpson 1/3)	0.9473117846849166	0.052688215315083387
3 (Simpson 3/8)	0.965194372879037	0.03480562712096302
4 (Bode)	0.9838206253495985	0.016179374650401535
5	0.9872489399864514	0.012751060013548643
8	0.9952701383291108	0.004729861670889179
Bölgeyi 2 eşit parçaya (0-0.5 ve 0.5-1) bölerek integrali hesaplarsak		
1 (Trapezoidal)	0.8696387816055828	0.13036121839441717
2 (Simpson 1/3)	0.9815388228080559	0.018461177191944067
3 (Simpson 3/8)	0.9877870530380952	0.01221294696190478
4 (Bode)	0.9943031391613758	0.005696860838624196
5	0.9955081956385212	0.0044918043614787795
8	0.998330566008359	0.0044918043614787795
Bölgeyi 10 eşit parçaya (0-0.1,0.1-0.2,…,0.9-1) bölerek integrali hesaplarsak		
1 (Trapezoidal)	0.9881988750835945	0.011801124916405503
2 (Simpson 1/3)	0.9983621080451945	0.0016378919548054593
3 (Simpson 3/8)	0.998914915292993	0.001085084707006967

4 (Bode)	0.9994921635232705	5.07836476729473E-4
5	0.9995994322513025	4.005677486974557E-4
8	0.9998508846967017	1.4911530329830835E-4

7.2 Richardson extrapolasyonu (Romberg integrasyonu) ve Aitken extrapolasyonu

İntegral formülündeki hataları azaltma yollarından birisi Richardson ekstrapolasyonu uygulamaktır.
Richardson interpolasyon formülü

$I \cong I(h_2) + \frac{1}{(h_1/h_2)^2 - 1}[I(h_2) - I(h_1)]$ şeklinde yazılabilir. Burada h_2, h_1 den küçük olmalıdır. Eğer $h_2=h_1/2$ olarak verilirse

$I \cong \frac{4}{3} I(h_2) - \frac{1}{3} I(h_1)]$ şeklini alır.

İnterpolasyon başlangıcı için polinom yaklaşımlarından birisi kullanılabilir.
İteratif formülümüz genel olarak yazılırsa

$I_{j,k} \cong \frac{4^{k-1} I_{j+1,k-1} - I_{j,k-1}}{4^{k-1} - 1}$ haline gelir. Bu genel denklem başlangıç integral değerleri olarak trapez kuralından başlayarak uygulandığında Romberg integrasyonu adını alır.

Program 7.2.1 Romberg integrali

```
//=====================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class f1 extends f_x
{
  double func(double x)
  {
  return 2.0/(1.0+2.0*x*x);
  }
}

class integ7
{
public static double integral_romberg(f_x ff,double a,double b)
{
//integral f(x)dx
//romberg integration
int n=8;//increase n to increase accuracy
double R[][];
R=new double[n+1][n+1];
int m=1;
double h=b-a;
double close=1;
double tol=1e-40;
int j=0,k=0;
double ret=0;
R[0][0]=h/2.0*(ff.func(a)+ff.func(b));
do
{
  j++;
```

```
  h/=2.0;
  trap(ff,a,h,j,m,R);
  m*=2;
  for(k=1;k<=j;k++)
  {
   R[j][k]=R[j][k-1]+(R[j][k-1]-R[j-1][k-1])/(Math.pow(4,k)-1.0);
   close=Math.abs(R[j-1][j-1]-R[j][j]);
   ret=R[j][k];
  }
}while(close>tol && j<n);
return ret;
}

public static void trap(f_x ff,double a,double h, int j,int m,double R[][])
{
   //Newton-cotes integral 2 points
              //  successive trapezoidal integration rule
   //  this program will be utilised in romberg integration
      double sum=0;
      int p;
      for(p=1;p<=m;p++)
        { sum+=ff.func(a+h*(2*p-1)); }
      R[j][0]=R[j-1][0]/2.0+h*sum;
}

  public static void main(String args[]) throws IOException
  {
  //Romberg integrali
  f1 b1=new f1();
  System.out.println("integral of class        f1 : "+
  integral_romberg(b1,-3.0,3.0));
 }}
```

Çıktı 7.2-1 Romberg integrali f(x)= 2.0/(1.0+2.0*x*x)

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ7
integral of class        f1 : 3.7881660832213675

> Terminated with exit code 0.
```

Oluşan Romberg tablosunu verirsek

Romberg integrali $I=\int_{-3}^{3}\frac{2}{1+2x^2}dx$ **tablo değerleri**

6.3157895	8.2105263							
4.2488038	3.5598086	3.2497608						
3.8058295	3.6581714	3.664729	3.6713158					
3.7866756	3.7802909	3.7884322	3.7903958	3.7908628				
3.7877768	3.7881439	3.7886674	3.7886711	3.7886644	3.7886622			
3.7880687	3.788166	3.7881675	3.7881596	3.7881576	3.7881571	3.7881569		
3.7881417	3.7881661	3.7881661	3.7881661	3.7881661	3.7881661	3.7881661	3.7881661	
3.78816	3.7881661	3.7881661	3.7881661	3.7881661	3.7881661	3.7881661	3.7881661	3.7881661

Şimdi de $I=\int_{0}^{\pi}\sin(x)dx=2$ integralini Romberg integral formülü kullanarak çözelim

Çıktı 7.2-3 Romberg integrali $I=\int_{0}^{\pi}\sin(x)dx=2$

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ7A
integral of class        f1 : 1.9999999999999993
```

Romberg integrali $I = \int_0^{\pi} \sin(x)dx = 2$ **tablo değerleri**

1.5707963267948900	2.0943951023931900							
1.8961188979370300	2.0045597549844200	1.9985707318238300						
1.9742316019455500	2.0002691699483800	1.9999831309459800	2.0000055499796700					
1.9935703437723300	2.0000165910479300	1.9999997524545700	2.0000000162880400	1.9999999945872900				
1.9983933609701400	2.0000010333694100	1.9999999961908400	2.0000000000596700	1.9999999999960300	2.0000000000013200			
1.9995983886400300	2.0000000645300000	1.9999999999407000	2.0000000000002200	1.9999999999999900	1.9999999999999900	1.9999999999999900		
1.9998996001842000	2.0000000040322500	1.9999999999990700	2.0000000000000000	2.0000000000000000	2.0000000000000000	2.0000000000000000	2.0000000000000000	
1.9999749002350500	2.0000000002520000	1.9999999999999800	1.9999999999999900	1.9999999999999900	1.9999999999999900	1.9999999999999900	1.9999999999999900	1.9999999999999900

Diğer bir interpolasyon formülü de Aitken interpolasyon formülüdür. Temel olarak Euler-McLaurin formülünden türetilen Aitken interpolasyon formülü :

$$I = I_{4n} \cong I_{4n} - \frac{(I_{4n} - I_{2n})^2}{(I_{4n} - I_{2n}) - (I_{2n} - I_n)}$$ şeklindedir.

Program 7.2.2 Aitken interpolasyonlu simpson integrali

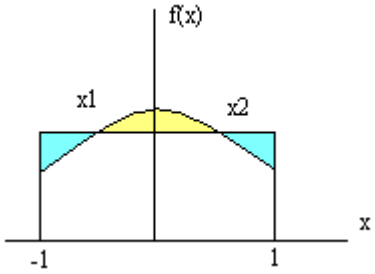
```
//=====================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class f1 extends f_x
{
  double func(double x)
  {return x*Math.sqrt(x);}
}


class integ11
{

public static double newton_cotes3(f_x ff,double a,double b, int n)
{      //Newton-cotes integral 3 points
       //Simpson 1/3 rule

       double h=(b-a)/n;
       double h1=h/2.0;
       double sum=0;
       int p,i;
       double x[]=new double[3];
       double f[]=new double[3];
       for(p=0;p<n;p++)
       {  {for(i=0;i<3;i++) {x[i]=a+i*h1+p*h;f[i]=ff.func(x[i]);}}
          sum+=h*(f[0]+4.0*f[1]+f[2])/6.0;
       }
       return sum;
}

public static double Aitken_simpson1_3(f_x ff,double a,double b, int n)
{   //Aitken interpolation formula
              double I4n=newton_cotes3(ff,a,b,4*n);
    double I2n=newton_cotes3(ff,a,b,2*n);
    double In=newton_cotes3(ff,a,b,n);
```

```
    double I=I4n-(I4n-I2n)*(I4n-I2n)/((I4n-I2n)-(I2n-In));
    //System.out.println("I4n="+I4n+"I2n="+I2n+"In="+In+"I="+I);
    return I;
}

  public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  for(int n=2;n<257;n*=2)
    System.out.println("n : "+n+"I = "+Aitken_simpson1_3(b1,0.0,1.0,n));
}
}
```

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ11
n : 2    I   = 0.39999984745800915
n : 4    I   = 0.3999999902487645
n : 8    I   = 0.3999999993877008
n : 16   I   = 0.3999999999617067
n : 32   I   = 0.3999999999976067
n : 64   I   = 0.3999999999998505
n : 128  I   = 0.3999999999999909
n : 256  I   = 0.4
> Terminated with exit code 0.
```

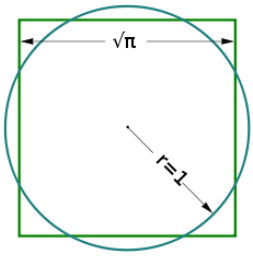
7.3 Gauss-Legendre, Gauss-Chebchev, Gauss-Jacobi, Gauss-Hermit, Gauss-Lequerre integral formülleri

Antik zamanlardan beri bilinen bir problem bir polinomla eşit alana sahip bir dikdörtgen oluşturma problemidir.



(bolum7_005.jpg,Resimaltı: polinom ile eşdeğer alana sahip doğru problemi)

Problemi bir polinomum bir doğru ile kesimiyle oluşacak alt alan olarak alırsak, ve eğer doğrunun polinomu alt ve üst tarafta eşit alan bırakacak şekilde kestiği x1 ve x2 noktaları bilinir ise polinomun alanı yerine dikdörtgenin alanını hesaplayarak aynı sonuca ulaşabiliriz. Veya daha genel ifadeyle integral yerine bir dizi toplama işlemi yaparak integral sonucunu doğru olarak saptayabiliriz.



(bolum7_006.jpg,Resimaltı: kare ile aynı alana sahip daire)

Genel tanım olarak eşdeğer toplama formülü

$$I_w(-1,1) \cong \int_{-1}^{1} f(x)w(x)dx \cong \sum_{k=1}^{n} c_k f(x_k)$$

şeklinde yazabiliriz. Burada w(x) ağırlık fonksiyonudur. Ağırlık fonksiyonunu Gauss-Legendre integral formülü için değerini 1 olarak alabiliriz.

$$I_w(-1,1) \cong \int_{-1}^{1} f(x)dx \cong \sum_{k=1}^{n} c_k f(x_k)$$

integrali genel polinom için çözersek

$\int_{-1}^{1} x^k dx \cong \frac{1-(-1)^{k+1}}{k+1}$ olur. Bu denklemi yukarıdaki integral denklemi için açık yazarsak

$c_1 x_1^k + c_2 x_2^k + ... + c_n x_n^k = \frac{1-(-1)^{k+1}}{k+1}, \quad 0 \le k \le 2n$ olur. Grafikte gösterilen n=2 durumu için :

$f(x)=a_0+a_1x+a_2x^2+a_3x^3$

$$I_w(-1,1) \cong \int_{-1}^{1} f(x)dx \cong c_1 f(x_1) + c_2 f(x_2) = \int_{-1}^{1} (a_0 + a_1 x + a_2 x^2 + a_3 x^3)dx$$

$$c_1(a_0 + a_1 x_1 + a_2 {x_1}^2 + a_3 {x_1}^3) + c_2 f(a_0 + a_1 x_2 + a_2 {x_2}^2 + a_3 {x_2}^3) = a_0 \int_{-1}^{1} dx + a_1 \int_{-1}^{1} x dx + a_2 \int_{-1}^{1} x^2 dx + a_3 \int_{-1}^{1} x^3 dx$$

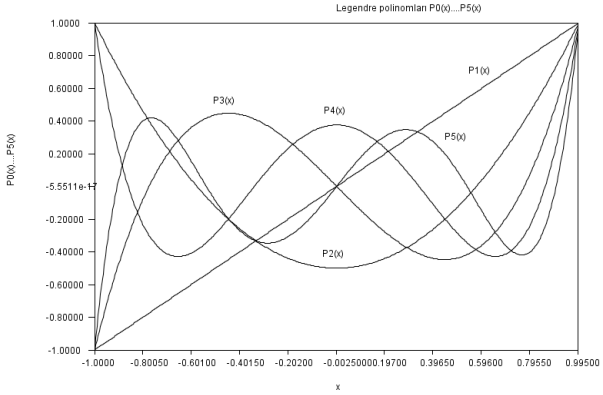
$c_1+c_2=2$

$c_1 x_1+c_2 x_2=0$

$c_1 {x_1}^2+c_2 {x_2}^2=2/3$

$c_1 {x_1}^3+c_2 {x_2}^3=0$

olur. Bu denklem sistemi çözülürse $x_{1,2} = \pm 1/\sqrt{3}$ ve $c_{1,2}=1$ bulunur. Programin genel çözümü Legendre polinomları ile tanımlanabilir. Legendre Polinomları $(k+1)P_{k+1}(x) = (2k+1)xP_k(x) - kP_{k-1}(x), \quad k \ge 1$ denklemiyle tanımlanır. Burada $P_0(x)=1$ ve $P_1(x)=x$ olarak tanımlanmıştır. İlk 6 Legendre polinomunu grafik formda verilmiştir. Legendre polinomları Mathd sınıfında P(n,x) olarak yer almaktadır.



(bolum7_007.jpg,Legendre polinomları)

İlk 10 Legendre polinomunu açık olarak listelersek:

n	$P_n(x)$
0	1
1	x
2	$(3x^2-1)/2$
3	$(5x^3-3x)/2$
4	$(35x^4-30x^2+3)/8$
5	$(63x^5-70x^3+15x)/8$
6	$(231x^6-315x^4+105x^2-5)/16$
7	$(429x^7-693x^5+315x^3-35x)/16$
8	$(6435x^8-12012x^6+6930x^4-1260x^2+35)/128$
9	$(12155x^9-25740x^7+18018x^5-4620x^3+315x)/128$
10	$(46189x^{10}-109395x^8+90090x^6-30030x^4+3465x^2-63)/256$

Legendre polinomunun kökleri Gauss integral denkleminin de köklerini verir. Örneğin yukardaki denklemi kullanarak $P_2(x)$ değerini hesaplarsak

$P_2(x)=(3x^2-1)/2$ olur. Bunun da kökü $x_{1,2} = \pm 1/\sqrt{3}$ değerine eşittir. Kök değerleri bulunduktan sonra katsayılar

$c_1 x_1^{2j} + c_2 x_2^{2j} + ... + c_n x_n^{2j} = \dfrac{2}{2j+1}, \quad 0 \le j \le n$ formülünden hesaplanabilir.

veya

$c_k = \dfrac{2(1-x_k^2)}{[nP_{n-1}(x_k)]^2}$ ifadesinden katsayıyı hesaplayabiliriz.

Son olarak burada integral sınırlarının -1 ile 1 aralığında olmasına dikkati çekelim. x=[-1,1] aralığı z=[a,b] aralığına değişken dönüşümü ile rahatlıkla çevrilebilir.

$$z = \left(\frac{b-a}{2}\right)x + \left(\frac{b+a}{2}\right) = \alpha x + \beta$$

$$\alpha = \left(\frac{b-a}{2}\right) \quad \beta = \left(\frac{b+a}{2}\right)$$

bu durumda integral formülü

$$I_w(a,b) \cong \int_a^b f(z)dx \cong \alpha \sum_{k=1}^{n} c_k f(\alpha x_k + \beta)$$

formunu alır. Gauss-Legendre integral formulünün katsayıları ve kök değerleri n=2 den n=10 a kadar Tablo 7.3-1 de verilmiştir.

Tablo 7.3.1 Gauss – Legendre integral katsayıları

n	x_k	c_k
2	-0.577350269189625	1.000000000000000
	0.577350269189625	1.000000000000000
3	-0.774596669241483	0.555555555555552
	0.000000000000000	0.888888888888888
	0.774596669241483	0.555555555555552
4	-0.861136311594052	0.347854845137447
	-0.339981043584856	0.652145154862546
	0.339981043584856	0.652145154862546
	0.861136311594052	0.347854845137447
5	-0.906179845938664	0.236926885056181
	-0.538469310105683	0.478628670499366
	0.000000000000000	0.568888888888888
	0.538469310105683	0.478628670499366
	0.906179845938664	0.236926885056181
6	-0.932469514203152	0.171324492379162
	-0.661209386466264	0.360761573048138
	-0.238619186083196	0.467913934572691
	0.238619186083196	0.467913934572691
	0.661209386466264	0.360761573048138
	0.932469514203152	0.171324492379162
7	-0.949107912342758	0.129484966168862
	-0.741531185599394	0.279705391489276
	-0.405845151377397	0.381830050505119
	0.000000000000000	0.417959183673469

	0.405845151377397	**0.381830050505119**
	0.741531185599394	**0.279705391489276**
	0.949107912342758	**0.129484966168862**
8	**-0.960289856497536**	**0.101228536290369**
	-0.796666477413626	**0.222381034453374**
	-0.525532409916329	**0.313706645877887**
	-0.183434642495649	**0.362683783378362**
	0.183434642495649	**0.362683783378362**
	0.525532409916329	**0.313706645877887**
	0.796666477413626	**0.222381034453374**
	0.960289856497536	**0.101228536290369**
9	**-0.968160239507626**	**0.081274388361569**
	-0.836031107326635	**0.180648160694857**
	-0.613371432700590	**0.260610696402935**
	-0.324253423403808	**0.312347077040002**
	0.000000000000000	**0.330239355001259**
	0.324253423403808	**0.312347077040002**
	0.613371432700590	**0.260610696402935**
	0.836031107326635	**0.180648160694857**
	0.968160239507626	**0.081274388361569**
10	**-0.973906528517171**	**0.066671344308684**
	-0.865063366688984	**0.149451349150580**
	-0.679409568299024	**0.219086362515982**
	-0.433395394129247	**0.269266719309996**
	-0.148874338981631	**0.295524224714752**
	0.148874338981631	**0.295524224714752**
	0.433395394129247	**0.269266719309996**
	0.679409568299024	**0.219086362515982**
	0.865063366688984	**0.149451349150580**
	0.973906528517171	**0.066671344308684**

Örnek Problem:

Aşağıdaki tam çözüm değeri verilmiş integrali **Gauss-Legendre** integral formülü ile hesaplayalım N=3

$$I = \int_0^1 x^2\sqrt{x}dx = \frac{2}{7} = 0.285714$$

x_k	c_k	yk	f	ck*f
-0.774596669	**0.555555556**	0.11270167	0.004264	0.002369
0	**0.888888889**	0.5	0.176777	0.157135
0.774596669	**0.555555556**	0.88729833	0.741608	0.412004
			sum	0.571508
			sum*alpha	0.285754
			exact	0.285714

a	**0**
b	**1**
α	0.5
β	0.5

Örnek problem:

$f(x) = \frac{4}{\pi}\sqrt{1-x^2}$ **integralini** 0 ile 1 arasında Gauss-Legendre integral formülü ile hesaplayalım (N=3)

α=(1-0)/2=0.5 β=(1+0)/2=0.5

I=0.5*(**0.555555555555553** *f(0.5*-0.774596669241483+0.5)+ **0.888888888888889***f(0.5*0+0.5)+
0.555555555555553 *f(0.5*0.774596669241483+0.5))

I=1.004609

Bu bölümdeki ilk programımız Gauss-Legendre integral denkleminin toplam 10 katsayı için olan bir formunu kullanmaktadır.

Program 7.3.1 10 noktalı Gauss-Legendre integrasyonu (kökler programda tanımlanmış)

```
/=====================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class f1 extends f_x
{
  public double func(double x)
  {           return x*x*Math.sqrt(x);}}

class integ8
{
public static double integral(f_x f_xnt,double a,double b)
{
//integral f(x)dx
//integral of a function by using gauss-legendre quadrature
//coefficients are pre-calculated for 60 terms for [-1,1]
//band then utilises variable transform
 double r[],c[];
 r=new double[10];
 c=new double[10];
 r[0]=-0.973906528517171;
 r[1]=-0.865063366688984;
 r[2]=-0.679409568299024;
 r[3]=-0.433395394129247;
 r[4]=-0.148874338981631;
 r[5]=0.148874338981631;
 r[6]=0.433395394129247;
 r[7]=0.679409568299024;
 r[8]=0.865063366688984;
 r[9]=0.973906528517171;

 c[0]=0.066671344308684;
 c[1]=0.149451349150580;
 c[2]=0.219086362515982;
 c[3]=0.269266719309996;
 c[4]=0.295524224714752;
 c[5]=0.295524224714752;
 c[6]=0.269266719309996;
 c[7]=0.219086362515982;
 c[8]=0.149451349150580;
 c[9]=0.066671344308684;

int n=10;
double z=0;
double x,y;
double k1=(b-a)/2.0;
double k2=(b+a)/2.0;
for(int i=0;i<n;i++)
{
x=k2+k1*r[i];
```

```
y=f_xnt.func(x);
z+=k1*c[i]*y;
}
return z;
}

  public static void main(String args[]) throws IOException
  {
  //Gauss-Legendre integrali
  f1 b1=new f1();
  System.out.println("integral of class         f1 : "+integral(b1,0.0,1.0));
}
}
```

10 noktalı Gauss-Legendre integrasyonu

```
 ---------- Capture Output ----------
> "C:\java\bin\java.exe" integ8
integral of class         f1 : 0.28571428571518753

> Terminated with exit code 0.
```

İkinci integral ise:

```
class f1 extends f_x
{
  public double func(double x)
  {            return 4.0/PI*Math.sqrt(1-x*x);}}
---------- Capture Output ----------
> "C:\java\bin\java.exe" integ8
integral of class         f1 : 1.0000027919429444

> Terminated with exit code 0.
```

İkinci programımızda ise katsayılar Legendre polinomları kullanılarak hesaplanmaktadır. Bu ikinci programımızda yer alan gauss_legendre_coefficiens metodunu kullanarak istediğimiz büyüklükteki Gauss-Legendre integral katsayılarını da elde edebiliriz.

Program 7.3.2 değişken nokta sayılı Gauss-Legendre integrasyonu (kökler hesaplanıyor)

```
 //=====================================================
//  java sayısal analiz paketi
//  Gauss - Legendre integrali
//  Katsayılar program tarafından hesaplanıyor
//  Dr. M. Turhan Coban
// =====================================================
import java.io.*;
class f1 extends f_x
{
  public double func(double x)
  {
  return 3.0/2.0*x*x;
  }
}

class integ9
{

public static double[][] gauss_legendre_katsayilari(double x1,double x2,int n)
{
                // Gauss-Legendre integral katsayılarını hesaplar
                // Legendre polinomu kökleri
                double EPS=3.0e-15;
                int m,j,i;
                double z1,z,xm,xl,pp,p3,p2,p1;
```

```
    double a[][]=new double[2][n];//a[0][i]=x[i]   a[1][i]=w[i]
            m=(n+1)/2;
            xm=0.5*(x2+x1);
            xl=0.5*(x2-x1);
            for (i=1;i<=m;i++)  {
                    z=Math.cos(Math.PI*((i-0.25)/(n+0.5)));
                    do {
                            p1=1.0;
                            p2=0.0;
                            for (j=1;j<=n;j++) {
                                    p3=p2;
                                    p2=p1;
                                    p1=((2.0*j-1.0)*z*p2-(j-1.0)*p3)/j;
                            }
                            pp=n*(z*p1-p2)/(z*z-1.0);
                            z1=z;
                            z=z1-p1/pp;
                    } while (Math.abs(z-z1) > EPS);
                    a[0][i-1]=xm-xl*z;
                    a[0][n-i]=xm+xl*z;
                    a[1][i-1]=2.0*xl/((1.0-z*z)*pp*pp);
                    a[1][n-i]=a[1][i-1];
                            }
return a;
}

public static double integral(f_x f_xnt,double x1,double x2,int n)
{
// n : integral katsayısı
// Bu program önce [x1,x2] bandı için Gauss Legender katsayılarını hesaplar
// sonra da integral hesabını yapar
double a[][]=new double[2][n];
a=gauss_legendre_katsayilari(x1,x2,n);
System.out.println("katsayılar matrisi : \n"+Matrix.toStringT(a));
double z=0;
for(int i=0;i<n;i++)
{ z+=a[1][i]*f_xnt.func(a[0][i]);}
return z;
}
  public static void main(String args[]) throws IOException
  {
  //Gauss-Legendre integrali
  f1 b1=new f1();
  System.out.println("integral  : "+integral(b1,-1.0,1.0,10));
}
}
```

değişken nokta sayılı Gauss-Legendre integrasyonu (toplam 10 nokta için, tam çözüm 1.0)

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" integ9
katsayılar matrisi :
  -0.9739065285   0.0666713443
  -0.8650633667   0.1494513492
  -0.6794095683   0.2190863625
  -0.4333953941   0.2692667193
  -0.1488743390   0.2955242247
   0.1488743390   0.2955242247
   0.4333953941   0.2692667193
   0.6794095683   0.2190863625
   0.8650633667   0.1494513492
   0.9739065285   0.0666713443

integral  : 0.9999999999999873

> Terminated with exit code 0.
```

Gauss-Legendre formülünde , genel integral formülündeki

$$I_w(-1,1) \cong \int_{-1}^{1} f(x)w(x)dx \cong \sum_{k=1}^{n} c_k f(x_k)$$

w(x) ağırlık fonksiyonunu 1 olarak almıştık. Şimdi de ağırlık fonksiyonunun birden farklı olduğu bazı formüllere göz atalım. **Gauss-Chebychev** formülünde ağırlık fonksiyonu

$w(x) = \frac{1}{\sqrt{1-x^2}}$ şeklinde tanımlanmıştır. Bu durumda integral formülü

$I_w(-1,1) \cong \int_{-1}^{1} f(x)w(x)dx \cong \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)dx \cong \sum_{k=1}^{n} c_k f(x_k)$ formunu alır.

Buradaki x_k değerleri sabit olarak değişmez, Chebychev polinomlarının kökleridir. Chebychev polinomları

$P_0(x)=1$

$P_1(x)=x$

$P_2(x)=2x^2-1$

$P_n(x)=2xP_{n-1}(x) - P_{n-2}(x)$, n=3,4,5.. olarak verilebilir. Bu polinomun kökleri ise

$x_k = \cos\frac{(k+\frac{1}{2})\pi}{n}$;k=0,1,2,….n-1 olarak alınabilir. Formülümüzdeki c_k katsayıları ise

$c_k = \frac{\pi}{n}$;k=0,1,2,….n-1 olarak hesaplanır.

Gauss-Chebychev formülünde de. x=[-1,1] aralığı z=[a,b] aralığına değişken dönüşümü ile rahatlıkla çevrilebilir.

Örnek problem olarak $I = \int_{-1}^{1} \frac{e^x}{\sqrt{(1-x^2)}} dx$ integralini çözelim

Program 7.3.3 değişken nokta sayılı Gauss-Chebychev integrasyonu

```
 import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class f1 extends f_x
{
  public double func(double x)
  {
  return Math.exp(x);
  }
}

public class integ_gauss_chebychev
{

public static double[][] gauss_chebychev_katsayilari(int n)
{   //Gauss-Chebychev quadratürü katsayılarını hesaplar
    double sgngam=1;
    double a[][]=new double[2][n];
    for(int j = 1; j <=n; j++)
    {int k=j-1;
     a[0][k]=Math.cos((2.0*j-1.0)*Math.PI/(2.0*n));
     a[1][k]=Math.PI/(double)n;
    }
```

```
return a;
}

public static double gauss_chebychev_integral(f_x f_xnt,int n)
{
//n : number of integral katsayilari
// this routine first generates gauss chebychev katsayilari
// for [xa,xb] band
// then calculates gauss chebychev integral
double a[][]=new double[2][n];
a=gauss_chebychev_katsayilari(n);
//System.out.println(Matrix.toString(a));

double toplam=0;
double ck;
double x,y;
double z1=0;
for(int k=0;k<n;k++)
{ toplam+=a[1][k]*f_xnt.func(a[0][k]); }
return toplam;
}

    public static void main(String[] args)
    {
            f1 b1=new f1();
    System.out.println("integral  : "+gauss_chebychev_integral(b1,50));
    }
}
```

Değişken nokta sayılı Gauss-Chebychev integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ_gauss_chebychev
integral  : 3.9774632605064237
> Terminated with exit code 0.
```

Gauss-Jacobi (Mehler) formülü

$$I_w(-1,1) \cong \int_{-1}^{1} f(x)w(x)dx \cong \int_{-1}^{1}(1-x)^{\alpha}(1+x)^{\beta} f(x)dx \cong \sum_{k=0}^{n-1} c_k f(x_k)$$

formundadır. Bu denklemde x_k Jacobi polinomlarının kökleridir. Jacobi polinomları :

$J_n(x;\alpha,\beta) = \sum_{i=0}^{n} A_n x^n$ formundadır. Buradaki katsayılar

$A_n = \frac{1}{2n}\frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+\alpha+\beta+1)}$ şeklinde tanımlanır.

Denklemin katsayıları ise :

$c_k = -\frac{(2n+\alpha+\beta+2)\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{(n+\alpha+\beta+1)\Gamma(n+\alpha+\beta+1)(n+1)!}\frac{2^{\alpha+\beta}}{J_n^{'}(x_k;\alpha,\beta)J_{n+1}(x_k;\alpha,\beta)}$ şeklinde tanımlanır. Buradaki □□gamma fonksiyonudur. Denklemin iki özel haline dikkatinizi çekelim. □=0 ve □=0 Gauss-Legendre formülünü verir. . □=-0.5 ve □=-0.5 Gauss-Chebychev formülünü verir. Görüldüğü gibi Gauss-Jacobi denklemi çok daha genel bir form oluşturur. Örnek problemimizde bir önceki problemde kullandığımız

$I = \int_{-1}^{1} \frac{e^x}{\sqrt{(1-x^2)}} dx$ integralini tekrar Gauss-legendre seti katsayılarını kullanarak çözelim.

Program 7.3.4 değişken nokta sayılı Gauss-Jacobi integrasyonu □=0 ve □=0

```
import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class f1 extends f_x
{
  public double func(double x)
  {
  return Math.exp(x)/Math.sqrt(1-x*x);
  }
}

public class integ_gauss_jacobi
{

    public static double [] lngamma(double x)
    {//logaritmik hata fonksiyonu
    //xx[0] hata fonksiyonunun logaritmasının değerini
    //xx[1] hata fonksiyonunun kendisinin işaretini verir
                double xx[]=new double[2];
    double result;
    double a;
    double b;
    double c;
    double p;
    double q;
    double u;
    double w;
    double z;
    int i;
    double logpi;
    double ls2pi;
    double tmp;
    double sgngam = 1;
    logpi = 1.14472988584940017414;
    ls2pi = 0.91893853320467274178;
    if( x<-34.0 )
    {
       q = -x;
       double yy[]=lngamma(q);
       w = yy[0];
       tmp=yy[1];
       p = Math.floor(q);
       i = (int)Math.round(p);
       if( i%2==0 )
       {
          sgngam = -1;
       }
       else
       {
          sgngam = 1;
       }
       z = q-p;
       if( z>0.5 )
       {
          p = p+1;
          z = p-q;
       }
       z = q*Math.sin(Math.PI*z);
       result = logpi-Math.log(z)-w;
       xx[0]=result;
       xx[1]=sgngam;
       return xx;
```

```
    }
    if( x<13 )
    {
       z = 1;
       p = 0;
       u = x;
       while(u>=3)
       {
          p = p-1;
          u = x+p;
          z = z*u;
       }
       while(u<2)
       {
          z = z/u;
          p = p+1;
          u = x+p;
       }
       if( z<0 )
       {
          sgngam = -1;
          z = -z;
       }
       else
       {
          sgngam = 1;
       }
       if( u==2 )
       {
          result = Math.log(z);
          //return result;
          xx[0]=result;
          xx[1]=sgngam;
          return xx;
       }
       p = p-2;
       x = x+p;
       b = -1378.25152569120859100;
       b = -38801.6315134637840924+x*b;
       b = -331612.992738871184744+x*b;
       b = -1162370.97492762307383+x*b;
       b = -1721737.00820839662146+x*b;
       b = -853555.664245765465627+x*b;
       c = 1;
       c = -351.815701436523470549+x*c;
       c = -17064.2106651881159223+x*c;
       c = -220528.590553854454839+x*c;
       c = -1139334.44367982507207+x*c;
       c = -2532523.07177582951285+x*c;
       c = -2018891.41433532773231+x*c;
       p = x*b/c;
       result = Math.log(z)+p;
       //return result;
       xx[0]=result;
       xx[1]=sgngam;
       return xx;
    }
    q = (x-0.5)*Math.log(x)-x+ls2pi;
    if( x>100000000 )
    {
       result = q;
       //return result;
       xx[0]=result;
       xx[1]=sgngam;
       return xx;
    }
```

```
    p = 1/(x*x);
    if( x>=1000.0 )
    {
       q = q+((7.936507936507936507936507936507936507936*0.0001*p-2.777777777777777777778*0.001)*p+0.08333333333333333333333)/x;
    }
    else
    {
       a = 8.116141674705084503000*0.0001;
       a = -5.950619042843014383240*0.0001+p*a;
       a = 7.936503404577169439450*0.0001+p*a;
       a = -2.777777777300996872050*0.001+p*a;
       a = 8.333333333333319277220*0.01+p*a;
       q = q+a/x;
    }
    xx[0]= q;
    xx[1]=sgngam;
    return xx;
    }
//====================

    public static double[][] gauss_jacobi_coefficients(int n,double alpha,double beta)
    {
       double eps=1.0e-30;
                int i = 0;
       int j = 0;
       double r = 0;
       double r1 = 0;
       double t1 = 0;
       double t2 = 0;
       double t3 = 0;
       double p1 = 0;
       double p2 = 0;
       double p3 = 0;
       double pp = 0;
       double an = 0;
       double bn = 0;
       double a = 0;
       double b = 0;
       double c = 0;
       double tmpsgn = 0;
       double alfbet = 0;
       double temp = 0;
       int its = 0;
       double A[][]=new double[2][n];
       for(i=0; i<=n-1; i++)
       {   if( i==0 )
          {
             an = alpha/n;
             bn = beta/n;
             t1 = (1+alpha)*(2.78/(4+n*n)+0.768*an/n);
             t2 = 1+1.48*an+0.96*bn+0.452*an*an+0.83*an*bn;
             r = (t2-t1)/t2;
          }
          else
          {
             if( i==1 )
             {
                t1 = (4.1+alpha)/((1+alpha)*(1+0.156*alpha));
                t2 = 1+0.06*(n-8)*(1+0.12*alpha)/n;
                t3 = 1+0.012*beta*(1+0.25*Math.abs(alpha))/n;
                r = r-t1*t2*t3*(1-r);
             }
             else
             {
                if( i==2 )
                {
```

```
            t1 = (1.67+0.28*alpha)/(1+0.37*alpha);
            t2 = 1+0.22*(n-8)/n;
            t3 = 1+8*beta/((6.28+beta)*n*n);
            r = r-t1*t2*t3*(A[0][0]-r);
         }
         else
         {
            if( i<n-2 )
            {
               r = 3*A[0][i-1]-3*A[0][i-2]+A[0][i-3];
            }
            else
            {
               if( i==n-2 )
               {
                  t1 = (1+0.235*beta)/(0.766+0.119*beta);
                  t2 = 1/(1+0.639*(n-4)/(1+0.71*(n-4)));
                  t3 = 1/(1+20*alpha/((7.5+alpha)*n*n));
                  r = r+t1*t2*t3*(r-A[0][i-2]);
               }
               else
               {
                  if( i==n-1 )
                  {
                     t1 = (1+0.37*beta)/(1.67+0.28*beta);
                     t2 = 1/(1+0.22*(n-8)/n);
                     t3 = 1/(1+8*alpha/((6.28+alpha)*n*n));
                     r = r+t1*t2*t3*(r-A[0][i-2]);
                  }
               }
            }
         }
      }
   }
   alfbet = alpha+beta;
   do
   {
      temp = 2+alfbet;
      p1 = (alpha-beta+temp*r)*0.5;
      p2 = 1;
      for(j=2; j<=n; j++)
      {
         p3 = p2;
         p2 = p1;
         temp = 2*j+alfbet;
         a = 2*j*(j+alfbet)*(temp-2);
         b = (temp-1)*(alpha*alpha-beta*beta+temp*(temp-2)*r);
         c = 2*(j-1+alpha)*(j-1+beta)*temp;
         p1 = (b*p2-c*p3)/a;
      }
      pp = (n*(alpha-beta-temp*r)*p1+2*(n+alpha)*(n+beta)*p2)/(temp*(1-r*r));
      r1 = r;
      r = r1-p1/pp;
   }
   while( Math.abs(r-r1)>=eps*(1+Math.abs(r))*100 );
   A[0][i] = r;
   double yy[]=lngamma(alpha+n);
   double w1 = yy[0];
   tmpsgn=yy[1];
   yy=lngamma(beta+n);
   double w2 = yy[0];
   tmpsgn=yy[1];
   yy=lngamma(n+1);
   double w3 = yy[0];
   tmpsgn=yy[1];
   yy=lngamma(n+alfbet+1);
```

```
          double w4 = yy[0];
          tmpsgn=yy[1];
          A[1][i] = Math.exp(w1+w2-w3-w4)*temp*Math.pow(2, alfbet)/(pp*p2);
       }
   return A;
   }


public static double gauss_jacobi_integral(f_x f_xnt,int n,double alpha,double beta)
{
//n : number of integral coefficients
// this routine first generates gauss legendre coefficients
// for [x1,x2] band
// then calculates gauss hermite integral
double a[][]=new double[2][n];
a=gauss_jacobi_coefficients(n,alpha,beta);
System.out.println(Matrix.toString(a));
double z=0;
for(int i=0;i<n;i++)
{ z+=a[1][i]*f_xnt.func(a[0][i]);}
return z;
}

     public static void main(String[] args)
     {
               f1 b1=new f1();
     System.out.println(gauss_jacobi_integral(b1,5,0.0,0.0));
     }
}
```

5 nokta sayılı Gauss-Jacobi integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ_gauss_jacobi
      0.906179845938664      0.538469310105683      0.000000000000000      -0.538469310105683      -0.906179845938664
      0.236926885056189      0.478628670499367      0.568888888888889      0.478628670499367      0.236926885056189

3.4865363609790476

> Terminated with exit code 0.
```

Görüldüğü gibi burada Gauss_legendre katsayılarını doğru olarak elde ettik,yalnız sadece 5 katsayı aldığımızdan integral doğruluğunu tam sağlayamadık. Katsayıyı artırarak çözersek :

5 nokta sayılı Gauss-Jacobi integrasyonu $\alpha=0$ ve $\beta=0$

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ_gauss_jacobi
n=20 integral = 3.8463361175531756

> Terminated with exit code 0.
```

50 nokta sayılı Gauss-Jacobi integrasyonu $\alpha=0$ ve $\beta=0$

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ_gauss_jacobi
n=50 integral = 3.9242400049441444

> Terminated with exit code 0.
```

100 nokta sayılı Gauss-Jacobi integrasyonu $\alpha=0$ ve $\beta=0$

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ_gauss_jacobi
n=100 integral = 3.9506992376105177

> Terminated with exit code 0.
```

Fonksiyonu değiştirerek α=-0.5 ve β=-0.5 (Gauss-Chebychev formülü) için çözersek

Program 7.3.4 (değiştirilmiş form) değişken nokta sayılı Gauss-Jacobi integrasyonu α=-0.5 ve β=-0.5

```
class f1 extends f_x
{
  public double func(double x)
  {
  return Math.exp(x);
  }
}
……
System.out.println("n="+n+" integral = "+gauss_jacobi_integral(b1,n,-0.5,-0.5));
……
```

α=-0.5 ve β=-0.5 (Gauss-Chebychev formülü) için çözersek

10 nokta sayılı Gauss-Jacobi integrasyonu α=-0.5 ve β=-0.5

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ_gauss_jacobi
n=10 integral = 3.97746265002148

> Terminated with exit code 0.
```

Görüldüğü gibi Gauss-Jacobi integral formülü çok daha genel bir format oluşturmaktadır.

Gauss-Hermit formülü

$$w(x) = e^{-x^2}$$

$$I_w(-\infty,\infty) \cong \int_{-\infty}^{\infty} f(x)w(x)dx \cong \int_{-\infty}^{\infty} e^{-x^2} f(x)dx \cong \sum_{k=0}^{n-1} c_k f(x_k)$$

formundadır. Bu denklemde x_k Hermit polinomlarının kökleridir. Hermit polinomları :

$P_{-1}(x)=0$

$P_0(x)=1$

$P_1(x)=2x$

$P_2(x)=4x^2-2$

$P_3(x)=8x^3-12x$

$P_n(x)=2xP_{n-1}(x) – 2(n-1)P_{n-2}(x)$, n=1,2,3,4,5.. olarak verilebilir.

Katsayılar,

$c_k = \dfrac{2^{n-1} n! \sqrt{\pi}}{n^2 P_{n-1}(x_k)^2}$ formülünden hesaplanabilir. İlk 7 açılım için katsayı ve kök değerleri tablo 7.4-2 de verilmiştir.

Tablo 7.3-2 Gauss – Hermit integral katsayıları

N	x_k	c_k
2	+/-0.707167811	0.886226926
3	0	1.181635901
	+/- 1.224744871391589	0.295408975
4	+/- 0.524647623275290	0.80491409
	+/- 1.650680123885784	0.081312835
5	0	0.945308721
	+/-0.9585724646	0.393619323
	+/-2.0201828705	0.019953242
6	+/- 2.350604973674492	0.00453001
	+/- 1.335849074013697	0.15706732
	+/- 0.436077411927616	0.724629595
7	+/- 2.651961356835233	0.000971781

	+/- 1.673551628767471	**0.054515583**
	+/- 0.816287882858965	**0.425607253**
	0	**0.810264618**

Bir örnek programla Gauss_Hermite integral denkleminin örneğini verelim :

Program 7.3.5 Gauss-Hermite integrasyonu

```
İmport java.util.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class f1 extends f_x
{
  public double func(double x)
  {
  return 1/(1+x*x);
  }
}

public class integ_gauss_hermite
{

public static double[][] gauss_hermite_coefficients(int n)
{
   double a[][]=new double[2][n];
   int i;
   int j;
   double r=0;
   double r1;
   double p1;
   double p2;
   double p3;
   double dp3;
   double pipm4;
   double eps=1e-10;
   pipm4 = Math.pow(Math.PI, -0.25);
   for(i = 0; i <= (n+1)/2-1; i++)
   {
      if( i==0 )
      { r =Math.sqrt((double)(2*n+1))-1.85575*Math.pow((double)(2*n+1), -1.0/6.0);}
      else if( i==1 )
         { r = r-1.14*Math.pow((double)(n), 0.426)/r; }
      else if( i==2 )
            { r = 1.86*r-0.86*a[0][0];}
      else if( i==3 )
            { r = 1.91*r-0.91*a[0][1];}
      else
            {r = 2*r-a[0][i-2];}
   do {
         p2 = 0;
         p3 = pipm4;
         for(j = 0; j <= n-1; j++)
         {   p1 = p2;
            p2 = p3;
            p3 = p2*r*Math.sqrt(2.0/(double)(j+1))-p1*Math.sqrt((double)(j)/(double)(j+1));
         }
         dp3 = Math.sqrt((double)(2.0*j))*p2;
         r1 = r;
         r = r-p3/dp3;
      }
      while(Math.abs(r-r1)>=eps*(1+Math.abs(r))*100.0);
   a[0][i] = r;
   a[1][i] = 2.0/(dp3*dp3);
```

```
    a[0][n-1-i] = -a[0][i];
    a[1][n-1-i] = a[1][i];
   }
  return a;
}

public static double gauss_hermite_integral(f_x f_xnt,int n)
{
//n : number of integral coefficients
// this routine first generates gauss hermite coefficients
// for [x1,x2] band
// then calculates gauss hermite integral
double a[][]=new double[2][n];
a=gauss_hermite_coefficients(n);
System.out.println(Matrix.toString(a));
double z=0;
for(int i=0;i<n;i++)
{ z+=a[1][i]*f_xnt.func(a[0][i])*Math.exp(a[0][i]*a[0][i]);}
return z;
}

    public static void main(String[] args)
    {
            f1 b1=new f1();
    System.out.println("integral  : "+gauss_hermite_integral(b1,20));
    }
}
```

Gauss-Hermite integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ_gauss_hermite
integral  : 2.9965945890817665
> Terminated with exit code 0.
```

Gauss-Laguerre formülü

$$w(x) = x^{\alpha}e^{-x}$$

$$I_w(0,\infty) \cong \int_0^{\infty} f(x)w(x)dx \cong \int_0^{\infty} x^{\alpha}e^{-x} f(x)dx \cong \sum_{k=0}^{n-1} c_k f(x_k)$$

formundadır. Bu denklemde x_k Laguerre polinomlarının kökleridir. Laguerre polinomları :

$P_{-1}(x)=0$

$P_0(x)=1$

$(n+1)P^{\alpha}_{n+1}(x)=(-x+2*n+\alpha+1)P^{\alpha}_n(x) - (n+\alpha)P^{\alpha}_{n-1}(x)$, n=1,2,3,4,5.. olarak verilebilir.

Katsayılar,

$$c_k = \frac{1}{x_k[\frac{dP_n(x_k)}{dx_k}]^2}$$ formülünden hesaplanabilir. İlk 5 açılım için katsayı değerleri tablo 7.4-3(α=0), tablo 7.4-4(α=1) ve tablo 7.4-5(α=2) verilmiştir.

Gauss-Laguerre integral formülünde α=0 için integral alt şartını değiştirebiliriz. Bu durumda denklem

$$I_w(a,\infty) \cong \int_a^{\infty} f(x)w(x)dx \cong \int_a^{\infty} e^{-(x+a)} f(x+a)dx \cong e^{-a}\sum_{k=0}^{n-1} c_k f(x_k+a)$$

halini alır.

Tablo 7.3-3 Gauss – Laquerre integral katsayıları (α=0)

2	**0.585786437626905**	**0.853553390592534**

	3.414213562373090	**0.146446609406723**
3	**0.415774556783479**	**0.711093009915089**
	2.294280360279040	**0.278517733568676**
	6.289945082937470	**0.010389256497354**
4	**0.322547689619392**	**0.603154104288259**
	1.745761101158340	**0.357418692437802**
	4.536620296921120	**0.038887908507759**
	9.395070912301130	**0.000539294705556**
5	**0.263560319718141**	**0.521755610467045**
	1.413403059106510	**0.398666821407258**
	3.596425771040720	**0.075942449681708**
	7.085810005858830	**0.003611758663878**
	12.640800844275700	**0.000023369972386**

Tablo 7.3-4 Gauss – Laquerre integral katsayıları (α=1)

N	x_k	c_k
2	**1.267949192431120**	**0.788675133123411**
	4.732050807568870	**0.211324865405185**
3	**0.935822227524088**	**0.588681472855445**
	3.305407289332270	**0.391216059222280**
	7.758770483143630	**0.020102459732679**
4	**0.743291927981432**	**0.446870579513482**
	2.571635007646270	**0.477635774492073**
	5.731178751689090	**0.074177784726293**
	10.953894312683100	**0.001315849683447**
5	**0.617030853278271**	**0.348014523429866**
	2.112965958578520	**0.502280674138866**
	4.610833151017530	**0.140915919102187**
	8.399066971204840	**0.008719893025972**

Tablo 7.3-5 Gauss – Laquerre integral katsayıları (α=2)

N	x_k	c_k
2	**2.000000000000000**	**1.500000000000000**
	6.000000000000000	**0.500000000000000**
3	**1.517387080677410**	**1.037494961490390**
	4.311583133719520	**0.905750004703039**
	9.171029785603060	**0.056755033772202**
4	**1.226763263500300**	**0.725524997698604**
	3.412507358696940	**1.063424292391060**
	6.902692605851610	**0.206696130999709**
	12.458036771951100	**0.004354579188558**
5	**1.031109144093380**	**0.520917396835042**
	2.837212823953820	**1.066705933159050**
	5.620294272598700	**0.383549720007093**
	9.682909837664020	**0.028564233510280**
	15.828473921690000	**0.000262712802303**

Gauss-Laguerre integral formülünü bir program ve örnek problemle görelim.

$I = \int_0^{\infty} \frac{x}{e^x + 1} dx$ integralinin türevini hesaplayacağız. İntegralin değeri I= 0.822467 dir.

Program 7.3-6 Gauss-Laguerre integrasyonu

```
İmport java.util.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class f1 extends f_x
```

```
{
  public double func(double x)
  {
  return x/(Math.exp(x)+1.0);
  }
}

public class integ_gauss_laguerre
{

                public static double [] lngamma(double x)
    {//logaritmik hata fonksiyonu
    //xx[0] hata fonksiyonunun logaritmasının değerini
    //xx[1] hata fonksiyonunun kendisinin işaretini verir
                double xx[]=new double[2];
    double result;
    double a;
    double b;
    double c;
    double p;
    double q;
    double u;
    double w;
    double z;
    int i;
    double logpi;
    double ls2pi;
    double tmp;
    double sgngam = 1;
    logpi = 1.14472988584940017414;
    ls2pi = 0.91893853320467274178;
    if( x<-34.0 )
    {
       q = -x;
       double yy[]=lngamma(q);
       w = yy[0];
       tmp=yy[1];
       p = Math.floor(q);
       i = (int)Math.round(p);
       if( i%2==0 )
       {
          sgngam = -1;
       }
       else
       {
          sgngam = 1;
       }
       z = q-p;
       if( z>0.5 )
       {
          p = p+1;
          z = p-q;
       }
       z = q*Math.sin(Math.PI*z);
       result = logpi-Math.log(z)-w;
       xx[0]=result;
       xx[1]=sgngam;
       return xx;
    }
    if( x<13 )
    {
       z = 1;
       p = 0;
       u = x;
       while(u>=3)
       {
```

```
        p = p-1;
        u = x+p;
        z = z*u;
    }
    while(u<2)
    {
        z = z/u;
        p = p+1;
        u = x+p;
    }
    if( z<0 )
    {
        sgngam = -1;
        z = -z;
    }
    else
    {
        sgngam = 1;
    }
    if( u==2 )
    {
        result = Math.log(z);
        //return result;
        xx[0]=result;
        xx[1]=sgngam;
        return xx;
    }
    p = p-2;
    x = x+p;
    b = -1378.25152569120859100;
    b = -38801.6315134637840924+x*b;
    b = -331612.992738871184744+x*b;
    b = -1162370.97492762307383+x*b;
    b = -1721737.00820839662146+x*b;
    b = -853555.664245765465627+x*b;
    c = 1;
    c = -351.81570143652347O549+x*c;
    c = -17064.2106651881159223+x*c;
    c = -220528.590553854454839+x*c;
    c = -1139334.44367982507207+x*c;
    c = -2532523.07177582951285+x*c;
    c = -2018891.41433532773231+x*c;
    p = x*b/c;
    result = Math.log(z)+p;
    //return result;
    xx[0]=result;
    xx[1]=sgngam;
    return xx;
}
q = (x-0.5)*Math.log(x)-x+ls2pi;
if( x>100000000 )
{
    result = q;
    //return result;
    xx[0]=result;
    xx[1]=sgngam;
    return xx;
}
p = 1/(x*x);
if( x>=1000.0 )
{
    q = q+((7.9365079365079365079365*0.0001*p-2.777777777777777777778*0.001)*p+0.083333333333333333333)/x;
}
else
{
    a = 8.116141674705084503O0*0.0001;
```

```
        a = -5.950619042843014383 24*0.0001+p*a;
        a = 7.936503404577169439 45*0.0001+p*a;
        a = -2.777777777309968720 5*0.001+p*a;
        a = 8.333333333333192772 2*0.01+p*a;
        q = q+a/x;
    }
    xx[0]= q;
    xx[1]=sgngam;
    return xx;
    }

public static double[][] gauss_laguerre_coefficients(int n,double alpha)
{   //Gauss-Laguerre quadratürü katsayılarını hesaplar
    double sgngam=1;
    double a[][]=new double[2][n];
    int i;
    int j;
    double r=0;
    double r1;
    double p1;
    double p2;
    double p3;
    double dp3;
    double tsg;
    double eps=1.0e-10;
    for(i = 0; i <= n-1; i++)
    {
        if( i==0 )
        {
            r = (1.0+alpha)*(3.0+0.92*alpha)/(1+2.4*n+1.8*alpha);
        }
        else
        {
            if( i==1 )
            {r = r+(15.0+6.25*alpha)/(1.0+0.9*alpha+2.5*n);}
            else
            {r = r+((1.0+2.55*(i-1))/(1.9*(i-1))+1.26*(i-1)*alpha/(1+3.5*(i-1)))/(1.0+0.3*alpha)*(r-a[0][i-2]);}
        }
        do
        {
            p2 = 0;
            p3 = 1;
            for(j = 0; j <= n-1; j++)
            {
                p1 = p2;
                p2 = p3;
                p3 = ((-r+2*j+alpha+1)*p2-(j+alpha)*p1)/(j+1);
            }
            dp3 = (n*p3-(n+alpha)*p2)/r;
            r1 = r;
            r = r-p3/dp3;
        }
        while(Math.abs(r-r1)>=eps*(1+Math.abs(r))*100);
        a[0][i] = r;
        double lng[]=lngamma((double)n);
        double lng1[]=lngamma((double)(alpha+n));
        a[1][i] = -Math.exp(lng1[0]-lng[0])/(dp3*n*p2);
        tsg=lng[1];
    }
return a;
}

public static double gauss_laquerre_integral(f_x f_xnt,double xa,int n,double alpha)
{
//n : number of integral coefficients
// this routine first generates gauss legendre coefficients
```

```
// for [x1,x2] band
// then calculates gauss hermite integral
double a[][]=new double[2][n];
a=gauss_laguerre_coefficients(n,alpha);
//System.out.println(Matrix.toString(a));
double z=0;
for(int i=0;i<n;i++)
{ z+=a[1][i]*f_xnt.func(a[0][i]+xa)*Math.exp(a[0][i])*Math.pow(a[0][i],-alpha);}
return z;
}

public static double gauss_laquerre_integral(f_x f_xnt,double xa,int n)
{
double alpha=1.0;
double a[][]=new double[2][n];
a=gauss_laguerre_coefficients(n,alpha);
//System.out.println(Matrix.toString(a));
double z=0;
for(int i=0;i<n;i++)
{ z+=a[1][i]*f_xnt.func(a[0][i]+xa)*Math.exp(a[0][i])/a[0][i];}
return z;
}

    public static void main(String[] args)
    {
            f1 b1=new f1();
    System.out.println("integral  : "+gauss_laquerre_integral(b1,0,30)+" tam = "+0.822467);
    }
}
```

Gauss-Laguerre integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ_gauss_laguerre
integral  : 0.8224670275951408 tam = 0.822467
> Terminated with exit code 0.
```

7.4 Ayarlanabilir hata miktarlı integrasyon formülleri

[a,b] bölgesinde integral işlemi tanımlayalım. İntegrali alınabilecel bir gerçek fonksiyon için integral

$$I = \int_a^b f(x)dx$$

şeklinde yazılabilir. Eğer integral için bir alt bölge tanımlarsak, örneğin $m = \frac{(a-b)}{2}$ tanımını yaparsak,

integralimizi iki alt integrale bölerek tanımlıyabiliriz.

$I = \int_a^m f(x)dx + \int_m^b f(x)dx$ Bu durumda genel bir kavram olarak hata miktarının azalacağını söyleyebiliriz.

Aynı işlemi her integrali de alt integrallere bölerek de sürdürmemiz mümkündür. Burada önemli olam kavram integral hata limiti belli bir değerin altına indiğinde prosesi durdurmaktır. Bu altbölümümüzde ayarlanabilir hata miktarlı çeşitli integral yaklaşımlarını inceleyeceğiz.

7.4.1 Ayarlanabilir Simpson 1/3 integrali

[a,b] aralığında simpson 1/3 integral denklemini tekrar yazalım.

$$I(a,b) = I(x_0, x_4) = \frac{h}{3}[f(x_0) + 4f(x_2) + f(x_4)]$$

$$\int_a^b f(x)dx = I(x_0, x_4) - h^5 \frac{f^{(4)}(d_1)}{90}$$

Buradaki h =(b-a)/2, x_0=a,x_2=(a+b)/2,x_4=b dir d_1 a ile b arasında yer alan bir sayıdır ve $h^5 \frac{f^{(4)}(d_1)}{90}$ denklemdeki hata miktarını verir. hata denklemindeki $f^{(4)}$ fonksiyonun 4. türevidir.
Şimdi integrali hesapladığımız bölgeyi iki alt bölgeye bölelim. Bu durumda denklem

$$I(x_0, x_2) = \frac{h}{6}[f(x_0) + 4f(x_1) + f(x_2)]$$

$$I(x_2, x_4) = \frac{h}{6}[f(x_2) + 4f(x_3) + f(x_4)]$$

$$\int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx = I(x_0, x_2) + I(x_2, x_4) - \frac{h^5}{16}\frac{f^{(4)}(d_2)}{90}$$

halini alacaktır. Buradaki x_1=(a+b)/4, x_3=3(a+b)/4 tür. d_2 değeri yine a ile b arasında bir değerdir.

Eğer $f^{(4)}(d_1) \approx f^{(4)}(d_2)$ kabul edersek

$h^5 \frac{f^{(4)}(d_1)}{90} \approx \frac{16}{15}[I(x_0, x_2) + I(x_2, x_4) - I(x_0, x_4)]$ olur.

$$\frac{h^5}{16}\frac{f^{(4)}(d_1)}{90} \approx \frac{1}{15}[I(x_0, x_2) + I(x_2, x_4) - I(x_0, x_4)]$$

Bu denklemi integral denklemine koyarsak

$$\left|\int_{x_0}^{x_4} f(x)dx - I(x_0, x_2) - I(x_2, x_4)\right| \approx \frac{1}{15}\left|I(x_0, x_2) + I(x_2, x_4) - I(x_0, x_4)\right|$$

Bu durumda □□□□hatası için

$$\frac{1}{15}\left|I(x_0, x_2) + I(x_2, x_4) - I(x_0, x_4)\right| < \varepsilon$$

yazılabilir. Eğer bu şart gerçekleşiyorsa

$\int_{x_0}^{x_4} f(x)dx \approx I(x_0, x_2) + I(x_2, x_4)$ yazılabilir. Eğer bu şart gerçekleşmiyorsa, integrasyon bölgesi ikiye bölünerek her bölge tekrar kontrol edilir, bölme işlemi istenilen hata şartı gerçekleşene kadar devam ederiz.

Program 7.4-1 de bu ayarlanabilir hata miktarlı simpson 1/3 integrali verilmiştir.

Program 7.4.1 Ayarlanabilir hata miktarlı Simson 1/3 integrasyonu

```
//=====================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;

class f1 extends f_x
{
  double func(double x)
  {return x*Math.sqrt(x);}
```

```
}


class integ10
{
public static double adaptive_simpson_integral(f_x ff,double a,double b)
{return adaptive_simpson_integral(ff,a,b,1.0e-20,30);}

public static double adaptive_simpson_integral(f_x ff,double a,double b,double eps)
{return adaptive_simpson_integral(ff,a,b,eps,30);}

public static double adaptive_simpson_integral(f_x ff,double a,double b,int MAX_ITERATION)
{return adaptive_simpson_integral(ff,a,b,1.0e-12,MAX_ITERATION);}

public static double adaptive_simpson_integral(f_x ff,double a,double b,double eps,int MAX_ITERATION)
{   int i;
   double h=(b-a);
   double h2=h/4.0;
            double x[]=new double[5];
   double f[]=new double[5];
   for(i=0;i<5;i+=2) {x[i]=a+i*h2;f[i]=ff.func(x[i]);}
            double sum=h*(f[0]+4.0*f[2]+f[4])/6.0;
            double Q=0;
   Q=simpson1_3(ff,x,f,sum,eps,0,MAX_ITERATION,Q);
   return Q;
}

public static double simpson1_3(f_x ff,double x[],double f[],double sum,double eps,int k,int MAX_ITERATION,double Q)
{       //Newton-cotes integral 3 points
       //adaptive Simpson 1/3 rule
       double h=(x[4]-x[0]);
       double h1=h/2.0;
       double h2=h/4.0;
       int p,i;
       double x1[]=new double[5];
       double f1[]=new double[5];
       double x2[]=new double[5];
       double f2[]=new double[5];
       k++;
       for(i=1;i<5;i+=2) {x[i]=x[0]+i*h2;f[i]=ff.func(x[i]);}
       double sum1=h1*(f[0]+4.0*f[1]+f[2])/6.0;
       double sum2=h1*(f[2]+4.0*f[3]+f[4])/6.0;
       x1[0]=x[0];f1[0]=f[0];
       x1[2]=x[1];f1[2]=f[1];
       x1[4]=x[2];f1[4]=f[2];
       x2[0]=x[2];f2[0]=f[2];
       x2[2]=x[3];f2[2]=f[3];
       x2[4]=x[4];f2[4]=f[4];
       if(1.0/15.0*Math.abs(sum1+sum2-sum)<eps || k>MAX_ITERATION) Q+=sum1+sum2;
       else Q+=simpson1_3(ff,x1,f1,sum1,eps,k,MAX_ITERATION,Q)+simpson1_3(ff,x2,f2,sum2,eps,k,MAX_ITERATION,Q);
       return Q;
}

  public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  System.out.println("integral of class        f1 : "+adaptive_simpson_integral(b1,0.0,1.0));
}
}
```

Ayarlanabilir hata miktarlı Simson 1/3 integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ10
integral of class        f1 : 0.4
```

```
> Terminated with exit code 0.
```

Aynı programın grafik üzerinde hesaplama noktalarını gösteren bir versiyonunu daha verelim.

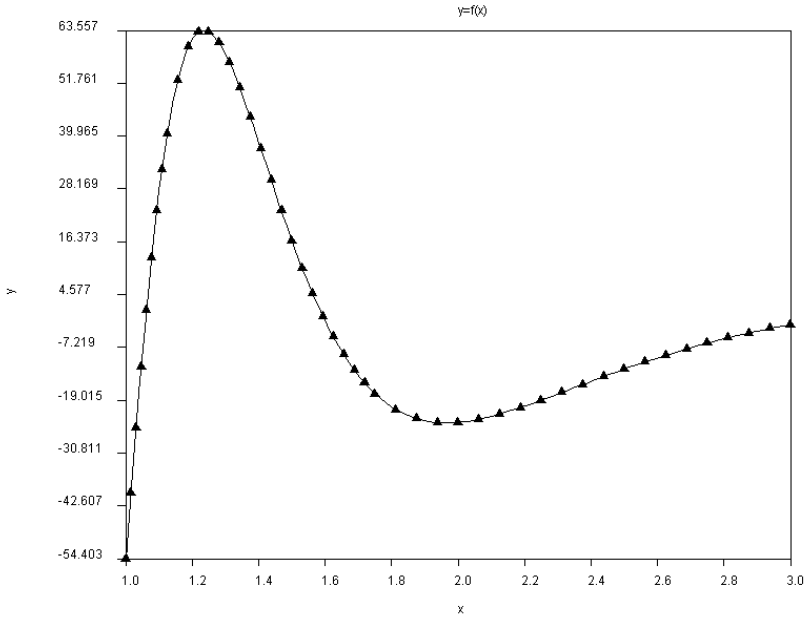
Program 7.4.1A Ayarlanabilir hata miktarlı Simson 1/3 integrasyonu (grafik çıktılı versiyonu)

```
//=======================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =======================================================
import java.io.*;

class f1 extends f_x
{
  double func(double x)
  {//return 100.0/(x*x)*Math.sin(10.0/x);
  return Math.sqrt(x);
  }
}


class integ10
{
public static double adaptive_simpson_integral(f_x ff,double a,double b)
{return adaptive_simpson_integral(ff,a,b,1.0e-20,30);}

public static double adaptive_simpson_integral(f_x ff,double a,double b,double eps)
{return adaptive_simpson_integral(ff,a,b,eps,30);}

public static double adaptive_simpson_integral(f_x ff,double a,double b,int MAX_ITERATION)
{return adaptive_simpson_integral(ff,a,b,1.0e-12,MAX_ITERATION);}

public static double adaptive_simpson_integral(f_x ff,double a,double b,double eps,int MAX_ITERATION)
{   int i;
            Plot pp=new Plot(ff,a,b,100);
   double h=(b-a);
   double h2=h/4.0;
            double x[]=new double[5];
   double f[]=new double[5];
   for(i=0;i<5;i+=2) {x[i]=a+i*h2;f[i]=ff.func(x[i]);}
            double sum=h*(f[0]+4.0*f[2]+f[4])/6.0;
            double Q=0;
   Q=simpson1_3(ff,x,f,sum,eps,0,MAX_ITERATION,Q);
   return Q;
}

public static double simpson1_3(f_x ff,double x[],double f[],double sum,double eps,int k,int MAX_ITERATION,double Q)
{
      //adaptive Simpson 1/3 rule
      double h=(x[4]-x[0]);
      double h1=h/2.0;
      double h2=h/4.0;
      int p,i;
      double x1[]=new double[5];
      double f1[]=new double[5];
      double x2[]=new double[5];
      double f2[]=new double[5];
      k++;
      for(i=1;i<5;i+=2) {x[i]=x[0]+i*h2;f[i]=ff.func(x[i]);}
      double sum1=h1*(f[0]+4.0*f[1]+f[2])/6.0;
      double sum2=h1*(f[2]+4.0*f[3]+f[4])/6.0;
      x1[0]=x[0];f1[0]=f[0];
      x1[2]=x[1];f1[2]=f[1];
```

```
        x1[4]=x[2];f1[4]=f[2];
        x2[0]=x[2];f2[0]=f[2];
        x2[2]=x[3];f2[2]=f[3];
        x2[4]=x[4];f2[4]=f[4];
        if(k>MAX_ITERATION) {System.out.println("Maksimum iterasyon sayısına ulaşıldı"); Q+=sum1+sum2;}
        else if(1.0/15.0*Math.abs(sum1+sum2-sum)<eps ) Q+=sum1+sum2;
        else Q+=simpson1_3(ff,x1,f1,sum1,eps,k,MAX_ITERATION,Q)+simpson1_3(ff,x2,f2,sum2,eps,k,MAX_ITERATION,Q);
        return Q;
}

  public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  System.out.println("integral of class        f1 : "+adaptive_simpson_integral(b1,0.0,1.0,1.0e-12,10));
  }
}
```

Ayarlanabilir hata miktarlı Simson 1/3 integrasyonu (grafik çıktılı versiyonu)

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ10A
integral of class        f1 : -1.425921623751469
```



(bolum7_008.jpg, **Ayarlanabilir hata miktarlı Simson 1/3 integrasyonu grafik çıktısı**)

7.4.2 Ayarlanabilir Gauss-Legendre integrali

Gauss-Legendre denklemi için ayarlanabilir hata miktarlı versiyonu altta verilmiştir. Buradaki temel fikir Gauss Legendre polinom formülünün katsayılarını birer birer arttırmak ve iki katsayı arasındaki hata miktarı istenilen düzeyin altına indiğinde iterasyon prosesini durdurmaktan ibarettir.

Program 7.4.2 Ayarlanabilir hata miktarlı Gauss-Legendre integrasyonu

```
///=====================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;
```

```
class f1 extends f_x
{
  public double func(double x)
  {return x*Math.sqrt(x);}
}


class integ13
{
public static double[][] gauss_legendre_coefficients(double x1,double x2,int n)
{
        //calculates legendre gauss-coefficients as coefficients of the integral
        //for n terms
        double EPS=3.0e-15;
        int m,j,i;
        double z1,z,xm,xl,pp,p3,p2,p1;
        double a[][]=new double[2][n];
        m=(n+1)/2;
        xm=0.5*(x2+x1);
        xl=0.5*(x2-x1);
        for (i=1;i<=m;i++)  {
                z=Math.cos(Math.PI*((i-0.25)/(n+0.5)));
                do {
                        p1=1.0;
                        p2=0.0;
                        for (j=1;j<=n;j++) {
                                p3=p2;
                                p2=p1;
                                p1=((2.0*j-1.0)*z*p2-(j-1.0)*p3)/j;
                        }
                        pp=n*(z*p1-p2)/(z*z-1.0);
                        z1=z;
                        z=z1-p1/pp;
                } while (Math.abs(z-z1) > EPS);
                a[0][i-1]=xm-xl*z;
                a[0][n-i]=xm+xl*z;
                a[1][i-1]=2.0*xl/((1.0-z*z)*pp*pp);
                a[1][n-i]=a[1][i-1];
                        }
return a;
}

public static double gauss_legendre_integral(f_x f,double x1,double x2,int n)
{
//n : number of integral coefficients
// this routine first generates gauss legendre coefficients
// for [x1,x2] band
// then calculates gauss legendre integral
double a[][]=new double[2][n];
a=gauss_legendre_coefficients(x1,x2,n);
//Text.print(Text.T(a));
//System.out.println(Matrix.toString(a));
double z=0;
for(int i=0;i<n;i++)
{ z+=a[1][i]*f.func(a[0][i]);}
return z;
}

public static double adaptive_gauss_legendre_integral(f_x f,double x1,double x2)
{return adaptive_gauss_legendre_integral(f,x1,x2,1.0e-12,100);}

public static double adaptive_gauss_legendre_integral(f_x f,double x1,double x2,double eps)
{return adaptive_gauss_legendre_integral(f,x1,x2,eps,100);}

public static double adaptive_gauss_legendre_integral(f_x f,double x1,double x2,double eps,double MAXITER)
```

```
{
int n=1;
double ans1=gauss_legendre_integral(f,x1,x2,n);
double ans2;
do
{ans2=ans1;n++;ans1=gauss_legendre_integral(f,x1,x2,n);}
while(Math.abs(ans2-ans1)>eps && n<MAXITER);
if(n==MAXITER) System.out.println("istenilen hata seviyesi tutturulamadı n="+n );
return ans2;
}

  public static void main(String args[]) throws IOException
  {
  //Gauss-Legendre integrali
  f1 b1=new f1();
  System.out.println("integral  : "+adaptive_gauss_legendre_integral(b1,0.0,1.0));
}
}
```

Ayarlanabilir hata miktarlı Gauss-Legendre integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ13
integral  : 0.3999999999829868
> Terminated with exit code 0.
```

7.4.3 Gauss-Kronrod integral formülü

$$I_w(-1,1) \cong \int_{-1}^{1} f(x)dx \cong \sum_{k=1}^{n} c_k f(x_k^G) + \sum_{k=1}^{n+1} c_{n+k} f(x_k^K)$$

Gauss-Kronrod integrasyon formülü integral hesabında iki toplama terimi kullanır, Birinci terim standart gauss formülünden oluşur. İkinci terimde ise katsayılar gauss katsayılarının aynı olarak alınır, ancak kök değerleri değişiktir. Örnek problemimizde sabit kök değerleri kullanacağız n = 15 için (+ ve – değerlerle 30 terim) programımız ilk terim olarak Gauss-Legendre terimini, ikinci terim olarak Kronroad terimini kullanmaktadır. Hassasiyeti artırmak için her iterasyondan sonra iterasyon bölgesi yarıya bölünerek iterasyolara devam edilir. Gauss-Kronrod integral formülü günümüzde bir çok modern sayısal analiz paketi tarafından tercih edilen integrasyon yöntemi olarak kullanılmaktadır.

Program 7.4.3 Ayarlanabilir hata miktarlı Gauss-Kronrod integrasyonu

```
class f1 extends f_x
{
  double func(double x)
  {return x*Math.sqrt(x);}
}

public class adaptive_Gauss_Kronrod_integral
{
   public static double gauss_konrad_integral(f_x ff,double a,double b,double eps,int n)
   {
                //eps hata miktarı
      int Aused=0;
                boolean sonuc;
      double integral=0;
      double A[][];
      int Aw = 0;
      double toplamhata = 0;
      double[] c;
      double[] xg;
      double[] xk;
      int nn = 0;
      int ng = 0;
```

```
int i = 0;
int j = 0;
int h = 0;
double v = 0;
double k1 = 0;
double k2 = 0;
double intg = 0;
double intk = 0;
double ta = 0;
double tb = 0;
Aw = 4;
A = new double[n-1+1][Aw-1+1];
nn = 61;
ng = 15;
c = new double[nn];
xk = new double[nn];
xg = new double[nn];
//Gauss katsayıları
xg[0] = 0.00796819249616660561546588347467 4;
xg[1] = 0.01846646831109095914230213191204 7;
xg[2] = 0.02878470788332336934971917961129 2;
xg[3] = 0.03879919256962704959680193644634 8;
xg[4] = 0.04840267283059405290293814042280 8;
xg[5] = 0.05749315621761906648172168940205 6;
xg[6] = 0.06597422988218049512812851511596 2;
xg[7] = 0.07375597473770520626824385002219 1;
xg[8] = 0.08075589522942021535469493846053 0;
xg[9] = 0.08689978720108297980238753071512 6;
xg[10] = 0.09212252223778612871763270708761 9;
xg[11] = 0.09636873717464425963946862635181 0;
xg[12] = 0.09959342058679526706278028210356 9;
xg[13] = 0.10176238974840550459642895216855 4;
xg[14] = 0.10285265289355884034128563670541 5;
c[0] = 0.99948441005049063757132589570581 1;
c[1] = 0.99689348407464954027163005091869 5;
c[2] = 0.99163099687040459485862836610948 6;
c[3] = 0.98366812327974720997003258160566 3;
c[4] = 0.97311632250112626837469386842370 7;
c[5] = 0.96002186496830751221687102558179 8;
c[6] = 0.94437444474855997941583132403743 9;
c[7] = 0.92620004742927432587932427708047 4;
c[8] = 0.90557330769990779854652255892595 8;
c[9] = 0.88256053579205268154311646253022 6;
c[10] = 0.85720523354606109895865851065894 4;
c[11] = 0.82956576238276839744289811973250 2;
c[12] = 0.79972783582183908301366894232268 3;
c[13] = 0.76777743210482619491797734097450 3;
c[14] = 0.73379006245322680472617113136952 8;
c[15] = 0.69785049479331579693229238802664 0;
c[16] = 0.66006106412662696137005366814927 1;
c[17] = 0.62052618298924286114047755643118 9;
c[18] = 0.57934523582636169175602493217254 0;
c[19] = 0.53662414814201989926416979331107 3;
c[20] = 0.49248046786177857499369306120770 9;
c[21] = 0.44703376953808917678060990032285 4;
c[22] = 0.40040125483039439253547621154266 1;
c[23] = 0.35270472553087811347103720708937 4;
c[24] = 0.30407320227362507737267710719925 7;
c[25] = 0.25463692616788984643980512981780 5;
c[26] = 0.20452511668230989143895767100202 5;
c[27] = 0.15386991360858354696379467274325 6;
c[28] = 0.10280693796673703014709675131800 1;
c[29] = 0.05147184255531769583302521316672 3;
c[30] = 0.00000000000000000000000000000000 0;
//Kronrod katsayıları
xk[0] = 0.00138901369867700762455159122676 0;
```

```
xk[1] = 0.0038904611270998840512672018445 16;
xk[2] = 0.0066307039159312921733198263697 50;
xk[3] = 0.0092732796595177634284411468920 24;
xk[4] = 0.0118230152534963417422328988532 51;
xk[5] = 0.0143697295070458048124514324435 80;
xk[6] = 0.0169208891890532726275722894203 22;
xk[7] = 0.0194141411939423811734089510501 28;
xk[8] = 0.0218280358216091922971674857383 39;
xk[9] = 0.0241911620780806013656863707252 32;
xk[10] = 0.0265099548823331016106017093350 75;
xk[11] = 0.0287540487650412928439787853543 34;
xk[12] = 0.0309072575623877624728842529430 92;
xk[13] = 0.0329814470574837260318141910168 54;
xk[14] = 0.0349793380280600241374996707314 68;
xk[15] = 0.0368823646518212292239110656171 36;
xk[16] = 0.0386789456247275929503486515322 81;
xk[17] = 0.0403745389515359591119952797524 68;
xk[18] = 0.0419698102151642461471475412859 70;
xk[19] = 0.0434525397013560693168317281170 73;
xk[20] = 0.0448148001331626631923555516167 23;
xk[21] = 0.0460592382710069881162717355593 74;
xk[22] = 0.0471855465692991539452614781810 99;
xk[23] = 0.0481858617570871291407794922983 05;
xk[24] = 0.0490554345550297788875281653672 38;
xk[25] = 0.0497956834270742063578115693799 42;
xk[26] = 0.0504059214027823468408930856535 85;
xk[27] = 0.0508817958987496064922974730498 05;
xk[28] = 0.0512215478492587721706562826049 44;
xk[29] = 0.0514261285374590259338628792157 81;
xk[30] = 0.0514947294294515675583404336470 99;
for(i=nn-1; i>=nn/2; i--)
{c[i] = -c[nn-1-i];}
for(i=nn-1; i>=nn/2; i--)
{ xk[i] = xk[nn-1-i]; }
for(i=ng-1; i>=0; i--)
{ xg[nn-2-2*i] = xg[i];
  xg[1+2*i] = xg[i];
}
for(i=0; i<=nn/2; i++)
{ xg[2*i] = 0; }
k1 = 0.5*(b-a);
k2 = 0.5*(b+a);
intg = 0;
intk = 0;
for(i=0; i<=nn-1; i++)
{ v = ff.func(k1*c[i]+k2);
  intk = intk+v*xk[i];
  if( i%2==1 )
  {intg = intg+v*xg[i];}
}
intk = intk*(b-a)*0.5;
intg = intg*(b-a)*0.5;
A[0][0] = Math.abs(intg-intk);
A[0][1] = intk;
A[0][2] = a;
A[0][3] = b;
toplamhata = A[0][0];
if( toplamhata<eps )
{
   sonuc = true;
   integral = intk;
   Aused = 1;
   return integral;
}
Aused = 1;
for(h=1; h<=n-1; h++)
```

```
    {
      Aused = h+1;
      gir(A, h, Aw);
      toplamhata = toplamhata-A[h-1][0];
      ta = A[h-1][2];
      tb = A[h-1][3];
      A[h-1][2] = ta;
      A[h-1][3] = 0.5*(ta+tb);
      A[h][2] = 0.5*(ta+tb);
      A[h][3] = tb;
      for(j=h-1; j<=h; j++)
      {
        k1 = 0.5*(A[j][3]-A[j][2]);
        k2 = 0.5*(A[j][3]+A[j][2]);
        intg = 0;
        intk = 0;
        for(i=0; i<=nn-1; i++)
        {   v = ff.func(k1*c[i]+k2);
          intk = intk+v*xk[i];
          if( i%2==1 )
          { intg = intg+v*xg[i];}
        }
        intk = intk*(A[j][3]-A[j][2])*0.5;
        intg = intg*(A[j][3]-A[j][2])*0.5;
        A[j][0] = Math.abs(intg-intk);
        A[j][1] = intk;
        toplamhata = toplamhata+A[j][0];
      }
      cik(A, h-1, Aw);
      cik(A, h, Aw);
      if( toplamhata<eps )
      {break;}
    }
    sonuc = toplamhata<eps;
    integral = 0;
    for(j=0; j<=Aused-1; j++)
    { integral = integral+A[j][1];}
    return integral;
  }

  public static void gir(double A[][],int n,int Awidth)
  {
    int i = 0;
    int p = 0;
    double t = 0;
    int maxcp = 0;
    if( n==1 )
    { return; }
    for(i=0; i<=Awidth-1; i++)
    {   t = A[n-1][i];
      A[n-1][i] = A[0][i];
      A[0][i] = t;
    }//end of for
    p = 0;
    while( 2*p+1<n-1 )
    {   maxcp = 2*p+1;
      if( 2*p+2<n-1 )
      {   if( A[2*p+2][0]>A[2*p+1][0] )
        { maxcp = 2*p+2; }
      }
      if( A[p][0]<A[maxcp][0] )
      {   for(i=0; i<=Awidth-1; i++)
        {   t = A[p][i];
          A[p][i] = A[maxcp][i];
          A[maxcp][i] = t;
        }
```

```
            p = maxcp;
          }
          else
          { break; }
      } //end of while
   }

   public static void cik(double A[][],int n,int Awidth)
   {
      int i = 0;
      int p = 0;
      double t = 0;
      int kk = 0;
      if( n==0 )
      {return;}
      p = n;
      while( p!=0 )
      {   kk = (p-1)/2;
         if( A[p][0]>A[kk][0] )
         { for(i=0; i<=Awidth-1; i++)
            { t = A[p][i];
              A[p][i] = A[kk][i];
              A[kk][i] = t;
            }
            p = kk;
         }
         else
         {break;}
      }
   }

  public static void main(String axgs[])
  {
  f1 b1=new f1();
  System.out.println("integral of class        f1 : "+gauss_konrad_integral(b1,0.0,1.0,1.0e-20,50));
}
}
```

Ayarlanabilir hata miktarlı Gauss-Kronrod integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" adaptive_Gauss_Kronrod_integral
integral of class        f1 : 0.4

> Terminated with exit code 0.
```

7.4.4 Ayarlanabilir simpson integrali, Gander ve Gautschi[31] yaklaşımı

Şu ana kadarki yaklaşımlarımızda adaptif prosesten çıkış işlemi

if(Math.abs(I1-I2)< tolerans veya

if(Math.abs((I1-I2)/I1) < tolerans gibi bir temel yaklaşım izledi. Bu tür döngüden çıkış prosesi her integralde başarılı olmıyabilir. Bu durumda genelde maksimum iterasyon sayısına ulaşıldığında döngüden bir uyarı mesajı vererek çıkılabilir. Örneğin $f(x) = \sqrt{x}$ fonksiyonunu [0,1] aralığında program7.4.1 de verdiğimiz adaptive_simpson_integral ile hata seviyesi 10^{-12} ve maksimum iterasyon sayısı 10 olmak üzere çözersek;

```
 Public static void main(String args[]) throws IOException
 {
 f1 b1=new f1();
 System.out.println("integral of class        f1 : "+adaptive_simpson_integral(b1,0.0,1.0,1.0e-12,10));
```

```
  }
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ10
Maksimum iterasyon sayısına ulaşıldı
integral of class        f1 : 0.6666663569478286

> Terminated with exit code 0.
```

Sonuç Maksimum iterasyon sayısına ulaşıldı mesajını verecektir. Hata miktarını daha azaltmak hatta çalıştığımız makinenin hata sınırına ulaştırmak istersek daha değişik bir kontrol sistemi kullanabiliriz. Önce farklı kaba bir integral değeri olarak is değerini bulduğumuzu varsayalım.

if(is+(I1-I2)==is)

gibi bir yaklaşım kullanırsak bu yaklaşım ancak I1 ve I2 sonuçları makine hassaslığında birbirine eşit olduğunda çıkış verecektir. Bu yaklaşımı $m = \frac{(a-b)}{2}$ tanımı için ek bir sınır değer kontrolu ile daha da iyi çalışacak bir forma getirebiliriz [a,b] integral bölgesi için

if((is+(I1-I2)==is) || (m<=a) || (b<=m)) bize çok daha garantili bir çıkış kriteri oluşturabilir. Şimdi bu kavramı kullanan bir ayarlanabilir Simpson integral formülüne bakalım. is ilk kaba integral değeri için sekizinci derece polinom formülü n=8, 9 noktalı Newton-Cotes formülünü kullanacağız.

$$is = \int_a^b f(x)dx = \frac{(b-a)}{28350}[989 f(x_0) + 5888 f(x_1) - 928 f(x_2) + 10496 f(x_3) - 4540 f(x_4)$$
$$+ 10496 f(x_5) - 928 f(x_6) + 5888 f(x_7) + 989 f(x_8)]$$

İterative prosesde ise Simpson 1/3 kuralını

$$I1 = I = \int_a^b f(x)dx = I[a,b] = \frac{(b-a)}{6}[f(x_0) + 4 f(x_2) + f(x_4)]$$

$$I2 = I[a,m] + I[m,b] = \int_a^m f(x)dx + \int_m^b f(x)dx =$$

$$\frac{(m-a)}{6}[f(x_0) + 4 f(x_1) + f(x_2)] + \frac{(b-m)}{6}[f(x_2) + 4 f(x_3) + f(x_4)] =$$

$$\frac{(b-a)}{12}[f(x_0) + 4 f(x_1) + 2 f(x_2) + 4 f(x_3) + f(x_4)]$$

olacaktır. Program kodu **Program 7.4-4** de verilmiştir.

Program 7.4.4 Gander ve Gautschi hata Ayarlanabilir Simpson 1/3 integral formülü

```
//=====================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;

class f1 extends f_x
{
  double func(double x)
  {return Math.sin(x);}
}


class integ16
{
```

```
public static double adaptive_simpson_integral2(f_x ff,double a,double b)
{   int i;
              Plot pp=new Plot(ff,a,b,100);
   double h=(b-a);
   double h2=h/4.0;
   double h3=h/8.0;
   double m=(a+b)/2.0;
   double x[]=new double[9];
   double y[]=new double[9];
   for(i=0;i<9;i++) {x[i]=a+i*h3;y[i]=ff.func(x[i]);}
   double is=h/28350.0*(989.0*y[0]+5888.0*y[1]-928.0*y[2]+10496.0*y[3]-4540.0*y[4]+
        10496.0*y[5]-928.0*y[6]+5888.0*y[7]+989.0*y[8]);
   double fa = y[0];
   double fm = y[4];
   double fb = y[8];
   if(is==0) is = b-a;
   double Q = adaptsimstp(ff,a,b,fa,fm,fb,is);
   return Q;
}

public static double adaptsimstp(f_x ff,double a,double b,double fa,double fm,double fb,double is)
{
  double m = (a + b)/2;
  double h = (b - a)/4;
  double x[] = {a + h, b - h};
  int n=x.length;
  double y[] = new double[n];
  for(int i=0;i<n;i++) {y[i]=ff.func(x[i]);}
  double fml = y[0];
  double fmr = y[1];
  double i1 = h/1.5 * (fa + 4*fm + fb);
  double i2 = h/3 * (fa + 4*(fml + fmr) + 2*fm + fb);
  double Q=0;
  i1 = (16.0*i2 - i1)/15.0; //romberg integrasyonu ile iyileştirme
  if (((is + (i1-i2)) == is) || (m <= a) || (b<=m))
  {
    if ((m <= a) || (b<=m))
       { System.out.println("istenilen toleransa ulaşılamadı");}
    Q = i1;
  }
  else
  {
    Q = adaptsimstp(ff,a,m,fa,fml,fm,is)+
        adaptsimstp(ff,m,b,fm,fmr,fb,is);
  }
  return Q;
}

  public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  System.out.println("integral of class         f1 : "+adaptive_simpson_integral2(b1,0.0,Math.PI));
  }
}
```

Gander ve Gautschi hata Ayarlanabilir Simpson 1/3 integral formülü: örnek integral :

$$I = \int_0^{\pi} \sin(x)dx = 2$$

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ16
integral of class         f1 : 2.0
> Terminated with exit code 0.
```

7.4.4 Ayarlanabilir Gauss-Lobatto integrali, Gander ve Gautschi[31] yaklaşımı

Gauss –Lobatto integral formülü Gauss integral formuna benzeyen genel bir formdadır, en önemli değişiklik Lobatto formülünün en uç iki noktayı da kapsamasıdır.

$$I_w(-1,1) \cong \int_{-1}^{1} f(x)dx \cong \sum_{k=0}^{n-1} c_k f(x_k)$$

Çeşitli n değerleri için integral formülünün kök ve katsayı değerleri Tablo 7.4-1 de verilmiştir.

Tablo 7.4-1 Gauss-Lobatto denkleminin katsayıları ve kökleri

N	**Xk**	**Ck**
4	-1.000000000000000000000	0.166666666666666700000000
	1.000000000000000000000	0.166666666666666700000000
	-0.447213595499958000000000	0.833333333333333000000000
	0.447213595499958000000000	0.833333333333333000000000
5	-1.000000000000000000000	0.100000000000000000000
	1.000000000000000000000	0.100000000000000000000
	-0.654653670707977000000000	0.544444444444444000000000
	0.654653670707977000000000	0.544444444444444000000000
	0.000000000000000000000	0.711111111111111000000000
6	-1.000000000000000000000	0.066666666666666600000000
	1.000000000000000000000	0.066666666666666600000000
	-0.765055323929465000000000	0.378474956297847000000000
	0.765055323929465000000000	0.378474956297847000000000
	-0.285231516480635000000000	0.554858377035486000000000
	0.285231516480635000000000	0.554858377035486000000000
7	-1.000000000000000000000	0.052380952380952400000000
	1.000000000000000000000	0.052380952380952400000000
	-0.816496580927726000000000	0.293877551020408000000000
	0.816496580927726000000000	0.293877551020408000000000
	-0.447213595499958000000000	0.425170068027211000000000
	-0.447213595499958000000000	0.425170068027211000000000
	0.000000000000000000000	0.457142857142857000000000
12	-1.000000000000000000000	0.015827191973480100000000
	1.000000000000000000000	0.015827191973480100000000
	-0.942882415695479000000000	0.094273840218850000000000
	0.942882415695479000000000	0.094273840218850000000000
	-0.816496580927726000000000	0.155071987336585000000000
	0.816496580927726000000000	0.155071987336585000000000
	-0.641853342345781000000000	0.188821573960182000000000
	0.641853342345781000000000	0.188821573960182000000000
	-0.447213595499958000000000	0.199773405226858000000000
	0.447213595499958000000000	0.199773405226858000000000
	-0.236383199662149000000000	0.224926465333339000000000
	0.236383199662149000000000	0.224926465333339000000000

	0.00000000000000000000000	0.24261107190140700000000

Bu ayarlanabilir integral programında da **Gander ve Gautschi[31]** yaklaşımı yaklaşımını kullanarak döngü sonlandırması yapacağız. [a,b] integral bölgesi için

if((is+(I1 - I2) == is) || (m<=a) || (b<=m))
şartını kullanacağız. Buradaki m değeri daha önce de belirttiğimiz gibi

$m = \frac{(a-b)}{2}$ şeklindedir. . is ilk kaba integral değeri için 12. derece Lobato integral formülünü kullanacağız.

$$is \cong \int_a^b f(x)dx \cong (b-a)*\{0.0158271919734801*[f(a)+f(b)]+$$
$$0.094273840218850*[f(m-h*0.942882415695479)+f(m+h*0.942882415695479)]+$$
$$0.155071987336585*[f(m-h*0.816496580927726)+f(m+h*0.816496580927726)]+$$
$$0.188821573960182*[f(m-h*0.641853342345781)+f(m+h*0.641853342345781)]+$$
$$0.199773405226858*[f(m-h*0.447213595499958)+f(m+h*0.447213595499958)]+$$
$$0.224926465333339*[f(m-h*0.236383199662149)+f(m+h*0.236383199662149)]+$$
$$0.242611071901407*[f(0)]\}$$

iterasyon integrasyon formülleri olarak ta 4. ve 7. derece Gauss-Lobatto açılımlarından yararlanacağız.

$$I1 \cong \int_a^b f(x)dx \cong (b-a)*\{0.16666666\ 6666667\ *[f(a)+f(b)]+$$
$$0.83333333\ 3333333\ *[f(m-h*0.44721359\ 5499958)\ +f(m+h*0.44721359\ 5499958)]\ \}$$

$$I2 \cong \int_a^b f(x)dx \cong (b-a)*\{0.0523809523809524*[f(a)+f(b)]+$$
$$0.293877551020408*[f(m-h*0.816496580927726)+f(m+h*0.816496580927726)]+$$
$$0.425170068027211*[f(m-h*0.447213595499958)+f(m+h*0.447213595499958)]+$$
$$0.457142857142857*[f(0)]\}$$

Bu değerler kullanarak Program 7.4.5 geliştirilmiştir.

Program 7.4.5 Gander ve Gautschi hata Ayarlanabilir Gauss-Lobatto integral formülü

```
//=======================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =======================================================
import java.io.*;

class f1 extends f_x
{
  double func(double x)
  {//return 13.0*(x-x*x)*Math.exp(-1.5*x);
  return Math.sin(x);
  }
}

class integ15
{

public static double adaptive_Lobatto_integral(f_x ff,double a,double b)
{   int i;
            double m=(a+b)/2.0; double h=(b-a)/2.0;
```

```
    double alpha=Math.sqrt(2.0/3.0);
    double beta=1.0/Math.sqrt(5.0);
    double x1,x2,x3;
    x1=0.94288241569547971905635175843185720232;
    x2=0.64185334234578130578123554132903188354;
    x3=0.23638319966214988028222377349205292599;
    double A,B,C,D,E,F,G;
    A=0.01582719197348018308716998673330551059l;
    B=0.09427384021885004553128250507710817l960;
    C=0.15507198733658539625363597980210298680;
    D=0.18882157396018245442000533937297167125;
    E=0.19977340522685852679206802206648840246;
    F=0.22492646533333952701601768799639508076;
    G=0.24261107190140773379964095790325635233;
              double x[]={a,(m-x1*h),(m-alpha*h),(m-x2*h),(m-beta*h),(m-
x3*h),m,(m+x3*h),(m+beta*h),(m+x2*h),(m+alpha*h),(m+x1*h),b};
              int n=x.length;
              double y[]=new double[n];
              for(i=0;i<n;i++) y[i]=ff.func(x[i]);
              double fa=y[0];
              double fb=y[12];
    double i2=(h/6.0)*(y[0]+y[12]+5.0*(y[4]+y[8]));
    double i1=(h/1470.0)*(77.0*(y[0]+y[12])+432.0*(y[2]+y[10])+
     625.0*(y[4]+y[8])+672.0*y[6]);
    double is=h*(A*(y[0]+y[12])+B*(y[1]+y[11])+C*(y[2]+y[10])+D*(y[3]+y[9])+E*(y[4]+y[8])+F*(y[5]+y[7])+G*y[6]);
    int s=(int)(is/Math.abs(is));
    if(s==0) s=1;
    double erri1=Math.abs(i1-is);
    double erri2=Math.abs(i2-is);
    double R;
    R=erri1/erri2;
    //if(R>0 && R<1) tol=tol/R;
    //is=s*Math.abs(is)*tol/eps;
    if(is==0) is=b-a;
  double Q=0;
  Q=Lobatto(ff,a,b,fa,fb,is);
  return Q;
}

public static double Lobatto(f_x ff,double a,double b,double fa,double fb,double is)
{
  double h=(b-a)/2;
  double m=(a+b)/2;
  double alpha=Math.sqrt(2.0/3.0);
  double beta=1.0/Math.sqrt(5.0);
  double mll=m-alpha*h;
  double ml=m-beta*h;
  double mr=m+beta*h;
  double mrr=m+alpha*h;
  double x[]={mll,ml,m,mr,mrr};
  int n=x.length;
  double y[]=new double[n];
  for(int i=0;i<n;i++) y[i]=ff.func(x[i]);
  double fmll=y[0];
  double fml=y[1];
  double fm=y[2];
  double fmr=y[3];
  double fmrr=y[4];
  double Q;
  double i2=(h/6)*(fa+fb+5*(fml+fmr));
  double i1=(h/1470)*(77*(fa+fb)+432*(fmll+fmrr)+625*(fml+fmr)+672*fm);
  boolean c1,c2,c3;
  double w1=i1-i2;;
  double w2=is;
  double w3=w2+w1;
  c1=(w3==w2);
```

```
  c2=(mll<=a);
  c3=(b<=mrr);
  if( c1|| c2 ||c3 )
  {
   if (((m <= a) || (b<=m)))
   {
      System.out.println("istenilen toleransa ulaşılamadı");
   }
   Q=i1;
  }
  else
  {
    Q=Lobatto(ff,a,mll,fa,fmll,is) + Lobatto(ff,mll,ml,fmll,fml,is)+
      Lobatto(ff,ml,m,fml,fm,is)   + Lobatto(ff,m,mr,fm,fmr,is)+
      Lobatto(ff,mr,mrr,fmr,fmrr,is)+Lobatto(ff,mrr,b,fmrr,fb,is);
  }
  return Q;
}

  public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  System.out.println("integral of class         f1 : "+adaptive_Lobatto_integral(b1,0.0,Math.PI));
  }
}
```

Gander ve Gautschi hata Ayarlanabilir Gauss-Lobatto integral formülü

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ15
integral of class         f1 : 2.0
> Terminated with exit code 0.
```

Program 7.4.6 da bir matematik fonksiyon uygulaması yapmak istiyoruz. Isı transferi ve diğer çözümlerde kulanılan hata fonksiyonun(erf) temel tanımı

$erf(\eta) = \frac{1}{\sqrt{\pi}} \int_0^{\eta} e^{-\eta^2} d\eta$ formülüyle yapılır ancak bu integral hesaplanması çok zor olan bir integral olduğundan genelde bu formül seri çözümlerle hesaplanır. Örnek programımızda seri çözüm ve Gauss-Lobatto integralinden hesaplanmaktadır.

Program 7.4.6 Gauss-Lobatto integral formülü ile erf fonksiyonun hesaplanması

```
//=====================================================
// Numerical Analysis package in java
// error function calculation by using
// Adaptive Gauss-Lobatto integral and serial solution
// Referans : Adapted Quadrature - Revisited
// Walter Gander - Walter Gautchi
// ftp.inf.ethz.ch
// Dr. Turhan Coban
// =====================================================

import java.io.*;
import javax.swing.*;

class f1 extends f_x
{ double func(double x)
  {return 2.0/Math.sqrt(Math.PI)*Math.exp(-x*x);}
}

class erf
{
```

```
public static double gammp(double a,double x)
{
	//incomplete gamma function P(a,x)
	double gamser,gammcf;
	if (x < 0.0 || a <= 0.0) System.out.println("Invalid arguments in routine GAMMP");
	if (x < (a+1.0)) {
		gamser=gser(a,x);
		return gamser;
	} else {
		gammcf=gcf(a,x);
		return 1.0-gammcf;
	}
}
public static double gser(double a,double x)
{
	int ITMAX=100;
	double EPS=3.0e-7;
	int n;
	double sum,del,ap;
	double gln=gammln(a);
	double gamser=0;
	if (x <= 0.0) {
		if (x < 0.0) System.out.println("x less than 0 in routine GSER");
		gamser=0.0;
		return gamser;
	} else {
		ap=a;
		del=sum=1.0/a;
		for (n=1;n<=ITMAX;n++) {
			ap += 1.0;
			del *= x/ap;
			sum += del;
			if (Math.abs(del) < Math.abs(sum)*EPS) {
				gamser=sum*Math.exp(-x+a*Math.log(x)-(gln));
				return gamser;
			}
		}
		System.out.println("a too large, ITMAX too small in routine GSER");
		return gamser;
	}
}

public static double gcf(double a,double x)
{
	int ITMAX=100;
	double EPS=3.0e-7;
	int n;
	double gln;
	double gammcf=0;
	double gold=0.0,g,fac=1.0,b1=1.0;
	double b0=0.0,anf,ana,an,a1,a0=1.0;
	gln=gammln(a);
	a1=x;
	for (n=1;n<=ITMAX;n++) {
		an=(double) n;
		ana=an-a;
		a0=(a1+a0*ana)*fac;
		b0=(b1+b0*ana)*fac;
		anf=an*fac;
		a1=x*a0+anf*a1;
		b1=x*b0+anf*b1;
		if (a1!=0) {
			fac=1.0/a1;
			g=b1*fac;
			if (Math.abs((g-gold)/g) < EPS) {
```

```
                                        gammcf=Math.exp(-x+a*Math.log(x)-gln)*g;
                                        return gammcf;
                                }
                                gold=g;
                        }
                }
                System.out.println("a too large, ITMAX too small in routine GCF");
    return gammcf;
}
public static double gammln(double xx)
{
                //gamma function
                double x,y,tmp,ser;
                double cof[]={76.18009172947146,-86.50532032941677,
                                24.01409824083091,-1.231739572450155,
                                0.1208650973866179e-2,-0.5395239384953e-5};
                int j;
                y=x=xx;
                tmp=x+5.5;
                tmp -= (x+0.5)*Math.log(tmp);
                ser=1.000000000190015;
                for (j=0;j<=5;j++) ser += cof[j]/++y;
                return -tmp+Math.log(2.5066282746310005*ser/x);
}
public static double  erf(double x)
{
   //error function from series solution
   return x < 0.0 ? -gammp(0.5,x*x) : gammp(0.5,x*x);
}

public static double  erf_int(double x)
{
   //error function from series solution
   f1 b1=new f1();
   return adaptive_Lobatto_integral(b1,0.0,x);
}


public static double adaptive_Lobatto_integral(f_x ff,double a,double b)
{double tol=Double.MIN_VALUE;
 return adaptive_Lobatto_integral(ff,a,b,tol);
}

public static double adaptive_Lobatto_integral(f_x ff,double a,double b,double tol)
{   int i;
                double m=(a+b)/2.0; double h=(b-a)/2.0;
    double alpha=Math.sqrt(2.0/3.0);
    double beta=1.0/Math.sqrt(5.0);
    double x1,x2,x3;
    double eps =Double.MIN_VALUE;
    //System.out.println("eps="+eps);
    x1=0.94288241569547971905635175843185720232;
    x2=0.64185334234578130578123554132903188354;
    x3=0.23638319966214988028222377349205292599;
    double A,B,C,D,E,F,G;
    A=0.015827191973480183087169986733305510591;
    B=0.094273840218850045531282505077108171960;
    C=0.15507198733658539625363597980210298680;
    D=0.18882157396018245442000533937297167125;
    E=0.19977340522685852679206802206648840246;
    F=0.22492646533333952701601768799639508076;
    G=0.24261107190140773379964095790325635233;
                double x[]={a,(m-x1*h),(m-alpha*h),(m-x2*h),(m-beta*h),(m-
x3*h),m,(m+x3*h),(m+beta*h),(m+x2*h),(m+alpha*h),(m+x1*h),b};
                int n=x.length;
                double y[]=new double[n];
```

```
                for(i=0;i<n;i++) y[i]=ff.func(x[i]);
                double fa=y[0];
                double fb=y[12];
    double i2=(h/6.0)*(y[0]+y[12]+5.0*(y[4]+y[8]));
    double i1=(h/1470.0)*(77.0*(y[0]+y[12])+432.0*(y[2]+y[10])+
     625.0*(y[4]+y[8])+672.0*y[6]);
    double is=h*(A*(y[0]+y[12])+B*(y[1]+y[11])+C*(y[2]+y[10])+D*(y[3]+y[9])+E*(y[4]+y[8])+F*(y[5]+y[7])+G*y[6]);
    int s=(int)(is/Math.abs(is));
    if(s==0) s=1;
    double erri1=Math.abs(i1-is);
    double erri2=Math.abs(i2-is);
    double R;
    R=erri1/erri2;
    if(R>0 && R<1) tol=tol/R;
    //System.out.println("is="+is+"s="+s+"tol="+tol+"eps="+eps);
    if(is==0) is=b-a;
  double Q=0;
  Q=Lobatto(ff,a,b,fa,fb,is,tol);
  return Q;
}

public static double Lobatto(f_x ff,double a,double b,double fa,double fb,double is,double tol)
{
  double h=(b-a)/2;
  double m=(a+b)/2;
  double alpha=Math.sqrt(2.0/3.0);
  double beta=1.0/Math.sqrt(5.0);
  double mll=m-alpha*h;
  double ml=m-beta*h;
  double mr=m+beta*h;
  double mrr=m+alpha*h;
  double x[]={mll,ml,m,mr,mrr};
  int n=x.length;
  double y[]=new double[n];
  for(int i=0;i<n;i++) y[i]=ff.func(x[i]);
  double fmll=y[0];
  double fml=y[1];
  double fm=y[2];
  double fmr=y[3];
  double fmrr=y[4];
  double Q;
  double i2=(h/6)*(fa+fb+5*(fml+fmr));
  double i1=(h/1470)*(77*(fa+fb)+432*(fmll+fmrr)+625*(fml+fmr)+672*fm);
  double di=Math.abs(i1-i2);
  if (((is + di) == is) || (mll <= a) || (b<=mrr) || di<tol)
  {
   if (((m <= a) || (b<=m)))
   { System.out.println("istenilen toleransa ulaşılamadı delta i = "+di);}
   Q=i1;
  }
  else
  {
    Q=Lobatto(ff,a,mll,fa,fmll,is,tol) + Lobatto(ff,mll,ml,fmll,fml,is,tol)+
      Lobatto(ff,ml,m,fml,fm,is,tol)   + Lobatto(ff,m,mr,fm,fmr,is,tol)+
      Lobatto(ff,mr,mrr,fmr,fmrr,is,tol)+Lobatto(ff,mrr,b,fmrr,fb,is,tol);
  }
  return Q;
}

  public static void main(String args[]) throws IOException
  {
  double x=Double.parseDouble(JOptionPane.showInputDialog(" x = "));
  System.out.println("erf("+x+") = "+erf(x)+"integral erf="+erf_int(x));
  }
}
```

Çıktı 7.4-6 Gauss-Lobatto integral formülü ile erf fonksiyonun hesaplanması

```
erf(0.1) = 0.1124629159993309  integral erf=0.1124629160182849
erf(0.4) = 0.4283923519241598  integral erf=0.42839235504666845
erf(1.0) = 0.8427007900291826  integral erf=0.8427007929497149
erf(2.0) = 0.995322264754383   integral erf=0.9953222650189527
```

7.5 Clenshaw-curtis integral formülü

Clenshaw-Curtis formülü x=cosθ transformuna dayanır. Bu transformasyon gerçekleştirildiğinde -1,1 aralığında integral denklemi

$$\int_{-1}^{1} f(x)dx = \int_{o}^{\pi} f(\cos\theta)\sin(\theta)d\theta$$

formunu alır. Bir fonksiyonun cos serisine açılma denklemi :

$$f(\cos\theta) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(k\theta)$$

şeklindedir. Bu açılım yapıldığında, seri üzerinden integral açılımını kolaylıkla gerçekleştirebiliriz.

$$\int_{o}^{\pi} f(\cos\theta)\sin(\theta)d\theta = a_0 + \sum_{k=1}^{\infty} \frac{2a_{2k}}{1-(2k)^2}$$

buradaki katsayılar :

$$a_k = \frac{2}{\pi}\int_{o}^{\pi} f(\cos\theta)\cos(k\theta)d\theta$$

denkleminden hesaplanabilir. Bu denklemi seriye açacak olursak, katsayıları hesaplamak için

$$a_k = \frac{2}{N}\left[\frac{f(1)}{2} + \frac{f(-1)}{2}(-1)^k + \sum_{n=1}^{N-1} f(\cos[\frac{n\pi}{N}])\cos[\frac{nk\pi}{N}]\right]$$

denklemini elde ederiz. Bu denklemi kullanarak üstteki formülden integral değerini hesaplayabiliriz.

Program 7.5-1 Clenshaw-Curtis integrasyonu

```
///=====================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class f1 extends f_x
{
  public double func(double x)
  {return x*Math.sqrt(x);}
}

class integ12
{
public static double[] clenshaw_curtis_coefficients(f_x f,double x1,double x2,int n)
{
   double a[]=new double[2*n];
              int cc=-1;
              int i,k;
              double x;
```

```
                double k1=(x2-x1)/2.0;
               double k2=(x2+x1)/2.0;
                for(k=0;k<50;k++)
                {
                a[k]=0.0;
                for (i=1;i<n;i++)
                {x=k2+k1*Math.cos(Math.PI*i/n); a[k]+=f.func(x)*Math.cos(i*k*Math.PI/n);}
               cc*=-1;
                a[k]+=f.func(x2)/2.0+f.func(x1)/2.0*cc;
                a[k]*=2.0/n;
    }
    return a;
}

public static double clenshaw_curtis_integral(f_x f,double x1,double x2,int n)
{
double a[]=clenshaw_curtis_coefficients(f,x1,x2,n);
double z=a[0];
double k1=(x2-x1)/2.0;
double k2=(x2+x1)/2.0;
for(int k=1;k<50;k+=2)
{ z+=2.0*a[2*k]/(1.0-4.0*k*k);}
return k1*z;
}

  public static void main(String args[]) throws IOException
  {
  //Gauss-Legendre integrali
  f1 b1=new f1();
  System.out.println("integral  : "+clenshaw_curtis_integral(b1,0.0,1.0,20));
}}
```

Çıktı 7.5-1 Clenshaw-Curtis integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ12
integral  : 0.40015015740404086
> Terminated with exit code 0.
```

7.6 Tek ve çok boyutlu Monte-Carlo integrali

Monte karlo integrali çok basit bir prensibe dayanmaktadır.

$$\int_a^b f(x)dx=(b-a)*f_{ortalama}$$

$$x = a+(b-a)*tesadüfi\_sayı(0,1)$$

$$f_{ortalama}=\frac{\sum_{i=0}^{n}\overline{f}(x)}{n},\qquad n>>1$$

a ve b arasında tesadüfi bir x noktası bularak fonksiyonu bu noktada değerlendiririz. Bu işlemi çok yüksek sayılar için tekrarladığımızda toplamın toplam sayıya bölümü ortalama fonksiyon değerini verecektir. Bu değerin bölge genişliğine çarpımı integrali verir. Denklemden ve programlardan görüldüğü gibi bu yöntem çok fazla sayıda fonksiyon değerlendirilmesini gerektirir. Bu yüzden birinci dereceden integraller için pek pratik bir formül değildir. Ancak katlı integrallerde, oldukça pratik hale gelir.

Program 7.6.1 tek değişkenli Monte-carlo integrasyonu

```
//=====================================================
// Numerical Analysis package in java
```

```
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// ======================================================

import java.io.*;

class f1 extends f_x
{
  public double func(double x)
  {return 3.0*x*x;}
}

class integ_monte_carlo
{

public static double monte_carlo(f_x ff,double a,double b, long n)
{       //Monte-carlo integrali
        double h=(b-a);
        double sum=0.0;
        double x=0.0;
        double f;
        for(int i=0;i<n;i++)
        { x=a+(b-a)*Math.random();
          sum+=ff.func(x);
        }
        return (b-a)*sum/(double)n;
}

  public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  System.out.println("integral of class         f1 : "+monte_carlo(b1,0.0,1.0,1000000));
}
}
```

tek değişkenli Monte-carlo integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ_monte_carlo
integral of class         f1 : 0.9990028422107341
> Terminated with exit code 0.
```

İki değişkenli integral için monte karlo denklemlerini yazarsak :

$$x = ax + (bx - ax) * tesadüfi\ \_ sayı(0,1)$$

$$y = ay + (by - ay) * tesadüfi\ \_ sayı(0,1)$$

$$f_{ortalama} = \frac{\sum_{i=0}^{n} \overline{f}(x,y)}{n}, \qquad n >> 1$$

integralin alt alanı

$$\mu(D) = \int_{ay}^{by} \int_{ax=\alpha(x)}^{bx=\beta(x)} dy\, dx$$

$$I = \mu(D)\, f_{ortalama}$$

halini alır, görüldüğü gibi iki değişkenli monte-carlo integrali bir değişkenli örneği ile hemen hemen aynı işlemlerle gerçekleştirilmektedir. Bu da n boyutlu integrale geçtiğimizde bu metodu cazip kılan önemli bir avantaj sağlar.

Program 7.6.2 iki değişkenli Monte-carlo integrasyonu

```
//=====================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;

class f1 extends f_xj
{
public double func(double x[])
{
//çözümü istenen fonksiyon
double ff;
ff=2.0*x[0]*x[1]+2.0*x[0]-x[0]*x[0]-2.0*x[1]*x[1];
return -ff; //minimum değil maxsimum istendiğinden - ile çarptık
}
}

class integ_monte_carlo2
{
public static double monte_carlo(f_xj ff,double a[],double b[], long n)
{      //Newton-cotes integral 6 points
                double x[]=new double[2];
                double h=1.0;
                double sum=0;
                for(int j=0;j<2;j++) {h*=(b[j]-a[j]);}
                for(int i=0;i<n;i++)
      { for(int j=0;j<2;j++) {x[j]=a[j]+(b[j]-a[j])*Math.random();}
        sum+=ff.func(x);
      }
      return h*sum/(double)n;
}

  public static void main(String args[]) throws IOException
  {
  f1 b1=new f1();
  double a[]=new double[2];
  double b[]=new double[2];
  a[0]=-1;
  a[1]=-1;
  b[0]=1;
  b[1]=1;
  System.out.println("integral of class        f1 : "+monte_carlo(b1,a,b,1000000));
}}
```

Çıktı 7.6-2 iki değişkenli Monte-carlo integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ_monte_carlo2
integral of class        f1 : 3.9945397938584297

> Terminated with exit code 0.
```

7.7 Çok boyutlu integralleri parametrik metodla çözme

İki boyutlu bir integrali

$$I(D) = \int\int_D f(x,y)dydx$$

D nin xy düzleminde bir bölge olduğunu varsayalım. Bu bölgeyi $a \le x \le b$ ve olarak tanımlıyalım.
İki boyutlu integrali hesaplamanın bir yolu iki stepte hesap yapmaktır. Birinci stepte g(x) fonksiyonunu hesaplarız.

$$\mathrm{g(x)} = \int_{c(x)}^{d(x)} f(x,y)dy, \quad a \le x \le b$$

ikinci stepte g(x) fonksiyonunu kullanarak tam integrali hesaplarız.

$$\mathrm{I(D)} = \int_{a}^{b} g(x)dx$$

Gauss-Legendre integrasyon formülü çok katlı integrallerin hesaplanması içinde kullanılabilir. Burada yapmamız gereken temel işlem integrali iç içe iki kere değerlendirmekten ibarettir. Örnek olarak 2 boyutlu Gauss-Legendre integrasyonunu problem 7.7.1 ve 7.7.2 de veriyoruz. İlk programımızda katsayılar sabit verilmiştir. İkinci programımızda Legendre polinomları üzerinden katsayılar hesaplanmaktadır.

$$I = \int_{a}^{b} \int_{c(x)}^{d(x)} f(x,y)dydx$$

iki boyutlu(değişkenli) integralde x boyutu sınırları yine bir boyutlu integral gibi iki nokta olarak verilirken , yboyutu sınırları x'in fonksiyonu olarak tanımlanabilir. Bu durumda x integral sınırları a ve b ve y integral sınırları c(x) ve y(x) bilinmelidir. Bu sınırlar temel olarak değişken fonksiyonlar olduklarından programımızda fonksiyon olarak tanımlanmışlardır. Gauss Legendre integralinde aynı zamanda değişkenleri de dönüştürmemiz gerekir.

$$I = \int_{a}^{b} \int_{c(x)}^{d(x)} f(x,y)dy = \int_{a}^{b} \frac{d(x)-c(x)}{2} \sum_{J=1}^{n} c_{n,j} f(x, \frac{(d(x)-c(x))r_{n,j} + d(x) + c(x)}{2})dx$$

buradaki $r_{n,j}$ kök değerleri, $c_{n,j}$ katsayılardır. Bu değerler Legendre polinomlarından hesaplanırlar.

Program 7.7-1 60 sabit noktalı 2 değişkenli Gauss-Legendre integrasyonu

```
//=====================================================
// Numerical Analysis package in java
// gauss integration of two dimensional function
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;

abstract class fdi
{
public double a,b;
public fdi(double ai,double bi)
{a=ai;b=bi;}
public fdi()
{a=0;b=1;}
public void setab(double ai,double bi)
{a=ai;b=bi;}
abstract public  double c(double x);
abstract public  double d(double x);
abstract public  double f(double x[]);
}

class f1 extends fdi
{
  public f1(double ai,double bi)
  {super(ai,bi);}

  public double c(double x)
  {return 1.0;}
```

```
  public double d(double x)
  {return 1.5;}

  public double f(double x[])
  {
  //çözümü istenen fonksiyon
  double ff;
  ff=Math.log(x[0]+2.0*x[1]);
  return ff;
  }
}


class integ_gauss2
{

public static double I(fdi f)
{
//integral f(x)dx
//integral of a function by using gauss-legendre quadrature
//coefficients are pre-calculated for 60 terms for [-1,1]
//band then utilises variable transform
 int i,j;
 int n=60;
 double r[],c[];
 r=new double[n];
 c=new double[n];
 r[ 0] = .15532579626752470000E-02;
 r[ 1] = .81659383601264120000E-02;
 r[ 2] = .19989067515846230000E-01;
 r[ 3] = .36899976285362850000E-01;
 r[ 4] = .58719732103973630000E-01;
 r[ 5] = .85217118808615820000E-01;
 r[ 6] = .11611128394758690000E+00;
 r[ 7] = .15107475260334210000E+00;
 r[ 8] = .18973690850537860000E+00;
 r[ 9] = .23168792592899010000E+00;
 r[10] = .27648311523095540000E+00;
 r[11] = .32364763723456090000E+00;
 r[12] = .37268153691605510000E+00;
 r[13] = .42306504319570830000E+00;
 r[14] = .47426407872234120000E+00;
 r[15] = .52573592127765890000E+00;
 r[16] = .57693495680429170000E+00;
 r[17] = .62731846308394490000E+00;
 r[18] = .67635236276543910000E+00;
 r[19] = .72351688476904450000E+00;
 r[20] = .76831207407100990000E+00;
 r[21] = .81026309149462140000E+00;
 r[22] = .84892524739665800000E+00;
 r[23] = .88388871605241310000E+00;
 r[24] = .91478288119138420000E+00;
 r[25] = .94128026789602640000E+00;
 r[26] = .96310002371463720000E+00;
 r[27] = .98001093248415370000E+00;
 r[28] = .99183406163987350000E+00;
 r[29] = .99844674203732480000E+00;
c[ 0] = .39840962480827790000E-02;
c[ 1] = .92332341555455000000E-02;
c[ 2] = .14392353941661670000E-01;
c[ 3] = .19399596284813530000E-01;
c[ 4] = .24201336415292590000E-01;
c[ 5] = .28746578108808720000E-01;
c[ 6] = .32987114941090080000E-01;
c[ 7] = .36877987368852570000E-01;
```

```
c[ 8] = .40377947614710090000E-01;
c[ 9] = .43449893600541500000E-01;
c[10] = .46061261118893050000E-01;
c[11] = .48184368587322120000E-01;
c[12] = .49796710293397640000E-01;
c[13] = .50881194874202750000E-01;
c[14] = .51426326446779420000E-01;
c[15] = .51426326446779420000E-01;
c[16] = .50881194874202750000E-01;
c[17] = .49796710293397640000E-01;
c[18] = .48184368587322120000E-01;
c[19] = .46061261118893050000E-01;
c[20] = .43449893600541500000E-01;
c[21] = .40377947614710090000E-01;
c[22] = .36877987368852570000E-01;
c[23] = .32987114941090080000E-01;
c[24] = .28746578108808720000E-01;
c[25] = .24201336415292590000E-01;
c[26] = .19399596284813530000E-01;
c[27] = .14392353941661670000E-01;
c[28] = .92332341555455000000E-02;
c[29] = .39840962480827790000E-02;
for(i=0;i<30;i++) {r[i+30]=-r[i];c[i+30]=c[i];}
int m=n;
double x[],h1,h2,J,a1,b1,c1,d1,k1,k2,JX,Q;
x=new double[2];
//1=========================
a1=f.a;
b1=f.b;
h1=(b1-a1)/2.0;
h2=(b1+a1)/2.0;
J=0;
//2=========================
for(i=0;i<m;i++)
{
//3=========================
JX=0;
x[0]=h1*r[i]+h2;
d1=f.d(x[0]);
c1=f.c(x[0]);
k1=(d1-c1)/2.0;
k2=(d1+c1)/2.0;
//4=========================
   for(j=0;j<n;j++)
   {
            x[1]=k1*r[j]+k2;
            Q=f.f(x);
            JX+=c[j]*Q;
            }
//System.out.println("JX="+JX);
//5=========================
J=J+c[i]*k1*JX;
}
J=h1*J;
return J;
}

  public static void main(String args[]) throws IOException
  {
  f1 ff=new f1(1.4,2.0);
  System.out.println("integral of class        f1 : "+I(ff));
}
}
```

60 sabit noktalı 2 değişkenli Gauss-Legendre integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ_gauss2
integral of class        f1 : 0.429554527548266
> Terminated with exit code 0.
```

Yukardaki problem 60 noktalı sabit katsayılar üzerinden tanımlanmıştır, aynı programı değişken nokta sayısı üzerinden, katsayılar program tarafından hesaplanacak şekilde de yazabiliriz.

Program 7.7-2 2 boyutlu değişken integrasyon sabit sayılı Gauss-Legendre integrasyonu

```
//=====================================================
// Numerical Analysis package in java
// gauss integration of two dimensional function
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;

abstract class fdi
{
public double a,b;
public fdi(double ai,double bi)
{a=ai;b=bi;}
public fdi()
{a=0;b=1;}
public void setab(double ai,double bi)
{a=ai;b=bi;}
abstract public  double c(double x);
abstract public  double d(double x);
abstract public  double f(double x[]);
}

class f1 extends fdi
{
  public f1(double ai,double bi)
  {super(ai,bi);}

  public double c(double x)
  {return 1.0;}

  public double d(double x)
  {return 1.5;}

  public double f(double x[])
  {
  //çözümü istenen fonksiyon
  double ff;
  ff=Math.log(x[0]+2.0*x[1]);
  return ff;
  }
}

class integ_gauss3
{

public static double[][] gauss_legendre_coefficients(double x1,double x2,int n)
{
                //calculates legendre gauss-coefficients as coefficients of the integral
                //for n terms
                double EPS=3.0e-15;
                int m,j,i;
                double z1,z,xm,xl,pp,p3,p2,p1;
               double a[][]=new double[2][n];
                m=(n+1)/2;
```

```
		xm=0.5*(x2+x1);
		xl=0.5*(x2-x1);
		for (i=1;i<=m;i++)  {
			z=Math.cos(Math.PI*((i-0.25)/(n+0.5)));
			do {
				p1=1.0;
				p2=0.0;
				for (j=1;j<=n;j++) {
					p3=p2;
					p2=p1;
					p1=((2.0*j-1.0)*z*p2-(j-1.0)*p3)/j;
				}
				pp=n*(z*p1-p2)/(z*z-1.0);
				z1=z;
				z=z1-p1/pp;
			} while (Math.abs(z-z1) > EPS);
			a[0][i-1]=xm-xl*z;
			a[0][n-i]=xm+xl*z;
			a[1][i-1]=2.0*xl/((1.0-z*z)*pp*pp);
			a[1][n-i]=a[1][i-1];
				}
return a;
}

public static double I(fdi f,int n,int m)
{
//integral f(x,y)dxdy
//integral of a function by using gauss-legendre quadrature
//coefficients are pre-calculated for n,m terms for [-1,1]
//band then utilises variable transform
 int i,j;
 double rn[],cn[];
 double rm[],cm[];
 rn=new double[n];
 rm=new double[m];
 cn=new double[n];
 double a[][]=new double[2][n];
 a=gauss_legendre_coefficients(-1.0,1.0,n);
 rn=a[0];
 cn=a[1];
 a=gauss_legendre_coefficients(-1.0,1.0,m);
 rm=a[0];
 cm=a[1];
double x[],h1,h2,J,a1,b1,c1,d1,k1,k2,JX,Q;
x=new double[2];
a1=f.a;
b1=f.b;
h1=(b1-a1)/2.0;
h2=(b1+a1)/2.0;
J=0;
for(i=0;i<m;i++)
{
JX=0;
x[0]=h1*rm[i]+h2;
d1=f.d(x[0]);
c1=f.c(x[0]);
k1=(d1-c1)/2.0;
k2=(d1+c1)/2.0;
//4=========================
   for(j=0;j<n;j++)
   {
		x[1]=k1*rn[j]+k2;
		Q=f.f(x);
		JX+=cn[j]*Q;
		}
//System.out.println("JX="+JX);
```

```
//5=========================
J=J+cm[i]*k1*JX;
}
J=h1*J;
return J;
}

public static double I(fdi f,int n)
{return I(f,n,n);}

public static double I(fdi f)
{return I(f,60,60);}

  public static void main(String args[]) throws IOException
  {
  f1 ff=new f1(1.4,2.0);
  System.out.println("integral of class        f1 : "+I(ff));
}
}
```

2 boyutlu Gauss-Legendre integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ_gauss3
integral of class        f1 : 0.4295545275482747

> Terminated with exit code 0.
```

İki değişkenli integrasyon için Newton-Cotes (Simpson) formüllerinden de yararlanabiliriz. Örneğin simpson 1/3 kuralını uygulamak istersek, hatırlıyacağımız gibi bu kural : ikinci derece polinom formülü(simpson 1/3 kuralı) n=2 için

$$h=\frac{(b-a)}{n}=\frac{(b-a)}{2}$$

$$I=\int_a^b f(x)dx=\frac{(b-a)}{6}[f(x_0)+4f(x_1)+f(x_2)]$$

şeklinde idi. Aynı formülü kompozit olarak yazarsak (alt parçalara ayrılmış şekilde) yazacak olursak

$$I=\int_a^b f(x)dx=\frac{h}{3}\left[f(a)+2\sum_{j=1}^{(n/2)-1}f(x_{2j})+4\sum_{j=1}^{(n/2)}f(x_{2j-1})\right]$$ şeklinde yazılabilir. Bu denklemi Gauss-

Legendre 2 boyutlu integralinde olduğu gibi bu formülü de 2 boyuta açabiliriz.

$$I=\int_a^b\int_{c(x)}^{d(x)}f(x,y)dydx$$

$k=\frac{(d-c)}{m}$ olarak adlandırdığımızda iç integral

$$I=\int_c^d f(x,y)dy=\frac{k}{3}\left[f(x,y_0)+2\sum_{j=1}^{(m/2)-1}f(x,y_{2j})+4\sum_{j=1}^{(m/2)}f(x,y_{2j-1})+f(x,y_m)\right]$$

formunu alır. Buradan dış integrale geçebiliriz.

$$I=\int_a^b\int_c^d f(x,y)dy=\int_a^b\frac{k}{3}\left[f(x,y_0)+2\sum_{j=1}^{(m/2)-1}f(x,y_{2j})+4\sum_{j=1}^{(m/2)}f(x,y_{2j-1})+f(x,y_m)\right]dx$$

aynı denklem ikinci kere dış integrale uygulanarak iki boyutlu integral çözülmüş olur.

Program 7.7.3 de 2 boyutlu simpson 1/3 integral formül uygulaması mevcuttur. Fonksiyon olarak Gauss-Legendre örneğinde aldığımız aynı fonksiyon alınmıştır.

Program 7.7-3 iki boyutlu simpson integrasyonu

```
//=====================================================
// Numerical Analysis package in java
// simpson integration of two dimensional function
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;

abstract class fdi
{
public double a,b;
public fdi(double ai,double bi)
{a=ai;b=bi;}
public fdi()
{a=0;b=1;}
public void setab(double ai,double bi)
{a=ai;b=bi;}
abstract public  double c(double x);
abstract public  double d(double x);
abstract public  double f(double x[]);
}

class f1 extends fdi
{
  public f1(double ai,double bi)
  {super(ai,bi);}

  public double c(double x)
  {return 1.0;}

  public double d(double x)
  {return 1.5;}

  public double f(double x[])
  {
  //çözümü istenen fonksiyon
  double ff;
  ff=Math.log(x[0]+2.0*x[1]);
  return ff;
  }}

class integ_simpson
{

public static double I(fdi f,int m,int n)
{
//integral f(x)dx
//integral of a function by using simpson metod
double h=(f.b-f.a)/n;
double J1=0.0;
double J2=0.0;
double J3=0.0;
double x[]=new double[2];
double HX=0;
```

```
double K1,K2,K3;
double d,c;
double Q,L,J;
for(int i=0;i<=n;i++)
{
x[0]=f.a+i*h;
d=f.d(x[0]);
c=f.c(x[0]);
HX=(d-c)/m;
x[1]=c;K1=f.f(x);
x[1]=d;K1+=f.f(x);
K2=0;
K3=0;
  for(int j=1;j<=(m-1);j++)
  {x[1]=f.c(x[0])+j*HX;
  Q=f.f(x);
  if((j/2)*2==j) K2=K2+Q;
  else          K3=K3+Q;
  }
L=(K1+2.0*K2+4.0*K3)*HX/3.0;
if((i==0)||(i==n)) J1=J1+L;
else if ((i/2)*2==i) J2=J2+L;
else J3=J3+L;
}
J=h*(J1+2.0*J2+4.0*J3)/3.0;
return J;
}
  public static void main(String args[]) throws IOException
  {
  f1 ff=new f1(1.4,2.0);
  System.out.println("integral of class        f1 : "+I(ff,20,20));
}}
```

Çıktı 7.7-3 iki boyutlu simpson integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\javaw.exe" integ_simpson
integral of class        f1 : 0.4295545273061155

> Terminated with exit code 0.
```

Şimdi de aynı prensibi üç boyutlu integraller için kullanalım. Üç boyutlu integralin genel formülü :

$$I = \int_a^b \int_{c(x)}^{d(x)} \int_{\alpha(x,y)}^{\beta(x,y)} f(x,y,z)dzdydx$$

şeklinde verilebilir. Burada çözüm metodu olarak Gauss-Legendre integrali kullanacağız. İlk örnek integral olarak :

$$I = \int_0^1 \int_x^1 \int_0^y y^2 z dz dy dx$$

integralini çözelim

Program 7.7.4 üç boyutlu Gauss-Legendre integrasyonu

```
//=====================================================
// Numerical Analysis package in java
// gauss integration of three dimensional function
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;

abstract class fdi3
```

```
{
public double a,b;
public fdi3(double ai,double bi)
{setab(ai,bi);}
public void setab(double ai,double bi)
{a=ai;b=bi;}
abstract public  double c(double x);
abstract public  double d(double x);
abstract public  double alfa(double x[]);
abstract public  double beta(double x[]);
abstract public  double f(double x[]);
}

class f1 extends fdi3
{
  public f1(double ai,double bi)
  {super(ai,bi);}

  public double c(double x)
  {return x;}

  public double d(double x)
  {return 1.0;}

  public double alfa(double x[])
  {return 0.0;}

  public double beta(double x[])
  {return x[1];}

  public double f(double x[])
  {
  //çözümü istenen fonksiyon
  double ff;
  ff=x[1]*x[1]*x[2];
  return ff;
  }
}

class integ_3D
{
public static double[][] gauss_legendre_coefficients(double x1,double x2,int n)
{
        //calculates legendre gauss-coefficients as coefficients of the integral
        //for n terms
        double EPS=3.0e-15;
        int m,j,i;
        double z1,z,xm,xl,pp,p3,p2,p1;
    double a[][]=new double[2][n];//a[0][i]=x[i]   a[1][i]=w[i]
        m=(n+1)/2;
        xm=0.5*(x2+x1);
        xl=0.5*(x2-x1);
        for (i=1;i<=m;i++)  {
                z=Math.cos(Math.PI*((i-0.25)/(n+0.5)));
                do {
                        p1=1.0;
                        p2=0.0;
                        for (j=1;j<=n;j++) {
                                p3=p2;
                                p2=p1;
                                p1=((2.0*j-1.0)*z*p2-(j-1.0)*p3)/j;
                        }
                        pp=n*(z*p1-p2)/(z*z-1.0);
                        z1=z;
                        z=z1-p1/pp;
                } while (Math.abs(z-z1) > EPS);
```

```
                        a[0][i-1]=xm-xl*z;
                        a[0][n-i]=xm+xl*z;
                        a[1][i-1]=2.0*xl/((1.0-z*z)*pp*pp);
                        a[1][n-i]=a[1][i-1];
                                }
return a;
}

public static double I(fdi3 f,int n,int m,int p)
{
//integral f(x)dx
//integral of a function by using gauss-legendre quadrature
//coefficients are pre-calculated for n,m terms for [-1,1]
//band then utilises variable transform
 int i,j;
 double rn[],cn[];
 double rm[],cm[];
 double rp[],cp[];
 rp=new double[p];
 cp=new double[p];
 int nn=Math.max(n,Math.max(m,p));
 int k;
 double x[]=new double[3];
 double a[][]=new double[2][nn];
 int mm=Math.max(Math.max(n,m),p);
 a=gauss_legendre_coefficients(-1.0,1.0,mm);
 rn=rm=rp=a[0];
 cn=cm=cp=a[1];
double h1,h2,J,a1,b1,c1,d1,k1,k2,JX,JY,Q,l1,l2,alfa1,beta1;
a1=f.a;
b1=f.b;
h1=(b1-a1)/2.0;
h2=(b1+a1)/2.0;
J=0;
for(i=0;i<m;i++)
{
JX=0;
x[0]=h1*rm[i]+h2;
d1=f.d(x[0]);
c1=f.c(x[0]);
k1=(d1-c1)/2.0;
k2=(d1+c1)/2.0;
//4=========================
for(j=0;j<n;j++)
{   JY=0;
    x[1]=k1*rn[j]+k2;
    beta1=f.beta(x);
    alfa1=f.alfa(x);
    l1=(beta1-alfa1)/2.0;
    l2=(beta1+alfa1)/2.0;
    for(k=0;k<p;k++)
    {
                x[2]=l1*rp[k]+l2;
                Q=f.f(x);
                JY+=cp[k]*Q;
    }
JX+=cn[j]*l1*JY;
}
//System.out.println("JX="+JX);
//5=========================
J=J+cm[i]*k1*JX;
}
J=h1*J;
return J;
}
```

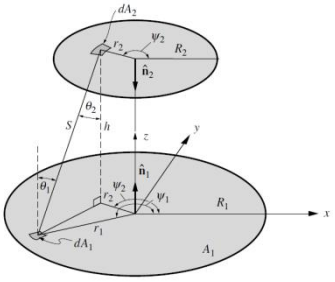
```
public static double I(fdi3 f,int n)
{return I(f,n,n,n);}

public static double I(fdi3 f)
{return I(f,60,60,60);}

  public static void main(String args[]) throws IOException
  {
  f1 ff=new f1(0.0,1.0);                ;
  System.out.println("İntegral="+I(ff));
  }
}
```

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integ_3D
İntegral=0.08333333333333286

> Terminated with exit code 0.
```

Aynı prensip kullanılarak dört boyutlu integraller de çözülebilir.

$$I=\int_{a}^{b}\int_{c(x)}^{d(x)}\int_{\alpha(x,y)}^{\beta(x,y)}\int_{\gamma(x,y)}^{\theta(x,y,z)} f(x,y,z,t)\,dtdzdydx$$

Bir örnek olarak radyasyon şekil faktörü hesaplama problemi göz önüne alalım. Bu problemde birbirine paralel iki diskin şekil fonksiyonu hesaplanacaktır.



Radyasyon şekil faktörü A yüzeyinden yayılan ışıma enerjisinin A2 yüzeyine ulaşan yüzdesini verir. Çapları R1 ve R2, aralarındaki mesafe h olan paralel iki diskin radyasyon şekil faktörü:

$$F_{A1-A2}=\frac{1}{\pi^2 R_1^2}\int_{r1=0}^{R1}\int_{r1=0}^{R2}\int_{\psi_1=0}^{2\pi}\int_{\psi_2=0}^{2\pi}\frac{h^2 r_1 r_2\,d\psi_2 d\psi_1 dr_2 dr_1}{\left[h^2+r_1^2+r_2^2-2r_1 r_2\cos(\psi_1-\psi_2)\right]}$$

Bu integralin analitik çözümü:

$$F_{A1-A2}=\frac{R_2}{R_1}\left(X-\sqrt{X^2-1}\right) \qquad X=\frac{h^2+R_1^2+R_2^2}{2R_1R_2}$$ şeklinde verilebilir. Program 7.7-5 de bu problemin çözüm kodu verilmiş ve sonuç analitik sonuçla karşılaştırılmıştır.

Program 7.7-5 four dimensional Gauss-Legendre integration

```
 //=====================================================
// Numerical Analysis package in java
// 4 boyutlu gauss integrali
// Hamza TAŞ
// =====================================================
import java.io.*;
abstract class fdi4
{
public double a,b;
public fdi4(double ai,double bi)
```

```
{a=ai;b=bi;}
public fdi4()
{a=0;b=1;}
public void setab(double ai,double bi)
{a=ai;b=bi;}
abstract public  double c(double x);
abstract public  double d(double x);
abstract public  double alfa(double x[]);
abstract public  double beta(double x[]);
abstract public  double gama(double x[]);
abstract public  double teta(double x[]);
abstract public  double f(double x[]);
}

class f1 extends fdi4
{ double h,R1,R2;
  public f1(double hi,double R1i, double R2i)
  {h=hi;R1=R1i;R2=R2i;a=0;b=R1;}

  public double c(double x)
  {return 0;}

  public double d(double x)
  {return R2;}

   public double alfa(double x[])
  {return 0;}

  public double beta(double x[])
  {return 2.0*Math.PI;}

  public double gama(double x[])
  {return 0.0;}

  public double teta(double x[])
  {return 2.0*Math.PI;}

  public double f(double x[])
  {
  //çözümü istenen fonksiyon
  double ff;
  double z=(h*h+x[0]*x[0]+x[1]*x[1]-2.0*x[0]*x[1]*Math.cos(x[2]-x[3]));
  ff=1.0/(Math.PI*Math.PI*R1*R1)*h*h*x[0]*x[1]/(z*z);
  return ff;
  }
  public double  F12(double h,double R1,double R2)
  {double X=(h*h+R1*R1+R2*R2)/(2.0*R1*R2);
   double F=R2/R1*(X-Math.sqrt(X*X-1));
   return F;
  }
}

class NA78A
{
public static double[][] gauss_legendre_coefficients(double x1,double x2,int n)
{
        //calculates legendre gauss-coefficients as coefficients of the integral for n terms
        double EPS=3.0e-15;
        int m,j,i;
        double z1,z,xm,xl,pp,p3,p2,p1;
   double a[][]=new double[2][n];//a[0][i]=x[i]   a[1][i]=w[i]
        m=(n+1)/2;
        xm=0.5*(x2+x1);
        xl=0.5*(x2-x1);
        for (i=1;i<=m;i++)  {
                z=Math.cos(Math.PI*((i-0.25)/(n+0.5)));
                do {
```

```
                                p1=1.0;
                                p2=0.0;
                                for (j=1;j<=n;j++) {
                                        p3=p2;
                                        p2=p1;
                                        p1=((2.0*j-1.0)*z*p2-(j-1.0)*p3)/j;
                                }
                                pp=n*(z*p1-p2)/(z*z-1.0);
                                z1=z;
                                z=z1-p1/pp;
                        } while (Math.abs(z-z1) > EPS);
                        a[0][i-1]=xm-xl*z;
                        a[0][n-i]=xm+xl*z;
                        a[1][i-1]=2.0*xl/((1.0-z*z)*pp*pp);
                        a[1][n-i]=a[1][i-1];
                                }
return a;
}
public static double I(fdi4 f,int n,int m,int p,int s)
{
 int i,j,k,h;
 double rn[],cn[];
 double rm[],cm[];
 double rp[],cp[];
 double rs[],cs[];
 rn=new double[n];
 rm=new double[m];
 rp=new double[p];
 rs=new double[s];
 cn=new double[n];
 cm=new double[m];
 cp=new double[p];
 cs=new double[s];
 double a[][]=new double[2][n];
 a=gauss_legendre_coefficients(-1.0,1.0,n);
 rn=a[0];
 cn=a[1];
 a=gauss_legendre_coefficients(-1.0,1.0,m);
 rm=a[0];
 cm=a[1];
 a=gauss_legendre_coefficients(-1.0,1.0,p);
 rp=a[0];
 cp=a[1];
 a=gauss_legendre_coefficients(-1.0,1.0,s);
 rs=a[0];
 cs=a[1];
double x[],h1,h2,J,a1,b1,c1,d1,k1,k2,JX,JY,JZ,Q,l1,l2,alfa1,beta1,gama1,teta1,m1,m2;
x=new double[4];
a1=f.a;
b1=f.b;
h1=(b1-a1)/2.0;
h2=(b1+a1)/2.0;
J=0;
for(i=0;i<m;i++)
{
JX=0;
x[0]=h1*rm[i]+h2;
d1=f.d(x[0]);
c1=f.c(x[0]);
k1=(d1-c1)/2.0;
k2=(d1+c1)/2.0;
      for(j=0;j<n;j++)
      {
              JY=0;
      x[1]=k1*rn[j]+k2;
      beta1=f.beta(x);
      alfa1=f.alfa(x);
```

```
    l1=(beta1-alfa1)/2.0;
    l2=(beta1+alfa1)/2.0;
        for(k=0;k<p;k++)
        {
                JZ=0;
                x[2]=l1*rp[k]+l2;
                teta1=f.teta(x);
                gama1=f.gama(x);
                m1=(teta1-gama1)/2.0;
                m2=(teta1+gama1)/2.0;
                    for (h=0;h<s;h++)
                    {
                            x[3]=m1*rs[h]+m2;
                    Q=f.f(x);
                    JZ+=cs[h]*Q;
            }
       JY+=cp[k]*m1*JZ;
       }
   JX+=cn[j]*l1*JY;
    }
J=J+cm[i]*k1*JX;
}
J=h1*J;
return J;
}
public static double I(fdi4 f,int n)
{return I(f,n,n,n,n);}
public static double I(fdi4 f)
{return I(f,60,60,60,60);}
  public static void main(String args[]) throws IOException
  {
  double h=1.0;
  double R1=1.0;
  double R2=1.0;
  f1 ff=new f1(h,R1,R2);
  System.out.println("FA1-A2= "+I(ff)+" dört boyutlu gauss-Legendre nümerik integrali");
  System.out.println("FA1-A2= "+ff.F12(h,R1,R2)+ " analitik çözüm ");
  }
}
```

```
 ---------- Capture Output ----------
> "c:\java\bin\javaw.exe" NA78A
FA1-A2= 0.3819660112501026 dört boyutlu gauss-Legendre nümerik integrali
FA1-A2= 0.3819660112501051 analitik çözüm

> Terminated with exit code 0.
```

7.8 Verilerin direk olarak integrali

Elimizde veri olarak verilmiş değerler varsa bu tür verilerin integralini almak için iki değişik yaklaşım kullanılabilir. Bu yaklaşımlardan birincisi veriye bir eğri uydurmak ve uydurulan eğrinin integralini almak, diğeri de veri noktalarına örneği Newton-Cotes integral formülleri gibi bir yaklaşımla direk olarak verilen noktaları kullanarak integral hesaplamak olabilir. Ancak Newton cotes integral formülünde belli nokta sayısı olması gerektiğiden eğer bu gerçekleşiyorsa birden fazla formülü beraber kullanmamız gerekebilir.
Eğri uydurma metoduna örnek olarak kübik şerit interpolasyonlu eğri uydurma formüllü integral hesaplamaya göz atalım. Eğer kübik polinomumuz

$$s_k(x)=a_k(x-x_k)+ b_k(x_{k+1}-x)+ [(x-x_k)^3 c_{k+1} +(x_{k+1}-x)^3 c_k]/(6h_k) \quad 1 \leq k \leq n$$

şeklinde verilmiş ise, bu polinomu kullanarak eğri uydurma (katsayıları hesaplama) prosesi eğri uydurma bölümümüzde tanımlanmıştı. Bu formülün integralini alacak olursak :

$$\int_a^b S_k(x)dx = \sum_{x=a}^{b} a_k \frac{(x-x_k)^2}{2} - b_k \frac{(x_{k+1}-x)^2}{2} + c_k \frac{(x-x_k)^4}{24h_k} - d_k \frac{(x_{k+1}-x)^4}{24h_k} \qquad a, x_n, x_{n+1}, ..., b \quad bölgeleri için$$

şerit interpolasyon formülleri kısmi devamlı formül olduğundan aynı kısmi devamlılığı integral işleminde kullanmamız gerekir. Bu yaklaşımın kullanıldığı örnek program aşağıda verilmiştir. programda kullanılan veri $f(x)=x^2$ fonksiyonundan türetilmiştir.

Program 7.8.1 Kübik şerit interpolasyon formülü kullanan direk veri integrasyonu

```
İmport java.io.*;
import javax.swing.*;

class SCO11FB
{

 public static double [] thomas(double a[][],double r[])
 {
 //
 int n=a.length;
 double f[]=new double[n];
 double e[]=new double[n];
 double g[]=new double[n];
 double x[]=new double[n];
 for(int i=0;i<n;i++)     {f[i]=a[i][i];}
 for(int i=0;i<(n-1);i++) {g[i]=a[i][i+1];}
 for(int i=0;i<(n-1);i++) {e[i+1]=a[i+1][i];}
 for(int k=1;k<n;k++)
   {e[k]=e[k]/f[k-1];
    f[k]=f[k]-e[k]*g[k-1];
   }
 for(int k=1;k<n;k++)
   {r[k]=r[k]-e[k]*r[k-1];
   }
 x[n-1]=r[n-1]/f[n-1];
 for(int k=(n-2);k>=0;k--)
   {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
 return x;
 }

 public static double [] thomas(double f[],double e[],double g[],double r[])
 {
 int n=f.length;
 double x[]=new double[n];
 for(int k=1;k<n;k++)
   {e[k]=e[k]/f[k-1];
    f[k]=f[k]-e[k]*g[k-1];
   }
 for(int k=1;k<n;k++)
   {r[k]=r[k]-e[k]*r[k-1];
   }
 x[n-1]=r[n-1]/f[n-1];
 for(int k=(n-2);k>=0;k--)
   {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
 return x;
 }


 public static double [][] cubic_spline(double xi[],double yi[],double c0,double cn)
 {
 // c0 ilk noktadai ikinci türev
```

```
  // cn son oktadaki ikici türev
  int n=xi.length;
  double h[]=new double[n];
  double w[]=new double[n];
  double f[]=new double[n];
  double e[]=new double[n];
  double g[]=new double[n];
  double d[]=new double[n];
  double x[]=new double[n];
  double S[][]=new double[4][n];
  int k;
  for(k=0;k<(n-1);k++)
     {h[k]=xi[k+1]-xi[k];
      w[k]=(yi[k+1]-yi[k])/h[k];
                }
  d[0]=c0;
  d[n-1]=cn;
  for(k=1;k<(n-1);k++)
     {d[k]=6.0*(w[k]-w[k-1]);}
  f[0]=1.0;
  f[n-1]=1.0;
  g[0]=0.0;
  g[n-1]=0.0;
  e[0]=0.0;
  e[n-1]=0.0;
  for(k=1;k<(n-1);k++)
  {f[k]=2.0*(h[k]+h[k-1]);e[k]=h[k-1];g[k]=h[k];}
  S[2]=thomas(f,e,g,d);
  S[3]=xi;
  for(k=0;k<(n-1);k++)
    {S[0][k]=(6.*yi[k+1]-h[k]*h[k]*S[2][k+1])/(6.0*h[k]);
     S[1][k]=(6.*yi[k]-h[k]*h[k]*S[2][k])/(6.0*h[k]);
    }
  return S;
  }

  public static double funcSpline(double S[][],double x)
  {
  int n=S[0].length;
  double xx1=0;
  double xx2=0;
  double y=0;
  double hk=0;
  for(int k=0;k<(n-1);k++)
     {if(S[3][k]<=x && x<=S[3][k+1])
      {hk=(S[3][k+1]-S[3][k]);
       xx1=(x-S[3][k]);
       xx2=(S[3][k+1]-x);
       y=S[0][k]*xx1+S[1][k]*xx2+(xx1*xx1*xx1*S[2][k+1]+xx2*xx2*xx2*S[2][k])/(6.0*hk);
       break;
       }
     }
       if(y==0 && S[3][n-2]<=x )
       {
                  int k=n-2;
                  hk=(S[3][k+1]-S[3][k]);
       xx1=(x-S[3][k]);
       xx2=(S[3][k+1]-x);
       y=S[0][k]*xx1+S[1][k]*xx2+(xx1*xx1*xx1*S[2][k+1]+xx2*xx2*xx2*S[2][k])/(6.0*hk);
       }
  return y;
  }

// Türev formülleri
// ==================
  public static double dfSpline(double xi[],double yi[],double c0,double cn,double x)
```

```
 { //kübik şerit türev formülü
            double S[][]=cubic_spline(xi,yi,c0,cn);
  return dfSpline(S,x);
 }

 public static double dfSpline(double S[][],double x)
 {
  //kübik şerit türev formülü
 int n=S[0].length;
 double xx1=0;
 double xx2=0;
 double y=0;
 double hk=0;
 for(int k=0;k<(n-1);k++)
   {if(S[3][k]<=x && x<=S[3][k+1])
    {hk=(S[3][k+1]-S[3][k]);
     xx1=(x-S[3][k]);
     xx2=(S[3][k+1]-x);
     y=S[0][k]-S[1][k]+(xx1*xx1*S[2][k+1]-xx2*xx2*S[2][k])/(2.0*hk);
     break;
     }
   }
     if(y==0 && S[3][n-2]<=x )
     {
            int k=n-2;
            hk=(S[3][k+1]-S[3][k]);
     xx1=(x-S[3][k]);
     xx2=(S[3][k+1]-x);
     y=S[0][k]*xx1+S[1][k]*xx2+(xx1*xx1*xx1*S[2][k+1]+xx2*xx2*xx2*S[2][k])/(6.0*hk);
     }
 return y;
 }

 public static double  dfSpline(double xi[],double yi[],double x)
 {
   //doğal kübik şerit türev formülü
            double S[][]=cubic_spline(xi,yi,0,0);
   return dfSpline(S,x);
 }
// İntegral formülleri
// ===================
 public static double intSpline(double S[][],double a,double b)
 {
  //kübik şerit türev formülü
 int n=S[0].length;
 double xx1=0;
 double xx2=0;
 double y1,y2;
 double hk=0;
 double toplam=0;
 y1=0;y2=0;
 for(int k=0;k<(n-1);k++)
   { hk=(S[3][k+1]-S[3][k]);
     //System.out.println("x1="+S[3][k]+"x2="+S[3][k+1]+"a="+a+"b="+b);
    if(a>S[3][k+1])
    {toplam=0;//System.out.println("şart 1 toplam = "+toplam);
    }//şart1
            else if(S[3][k]<=a && a<=S[3][k+1] && b>S[3][k+1])//şart 2
     {
     xx1=(a-S[3][k]);
     xx2=(S[3][k+1]-a);
     y1=S[0][k]*xx1*xx1/2.0-S[1][k]*xx2*xx2/2.0+(xx1*xx1*xx1*xx1*S[2][k+1]-xx2*xx2*xx2*xx2*S[2][k])/(24.0*hk);
     xx1=hk;
     xx2=0;
     y2=S[0][k]*xx1*xx1/2.0-S[1][k]*xx2*xx2/2.0+(xx1*xx1*xx1*xx1*S[2][k+1]-xx2*xx2*xx2*xx2*S[2][k])/(24.0*hk);
     toplam+=(y2-y1);
```

```
      //System.out.println("şart 2toplam = "+toplam+"y1="+y1+"y2="+y2);
      }
      else if(S[3][k]<=a && a<=S[3][k+1] && S[3][k]<=b && b<=S[3][k+1])//şart 3
      {
      xx1=(a-S[3][k]);
      xx2=(S[3][k+1]-a);
      y1=S[0][k]*xx1*xx1/2.0-S[1][k]*xx2*xx2/2.0+(xx1*xx1*xx1*xx1*S[2][k+1]-xx2*xx2*xx2*xx2*S[2][k])/(24.0*hk);
      xx1=(b-S[3][k]);
      xx2=(S[3][k+1]-b);
      y2=S[0][k]*xx1*xx1/2.0-S[1][k]*xx2*xx2/2.0+(xx1*xx1*xx1*xx1*S[2][k+1]-xx2*xx2*xx2*xx2*S[2][k])/(24.0*hk);
      toplam+=(y2-y1);
      //System.out.println("şart 3 toplam = "+toplam+"y1="+y1+"y2="+y2);
      }
      else if(a<S[3][k] && b>=S[3][k+1]) //şart 4
      {
      xx1=0;
      xx2=hk;
      y1=S[0][k]*xx1*xx1/2.0-S[1][k]*xx2*xx2/2.0+(xx1*xx1*xx1*xx1*S[2][k+1]-xx2*xx2*xx2*xx2*S[2][k])/(24.0*hk);
      xx1=hk;
      xx2=0;
      y2=S[0][k]*xx1*xx1/2.0-S[1][k]*xx2*xx2/2.0+(xx1*xx1*xx1*xx1*S[2][k+1]-xx2*xx2*xx2*xx2*S[2][k])/(24.0*hk);
      toplam+=(y2-y1);
      //System.out.println("şart 4 toplam = "+toplam+"y1="+y1+"y2="+y2);
      }
      else if(S[3][k]<=b && b<=S[3][k+1])//şart 5
      {xx1=0;
       xx2=hk;
       y1=S[0][k]*xx1*xx1/2.0-S[1][k]*xx2*xx2/2.0+(xx1*xx1*xx1*xx1*S[2][k+1]-xx2*xx2*xx2*xx2*S[2][k])/(24.0*hk);
       xx1=(b-S[3][k]);
       xx2=(S[3][k+1]-b);
       y2=S[0][k]*xx1*xx1/2.0-S[1][k]*xx2*xx2/2.0+(xx1*xx1*xx1*xx1*S[2][k+1]-xx2*xx2*xx2*xx2*S[2][k])/(24.0*hk);
       toplam+=(y2-y1);
       //System.out.println("şart 5 toplam = "+toplam+"y1="+y1+"y2="+y2);
       }
       else break;
    }
  return toplam;
  }

  public static double intSpline(double xi[],double yi[],double c0,double cn,double a,double b)
  {double S[][]=cubic_spline(xi,yi,c0,cn);
   return intSpline(S,a,b);
  }

  public static void main(String args[]) throws IOException
  {
  double x[];
  double y[];
  String s1="a.txt";
  //JFileChooser fc=new JFileChooser();
  //if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName(); }
  double a[][]=Text.readDoubleT(s1);
  x=a[0];
  y=a[1];
  int n=x.length;
  double S[][]=cubic_spline(x,y,2.0,2.0);
  System.out.println("s="+intSpline(S,0.0,10.0));
  }
  }
```

Kübik şerit interpolasyon formülü kullanan direk veri integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" SCO11FB
s=333.3333333333334
```

```
> Terminated with exit code 0.
```

Kübik şerit interpolasyonu kullandığımızda dikkat etmemiz gereken bir husus da uç noktalarda ikinci türev değerlerini bilme zorunluluğudur. Bu değerler yanlış tahmin edildiğinde ve integral iki uç noktaya kadar tüm veri bölgesini kapsadığında yanlış ikinci türev değerleri belli bir hatanın oluşmasına sebep olacaktır. Örneğin test fonksiyonumuzdaki veri $f(x)=x^2$ fonksiyonundan türetilmişti. Bu yüzden f''(x)=2 (sabit) ikinci türevin alması gereken değerdir. Bu degerle integrali aldığımızda, kübik polinom üçüncü derece polinomu tam olarak integre edebildiği için tam sonuç aldık. Şimdi uç şartları olarak f''(x)=0 değeri (doğal kübik şerit) alarak aynı işlemi yapalım. Verimiz 0-10 bölgesini kapsamaktaydı integral sınırları olarak da 0-10 bölgesini alırsak bu durumda sonuç:

Kübik şerit interpolasyon formülü kullanan direk veri integrasyonu, kübik şerit sınır şarları f''(0)=0 ve f''(10)=0 olarak alınmış, veri 0 dan 10 a kadar.

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" SCO11FB
s=333.3334295583782

> Terminated with exit code 0.
```

Görüldüğü gibi sonuçta hata payında belli bir artma oldu, ancak sonuç yine de oldukça iyi. Bu hata payını uç noktalarda ek veri alarak integre ettiğimiz bölgeden uzaklaştırarak azaltabiliriz. Örneğin veriyi -0.3 den 10.3 e kadar aldığımızda ve integral sınırlarımız 0 dan 10 a kadar kaldığında

Kübik şerit interpolasyon formülü kullanan direk veri integrasyonu, kübik şerit sınır şarları f''(0)=0 ve f''(10)=0 olarak alınmış, veri -0 .3 den 10.3 e kadar.

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" SCO11FB
s=333.33333148216684

> Terminated with exit code 0.
```

Şimdi de Newton_cotes integral yaklaşımı kullanan bir örnek verelim.Yukarda da değindiğimiz gibi Newton_Cotes yaklaşımında eğri formülü elde etme zorunluluğu olmadan integrali direk olarak alabiliriz.

Program 7.8.2 9. derece polinom yaklaşımlı(n=8) Newton-Cotes integral formülü kullanan direk veri integrasyonu

```
//=====================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;

class integData
{

public static double newton_cotes9(double xi[],double yi[])
{       //Newton-cotes integral 9 points, n=8
                int n1=xi.length;
                int n=n1/8;
                int n2=n1%8;
                int i1=0;
      double h;
      double h1;
      double sum=0;
      double sum1=0;
      int p,i;
      double x[]=new double[9];
      double f[]=new double[9];
      for(p=0;p<n;p++)
      {  {for(i=0;i<9;i++)
```

```
{i1=p*8+i;x[i]=xi[i1];f[i]=yi[i1];System.out.println("p="+p+"n1="+n1+"n="+n+"n2="+n2+"i1="+i1+"i="+i+"xi="+x[i]+"yi="+f[i]);} }

        h=x[8]-x[0];
        h1=h/8.0;
        sum+=h/28350.0*(989.0*f[0]+5888.0*f[1]-928.0*f[2]+10496.0*f[3]-4540.0*f[4]+
        10496.0*f[5]-928.0*f[6]+5888.0*f[7]+989.0*f[8]);
        //System.out.println("sum="+sum);
      }
      {for(i=0;i<n2;i++)
{i1=p*8+i;x[i]=xi[i1];f[i]=yi[i1];System.out.println("p="+p+"n1="+n1+"n="+n+"n2="+n2+"i1="+i1+"i="+i+"xi="+x[i]+"yi="+f[i]);} }
      if(n2==7)  {h=(x[6]-x[3]);h1=h/3.0;sum1=h*(f[3]+3.0*f[4]+3.0*f[5]+f[6])/8.0;
              h=(x[3]-x[0]);h1=h/3.0;sum1+=h*(f[0]+3.0*f[1]+3.0*f[2]+f[3])/8.0;}
      else if(n2==6) {h=x[n2-1]-x[0];h1=h/5.0;sum1=h*(19.0*f[0]+75.0*f[1]+50.0*f[2]+50.0*f[3]+75.0*f[4]+19.0*f[5])/288.0;}
      else if(n2==5) {h=x[n2-1]-x[0];h1=h/4.0;sum1=h*(7.0*f[0]+32.0*f[1]+12.0*f[2]+32.0*f[3]+7.0*f[4])/90.0; }
      else if(n2==4) {h=x[n2-1]-x[0];h1=h/3.0;sum1=h*(f[0]+3.0*f[1]+3.0*f[2]+f[3])/8.0; }
      else if(n2==3) {h=x[n1]-x[0];h1=h/2.0;sum1=h*(f[0]+4.0*f[1]+f[2])/6.0;  }
      else if(n2==2) {h=x[n2-1]-x[0];h1=h; sum1=h*(f[0]+f[1])/2.0;   }
      //System.out.println("sum="+sum+"sum1="+sum1+"ss="+(sum1+sum));
      return sum+sum1;
}

 public static void main(String args[]) throws IOException
 {
 double x[];
 double y[];
 String s1="a.txt";
 //JFileChooser fc=new JFileChooser();
 //if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName(); }
 double a[][]=Text.readDoubleT(s1);
 x=a[0];
 y=a[1];
 System.out.println("s="+newton_cotes9(x,y));
}
}
```

Çıktı 7.8-4 9. derece polinom yaklaşımlı(n=8) Newton-Cotes integral formülü kullanan direk veri integrasyonu

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" integData
s=333.33333333333337
> Terminated with exit code 0.
```

Newton-Cotes yaklaşımında eğri uydurma olmadığından (fonksiyon değerleri direk olarak verilerden alındığından) veri bölgesi ve integral bölgesi aynı olmak zorundadır. Elimizde daha fazla veri olduğunda sadece integral alacağımız bölgenin verisini alarak geri kalan veriyi silmemiz gerekir, veya daha düşük bir Newton-cotes yaklaşımıyla geri kalan bölgenin integrali alınabilir.

7.9 Türev formülleri

Türev hesaplamak için fark denklemlerinden yararlanabiliriz. Fark denklemleri temel olarak Taylor formülü açılımlarını kullanır.

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \frac{f^{(4)}(x_i)}{4!}h^4 + \ldots$$

Bu eşitlikten birinci türev çözülürse

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{f''(x_i)}{2}h - \frac{f^{(3)}(x_i)}{3!}h^2 - \frac{f^{(4)}(x_i)}{4!}h^3 + O(h^4)\ldots$$

bulunur. Veya

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h)$$

O(h) terimi hata miktarını bildirmektedir. Hata h ile orantılıdır. Yani h küçüldükçe hata h ile orantılı olarak küçülecektir. Aynı şekilde ikinci türevleri yazarsak

$$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2} + O(h)$$

Hata terimini azaltmak için ikinci türevi orijinal denkleme geri koyacak olursak

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{2h^2} h + O(h^2)$$

$$f'(x_i) = \frac{2f(x_{i+1}) - 2f(x_i) - f(x_{i+2}) + 2f(x_{i+1}) - f(x_i)}{h} + O(h^2)$$

$$f'(x_i) = \frac{-f(x_{i+2}) + 4f(x_{i+1}) - 3f(x_i)}{2h} + O(h^2)$$ halini alacaktır.

Fark denklemlerini ileriye doğru veya geriye doğru oluşturabiliriz. Geriye doğru fark denklemini

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + O(h)$$ şeklinde yazabiliriz. İleri ve geriye doğru fark denklemlerinden merkezi fark denklemini oluşturabiliriz.

$$f'(x_i) = \frac{f'(x_i)ileri + f'(x_i)geri}{2} = \frac{\frac{f(x_i) - f(x_{i-1})}{h}}{2} + \frac{\frac{f(x_i) - f(x_{i-1})}{h}}{2} = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2)$$

Türev fark denklemlerinin katsayılarını bir tablo olarak listelersek:

Tablo 7.9-1 Türev fark denklemleri

Merkezi farklar

Türev	Terim sayısı	−4	−3	−2	−1	0	1	2	3	4
1(f')	2				−1/2	0	1/2			
	4			1/12	−2/3	0	2/3	−1/12		
	6		−1/60	3/20	−3/4	0	3/4	−3/20	21916	
	8	1/280	−4/105	1/5	−4/5	0	4/5	−1/5	4/105	−1/280
2(f'')	2				1	−2	1			
	4			−1/12	4/3	−5/2	4/3	−1/12		
	6		32874	−3/20	3/2	−49/18	3/2	−3/20	32874	
	8	−1/560	8/315	−1/5	8/5	−205/72	8/5	−1/5	8/315	−1/560
3(f''')	2			−1/2	1	0	−1	1/2		
	4		41122	−1	13/8	0	−13/8	1	−1/8	
	6	−7/240	41185	−169/120	61/30	0	−61/30	169/120	−3/10	7/240
4($f^{(4)}$)	2			1	−4	6	−4	1		
	4		−1/6	2	−13/2	28/3	−13/2	2	−1/6	
	6	7/240	−2/5	169/60	−122/15	91/8	−122/15	169/60	−2/5	7/240

İleriye doğru farklar

Türev	Terim sayısı	0	1	2	3	4	5	6	7	8
1(f')	1	−1	1							
	2	−3/2	2	−1/2						
	3	−11/6	3	−3/2	1/3					
	4	−25/12	4	−3	4/3	−1/4				
	5	−137/60	5	−5	10/3	−5/4	41030			
	6	−49/20	6	−15/2	20/3	−15/4	41035	−1/6		
2(f'')	1	1	−2	1						
	2	2	−5	4	−1					
	3	35/12	−26/3	40958	−14/3	41254				
	4	41014	−77/6	107/6	−13	61/12	−5/6			

	5	203/45	−87/5	117/4	−254/9	33/2	−27/5	137/180		
	6	469/90	−223/10	879/20	−949/18	41	−201/10	1019/180	−7/10	
3(f’’’)	1	−1	3	−3	1					
	2	−5/2	9	−12	7	−3/2				
	3	−17/4	71/4	−59/2	49/2	−41/4	7/4			
	4	−49/8	29	−461/8	62	−307/8	13	−15/8		
	5	−967/120	638/15	−3929/40	389/3	−2545/24	268/5	−1849/120	29/15	
	6	−801/80	349/6	−18353/120	2391/10	−1457/6	4891/30	−561/8	527/30	−469/240
4($f^{(4)}$)	1	1	−4	6	−4	1				
	2	3	−14	26	−24	11	−2			
	3	35/6	−31	137/2	−242/3	107/2	−19	41077		
	4	28/3	−111/2	142	−1219/6	176	−185/2	82/3	−7/2	
	5	1069/80	−1316/15	15289/60	−2144/5	10993/24	−4772/15	2803/20	−536/15	967/240

Tablo 7.9.1 Türev (fark denklemleri kullanılması)

```
//=====================================================
// Numerical Analysis package in java
// example to show utilisation of integration (integral)
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;

class f1 extends f_x
{ double func(double x)
  {return x*x*x*x*x;}
}

class dif1
{
public static double derivative(f_x f,double x)
{
double h0=0.001;
int i,m;
int n=7;
double T[][]=new double[n][n];
double h[];
h=new double[n];
h[0]=h0;
double r=0.5;
for( i=1;i<n;i++) {h[i]=h0*Math.pow(r,i);}
for(i=0;i<n;i++)  {T[i][0]=dfdx(f,x,h[i]);}
for(m=1;m<n;m++)
{ for(i=0;i<n-m;i++)
  {T[i][m]=(h[i]*h[i]*T[i+1][m-1]-h[i+m]*h[i+m]*T[i][m-1])/(h[i]*h[i]- h[i+m]*h[i+m]);}
}
double xx=T[0][n-1];
return xx;
}

public static double dfdx_richardson(f_x f,double x)
{
double h0=0.01;
double h=h0;
int i,m;
int n=10;
double T[][]=new double[n][n];
for(i=0;i<n;i++)  {T[i][0]=dfdx(f,x,h);h/=2.0;}
for(m=1;m<n;m++)
{ for(i=0;i<n-m;i++)
  {T[i][m]=(4.0/3.0*T[i+1][m-1]-1.0/3.0*T[i][m-1]);}
}
double xx=T[0][n-1];
```

```
return xx;
}

public static double dfdx(f_x f,double x)
{ double h=0.00001;
  return dfdx(f,x,h);
}
public static double dfdx(f_x f,double x,double h)
{ double hh=1.0/h;
  return (-f.func(x+2.0*h)+8.0*f.func(x+h)-8.0*f.func(x-h)+f.func(x-2.0*h))/12.0*hh;
}

public static double d2fdx2(f_x f,double x)
{ double h=0.00001;
  return d2fdx2(f,x,h);
}
public static double d2fdx2(f_x f,double x,double h)
{
  double hh=1.0/h;
            return (-f.func(x+2.0*h)+16.0*f.func(x+h)-30.0*f.func(x)+16.0*f.func(x-h)-f.func(x-2.0*h))/(12.0)*hh*hh;}

public static double d3fdx3(f_x f,double x)
{ double h=0.00001;
  return d3fdx3(f,x,h);}

public static double d3fdx3(f_x f,double x,double h)
{  double hh=1.0/h;
            return (-f.func(x+3.0*h)+8.0*f.func(x+2.0*h)-13.0*f.func(x+h)+13.0*f.func(x-h)-8.0*f.func(x-2.0*h)+f.func(x-
3.0*h))/(8.0)*hh*hh*hh;}

  public static void main(String args[])
  {
  //türev formülleri
  f1 b1=new f1();
  System.out.println("turev dfdx    : "+dfdx(b1,1.0));
  System.out.println("turev d2dfx2  : "+d2fdx2(b1,1.0));
  System.out.println("turev d3dfx3  : "+d3fdx3(b1,1.0));

  System.out.println("turev dfdx    : "+derivative(b1,1.0));
  System.out.println("turev dfdx    : "+dfdx_richardson(b1,1.0));
  }
}
```

Türev (fark denklemleri kullanılması)

```
Turev dfdx    : 4.999999999989272
turev d2dfx2  : 20.00000674332961
turev d3dfx3  : 60.14633235906783
turev dfdx    : 5.000000000009831
turev dfdx    : 4.99999999994733
```

7.10 Türev işleminde Richardson extrapolasyonu

İntegral hesaplarında gördüğümüz Richardson extrapolasyon formülü türev hesaplarına da uygulanabilir**.**
İntegral için Richardson Interpolasyon formülü

$I \cong I(h_2) + \frac{1}{(h_1/h_2)^2 - 1}[I(h_2) - I(h_1)]$ şeklinde idi. Burada h_2, h_1 den küçük olmalıdır. Eğer $h_2=h_1/2$ olarak verilirse

$I \cong \frac{4}{3}I(h_2) - \frac{1}{3}I(h_1)]$ şeklini alır. Aynı formülü türev için yazacak olursak :

$$D \cong \frac{4}{3} D(h_2) - \frac{1}{3} D(h_1)]$$

İnterpolasyon başlangıcı için türev yaklaşımlarından birisi kullanılabilir.
İteratif formülümüz genel olarak yazılırsa

$$D_{j,k} \cong \frac{4^{k-1} D_{j+1,k-1} - D_{j,k-1}}{4^{k-1} - 1}$$ haline gelir.

7.11 Türev yaklaşımında Pad'e formülü

Türev fark formülleri görüldüğü gibi Taylor serisinden türetilmiştir. Taylor serisinde eğer belli bir noktada fonksiyon değerleri verilmişse türev değerleri de denkleme eklenebilir. Eğer f'_j, f'_{j+1} ve f'_{j-1} kullanan hassa bir denklem oluşturmak istiyorsak önce denklemimizi bu terimlerin olduğu bir seri olarak düşünürüz.

$$f'_j + a_0 f_j + a_1 f_{j+1} + a_2 f_{j-1} + a_3 f'_{j+1} + a_4 f'_{j-1} = O(?)$$

PAD'E METODU İÇİN TAYLOR TABLOSU

	f_j	f_j'	f_j''	$f_j^{(3)}$	$f_j^{(4)}$	$f_j^{(5)}$
f_j'	0	1	0	0	0	0
$a_0 f_j$	a_0	0	0	0	0	0
$a_1 f_{j+1}$	a_1	$a_1 h$	$a_1 h^2/2$	$a_1 h^3/6$	$a_1 h^4/24$	$a_1 h^5/120$
$a_2 f_{j-1}$	a_2	$-a_2 h$	$a_2 h^2/2$	$-a_2 h^3/6$	$a_2 h^4/24$	$-a_2 h^5/120$
$a_3 f_{j+1}$	0	a_3	$a_3 h$	$a_3 h^2/2$	$a_3 h^3/6$	$a_3 h^4/24$
$a_4 f_{j-1}$	0	a_4	$-a_4 h$	$a_4 h^2/2$	$-a_4 h^3/6$	$a_4 h^4/24$

Şimdi her satırı toplayacağız ve alt türev dereceli terimlerin katsayılarını mümkün olduğunca sıfırlayacağız.

$a_0 + a_1 + a_2 = 0$

$a_1 h - a_2 h + a_3 + a_4 = -1$

$$a_1 \frac{h^2}{2} + a_2 \frac{h^2}{2} + a_3 h - a_4 h = 0$$

$$a_1 \frac{h}{3} - a_2 \frac{h}{3} + a_3 + a_4 = 0$$

$$a_1 \frac{h}{4} + a_2 \frac{h}{4} + a_3 - a_4 = 0$$

Görüldüğü gibi lineer bir denklem sistemi oluştu. Bu denklemi çözerek katsayıları bulabiliriz.

a_0=0 $a_1 = -\frac{3}{4h}$ $a_2 = \frac{3}{4h}$ $a_3 = a_4 = \frac{1}{4}$

Bu katsayılar bize sayısal türev için Padé Formülünü verir.

$$f'_{j+1} + f'_{j-1} + 4 f'_j = \frac{3}{h}(f_{j+1} - f_{j-1}) + \frac{h^4}{30} f_j^{(5)}$$

Bu denklemin çözülebilmesi sınır noktalarında için iki ek denkleme daha gereksinme vardır. Ek denklemler olarak

$$f'_0 + 2 f'_1 = \frac{1}{h}(-\frac{5}{2} f_0 + 2 f_1 + \frac{1}{2} f_2)$$

$$f'_n + 2 f'_{n-1} = \frac{1}{h}(\frac{5}{2} f_n + 2 f_{n-1} - \frac{1}{2} f_{n-2})$$

Denklemlerini alırsak toplam denklem sistemimatris formunda :

$$\begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 \\ .. & .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. & .. \\ & & & & 1 & 4 & 1 \\ & & & & & 2 & 1 \end{bmatrix} \begin{bmatrix} f_0' \\ f_1' \\ f_2' \\ .. \\ .. \\ f_{n-1}' \\ f_n' \end{bmatrix} = \begin{bmatrix} -\frac{5}{2} f_0 + 2 f_1 + \frac{1}{2} f_2 \\ 3(f_2 - f_0) \\ 3(f_3 - f_1) \\ .. \\ .. \\ 3(f_n - f_{n-1}) \\ \frac{5}{2} f_n - 2 f_{n-1} - \frac{1}{2} f_{n-2} \end{bmatrix}$$

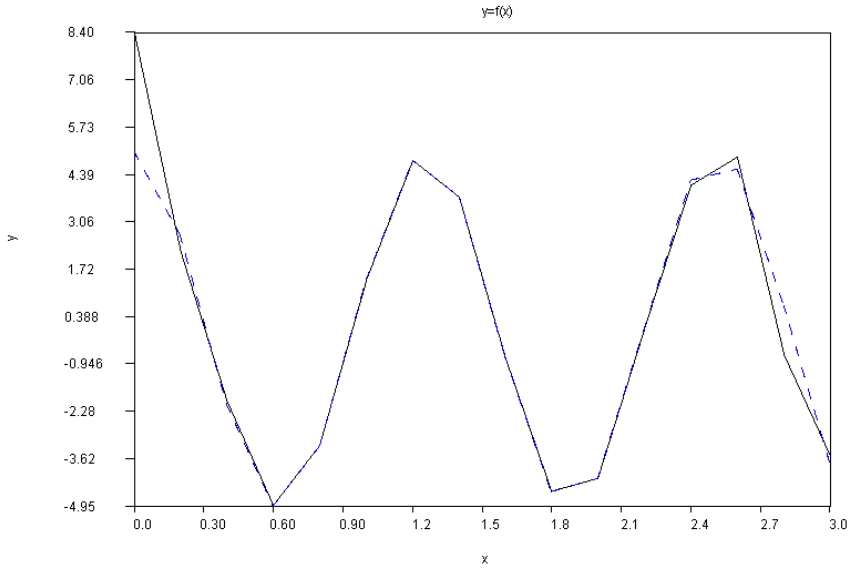
Şekini alır. 3 lü bant matris olan bu sistemi Thomas algoritması kullanarak çözebiliriz.

Program 7.11.1 Padé formülüyle türev hesabı

```
import java.io.*;
import javax.swing.*;

class f1 extends f_x
{ double func(double x)
{return Math.sin(5.0*x);}
}
class f2 extends f_x
{//exact derivative of the function
double func(double x)
{return 5.0*Math.cos(5.0*x);}
}

class NA81A
{
public static double [] thomas(double a[][],double r[])
{
//Padé formula
int n=a.length;
double f[]=new double[n];
double e[]=new double[n];
double g[]=new double[n];
double x[]=new double[n];
for(int i=0;i<n;i++)     {f[i]=a[i][i];}
for(int i=0;i<(n-1);i++) {g[i]=a[i][i+1];}
for(int i=0;i<(n-1);i++) {e[i+1]=a[i+1][i];}
for(int k=1;k<n;k++)
{e[k]=e[k]/f[k-1];
f[k]=f[k]-e[k]*g[k-1];
}
for(int k=1;k<n;k++)
{r[k]=r[k]-e[k]*r[k-1];
}
x[n-1]=r[n-1]/f[n-1];
for(int k=(n-2);k>=0;k--)
{x[k]=(r[k]-g[k]*x[k+1])/f[k];}
return x;
}

public static double [] thomas(double f[],double e[],double g[],double r[])
{
```

```
int n=f.length;
double x[]=new double[n];
for(int k=1;k<n;k++)
{e[k]=e[k]/f[k-1];
f[k]=f[k]-e[k]*g[k-1];
}
for(int k=1;k<n;k++)
{r[k]=r[k]-e[k]*r[k-1];
}
x[n-1]=r[n-1]/f[n-1];
for(int k=(n-2);k>=0;k--)
{x[k]=(r[k]-g[k]*x[k+1])/f[k];}
return x;
}
public static double[][] df(f_x ff,double x0,double xn, int n)
{
int nn=n+1;
double h=(xn-x0)/n;
double y[]=new double[nn];
double x[]=new double[nn];
double f[]=new double[nn];
double g[]=new double[nn];
double e[]=new double[nn];
for(int i=0;i<=n;i++)
{x[i]=x0+h*i;}

for(int i=0;i<=n;i++)
{
if(i==0) {f[i]=1;}
else if(i==n) {f[i]=1;}
else {f[i]=4;}
if(i==1) {g[i]=2;}
else if(i==n) {g[i]=0;}
else {g[i]=1;}
if(i==(n-1)) {e[i]=2;}
else if(i==0) {e[i]=0;}
else {e[i]=1;}
if(i==0) {y[i]=1.0/h*(-2.5*ff.func(x[0])+2.0*ff.func(x[1])+0.5*ff.func(x[2]));}
else if(i==n) {y[i]=1.0/h*(2.5*ff.func(x[n])-2.0*ff.func(x[n-1])-0.5*ff.func(x[n-1]));}
else {y[i]=(1.0/h*3.0*(ff.func(x[i+1])-ff.func(x[i-1])));}
}
double a[][]=new double[3][nn];
a[0]=x;
a[1]=y;
a[2]=thomas(f,e,g,y);
return a;
}
public static void main(String args[]) throws IOException
{
double a[][]=df(new f1(),0.0,3.0, 15);
Plot pp=new Plot(a[0],a[2]);
int n=a[0].length;
double y[]=new double[n];
f2 dy=new f2();
for(int i=0;i<n;i++)
{y[i]=dy.func(a[0][i]);
System.out.println(a[0][i]+" "+a[2][i]+" "+y[i]);
}
```

```
pp.addData(a[0],y);
pp.plot();
}
}
```



(bolum7_009.jpg, Resimaltı:Pade türev yaklaşımı)

7.12 Verilerin direk olarak türevinin alınması

Elimizde veri olarak verilmiş değerler varsa bu tür verilerin türevini almak için iki değişik yaklaşım kullanılabilir. Bu yaklaşımlardan birincisi veriye bir eğri uydurmak ve uydurulan eğrinin türevini almak, diğeri de veri noktalarına tablo **7.9-1 Türev fark denklemleri** gibi bir yaklaşımla direk olarak verilen noktaları kullanarak türev hesaplamak olabilir.

Önce fonksiyon tanımlama prosesinde interpolasyon formüllerini kullanabiliriz. Örneğin İkinci derece Lagrange interpolasyonu kullanırsak Fonksiyon :

$$f_2(x) = \frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} f(x_{i-1}) + \frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})} f(x_i) + \frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)} f(x_{i+1})$$

türev fonksiyonu ise

$$f_2'(x) = \frac{(2x-x_i-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} f(x_{i-1}) + \frac{(2x-x_{i-1}-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})} f(x_i) + \frac{(2x-x_{i-1}-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)} f(x_{i+1})$$

formunu alır. Bu formulasyonu başka dereceler ve başka interpolasyon formülleri kullanarak ta uygulayabiliriz.

Fonksiyon tanımlama (eğri uydurma) metodlarına diğer bir örnek olarak kübik şerit interpolasyonlu eğri uydurma formüllü türev hesaplamaya göz atalım. Eğer kübik polinomumuz

$$\mathbf{s_k(x)=a_k(x-x_k)+ b_k(x_{k+1}-x)+ [(x-x_k)^3 c_{k+1} +(x_{k+1}-x)^3 c_k]/(6h_k) \quad 1 \le k \le n}$$

şeklinde verilmiş ise, bu polinomu kullanarak eğri uydurma (katsayıları hesaplama) prosesi eğri uydurma bölümümüzde tanımlanmıştı. Bu formülün türevini alacak olursak :

$$\mathbf{s'_k(x)=a_k- b_k+ [(x-x_k)^2 c_{k+1} -(x_{k+1}-x)^2 c_k]/(2h_k) \quad 1 \le k \le n}$$

şeklini alır. Şerit interpolasyon formülleri kısmi devamlı formül olduğundan aynı kısmi devamlılığı türev işleminde de kullanmamız gerekir. Bu yaklaşımın kullanıldığı örnek program aşağıda verilmiştir. programda kullanılan veri $f(x)=x^2$ fonksiyonundan türetilmiştir.

Program 7.12.1 Şerit interpolasyon formülü kullanan Türev programı

```
import java.io.*;
import javax.swing.*;

class SCO11FA
{

  public static double [] thomas(double a[][],double r[])
  {
  //
  int n=a.length;
  double f[]=new double[n];
  double e[]=new double[n];
  double g[]=new double[n];
  double x[]=new double[n];
  for(int i=0;i<n;i++)     {f[i]=a[i][i];}
  for(int i=0;i<(n-1);i++) {g[i]=a[i][i+1];}
  for(int i=0;i<(n-1);i++) {e[i+1]=a[i+1][i];}
  for(int k=1;k<n;k++)
     {e[k]=e[k]/f[k-1];
      f[k]=f[k]-e[k]*g[k-1];
     }
  for(int k=1;k<n;k++)
     {r[k]=r[k]-e[k]*r[k-1];
     }
  x[n-1]=r[n-1]/f[n-1];
  for(int k=(n-2);k>=0;k--)
     {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
  return x;
  }

  public static double [] thomas(double f[],double e[],double g[],double r[])
  {
  int n=f.length;
  double x[]=new double[n];
  for(int k=1;k<n;k++)
     {e[k]=e[k]/f[k-1];
      f[k]=f[k]-e[k]*g[k-1];
     }
  for(int k=1;k<n;k++)
     {r[k]=r[k]-e[k]*r[k-1];
     }
  x[n-1]=r[n-1]/f[n-1];
  for(int k=(n-2);k>=0;k--)
     {x[k]=(r[k]-g[k]*x[k+1])/f[k];}
  return x;
  }


  public static double [][] cubic_spline(double xi[],double yi[],double c0,double cn)
  {
  // c0 ilk noktadai ikinci türev
  // cn son oktadaki ikici türev
  int n=xi.length;
  double h[]=new double[n];
  double w[]=new double[n];
  double f[]=new double[n];
  double e[]=new double[n];
  double g[]=new double[n];
  double d[]=new double[n];
```

```
double x[]=new double[n];
double S[][]=new double[4][n];
int k;
for(k=0;k<(n-1);k++)
   {h[k]=xi[k+1]-xi[k];
    w[k]=(yi[k+1]-yi[k])/h[k];
                }
d[0]=c0;
d[n-1]=cn;
for(k=1;k<(n-1);k++)
   {d[k]=6.0*(w[k]-w[k-1]);}
f[0]=1.0;
f[n-1]=1.0;
g[0]=0.0;
g[n-1]=0.0;
e[0]=0.0;
e[n-1]=0.0;
for(k=1;k<(n-1);k++)
{f[k]=2.0*(h[k]+h[k-1]);e[k]=h[k-1];g[k]=h[k];}
S[2]=thomas(f,e,g,d);
S[3]=xi;
for(k=0;k<(n-1);k++)
  {S[0][k]=(6.*yi[k+1]-h[k]*h[k]*S[2][k+1])/(6.0*h[k]);
   S[1][k]=(6.*yi[k]-h[k]*h[k]*S[2][k])/(6.0*h[k]);
  }
return S;
}

public static double funcSpline(double S[][],double x)
{
int n=S[0].length;
double xx1=0;
double xx2=0;
double y=0;
double hk=0;
for(int k=0;k<(n-1);k++)
   {if(S[3][k]<=x && x<=S[3][k+1])
    {hk=(S[3][k+1]-S[3][k]);
     xx1=(x-S[3][k]);
     xx2=(S[3][k+1]-x);
     y=S[0][k]*xx1+S[1][k]*xx2+(xx1*xx1*xx1*S[2][k+1]+xx2*xx2*xx2*S[2][k])/(6.0*hk);
     break;
     }
   }
     if(y==0 && S[3][n-2]<=x )
     {
                int k=n-2;
                hk=(S[3][k+1]-S[3][k]);
     xx1=(x-S[3][k]);
     xx2=(S[3][k+1]-x);
     y=S[0][k]*xx1+S[1][k]*xx2+(xx1*xx1*xx1*S[2][k+1]+xx2*xx2*xx2*S[2][k])/(6.0*hk);
     }
return y;
}

public static double dfSpline(double xi[],double yi[],double c0,double cn,double x)
{ //kübik şerit türev formülü
              double S[][]=cubic_spline(xi,yi,c0,cn);
  return dfSpline(S,x);
}

public static double dfSpline(double S[][],double x)
{
  //kübik şerit türev formülü
int n=S[0].length;
double xx1=0;
```

```
 double xx2=0;
 double y=0;
 double hk=0;
 for(int k=0;k<(n-1);k++)
   {if(S[3][k]<=x && x<=S[3][k+1])
    {hk=(S[3][k+1]-S[3][k]);
     xx1=(x-S[3][k]);
     xx2=(S[3][k+1]-x);
     y=S[0][k]-S[1][k]+(xx1*xx1*S[2][k+1]-xx2*xx2*S[2][k])/(2.0*hk);
     break;
     }
   }
     if(y==0 && S[3][n-2]<=x )
     {
                int k=n-2;
                hk=(S[3][k+1]-S[3][k]);
     xx1=(x-S[3][k]);
     xx2=(S[3][k+1]-x);
     y=S[0][k]*xx1+S[1][k]*xx2+(xx1*xx1*xx1*S[2][k+1]+xx2*xx2*xx2*S[2][k])/(6.0*hk);
     }
 return y;
 }

public static double  dfSpline(double xi[],double yi[],double x)
{
  //doğal kübik şerit türev formülü
             double S[][]=cubic_spline(xi,yi,0,0);
  return dfSpline(S,x);
}

 public static void main(String args[]) throws IOException
 {
 double x[];
 double y[];
 String s1="a.txt";
 //JFileChooser fc=new JFileChooser();
 //if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName(); }
 double a[][]=Text.readDoubleT(s1);
 x=a[0];
 y=a[1];
 int n=x.length;
 double S[][]=cubic_spline(x,y,2.0,2.0);
 System.out.println(dfSpline(S,2.5));
 }
 }
```

Çıktı 7.9-2 Şerit interpolasyon formülü kullanan Türev programı

```
---------- Capture Output ----------
> "C:\co\java\bin\java.exe" SCO11FA
4.999999999999999
> Terminated with exit code 0.
```

7.13 Problemler

PROBLEM 1

$y=f(x)=1/(1+x^2)$ fonksiyonunun –1 ile 1 sınırları arasındaki integralini 3 noktalı Gauss - Legendre integral formülü kullanarak hesaplayınız

Not: 3 lüGauss_Legendre formülü katsayıları(katsayılar kitapta da mevcuttur) :

3 lü formül ağırlık faktörleri 3 lü formül fonksiyon noktaları :

3 lü formül ağırlık faktörleri	3 lü formül fonksiyon noktaları :
c_0 0.555555555555553	x_0 0.774596669241483
c_1 0.888888888888889	x_1 0.000000000000000
c_2 0.555555555555553	x_2 0.774596669241483

PROBLEM 2

$\int_{-3}^{3} \frac{2}{1+2x^2} dx$ integralini n=2 adım büyüklüğü için Bole kuralı kullanarak çözünüz

Ek bilgi : bir adım büyüklüğü (n=1) için Bole kuralı :
I=(b-a)*[7*f(x0)+32*f(x1)+12*f(x2)+32*f(x3)+7*f(x4)]/90
Adım büyüklüğü h=(b-a)/n

PROBLEM 3

$$f_{0-\lambda} = \frac{c_1}{\sigma} \int_{x=0}^{\lambda T} \frac{dx}{x^5 \left[\exp(\frac{c_2}{x}) - 1 \right]}$$

fonksiyonu radyasyon ısı değişiminde siyah cisim radyasyon fonksiyonu olarak bilinir. Formüldeki katsayıların değerleri c_1 =3.743 x10^8 W.□m^4/m^2, c_2 =1.4387x10^4 □m.K, □=5.67x10^{-8} W/(m^2.K^4) dür.
Gauss-Legendre integral formülünü kullanarak verilen bir □T değeri için siyah cisim radyasyon faktörünü hesaplayacak bir metod yazınız. □T nin 5000 ile sonsuz arasındaki değerleri için bir tablo oluşturunuz.

PROBLEM 4

$$I = \int_{x=0}^{1} \int_{y=0}^{1} \frac{dy\,dx}{(1+y)^2}$$

integralini

a) monte-carlo metodu kullanarak çözünüz.
b) 2 boyutlu gauss integrali kullanarak çözünüz
c) 2 boyutlu simpson 1/3 integrali kullanarak çözünüz
d) 2 boyutlu 4. derece poliom(booole) kuralını kullanan bir program geliştiriniz ve çözünüz

PROBLEM 5

$I = \int_{x=0}^{1} \frac{1}{1+x^2} dx$ integralinin

a) tam değerini analitik metod kullanarak çözünüz.
b) Trapez kuralı ile n=2 için çözünüz (iki alt parçaya ayırınız). Yüzde hatayı hesaplayınız.
c) Trapez kuralı ile n=4 için çözünüz (dört alt parçaya ayırınız). Yüzde hatayı hesaplayınız.

PROBLEM 6

$I = \int_{x=0}^{2} \frac{1}{\sqrt{1+x^2}} dx$ integralini

Gauss-Legendre integrali ile hesaplayınız.

PROBLEM 7

$$f(x) = \ln\left(\frac{1}{1+x^2}\right)$$ fonksiyonunu göze alınız

a) bu fonksiyonun x=1 noktasındaki analitik türevini hesaplayınız.
b) 3 noktalı merkezi fark denklemi ve h=0.1 için türev değerini hesaplayınız
c) 3 noktalı merkezi fark denklemi ve h=0.05 için türev değerini hesaplayınız
d) Richardson interpolasyonu h1=0.5 ve h2=0.05 kullanarak türev değerini hesaplayınız

PROBLEM 8

$$f(x) = \int_{-1}^{1} x.\exp(x)\ dx$$ fonksiyonunu göze alınız

a) Trapez integrali
b) Simpson 3/8 integrali
c) Gauss_legendre integrali
d) Romberg inregrali integrali

Denklemlerini kullanarak integralini hesaplayınız.

PROBLEM 9

$$f(x) = \int_{0}^{1} \sin(1-x+x^2)\ dx$$ fonksiyonunu göze alınız

e) Trapez integrali
f) boole integrali
g) Gauss_legendre integrali
h) Romberg inregrali integrali

Denklemlerini kullanarak integralini hesaplayınız.

PROBLEM 10

$$f(x) = \int_{-1}^{1} \frac{\ln(1+x^2)}{\sqrt{1-x^2}}\ dx$$ fonksiyonunu göze alınız

i) Trapez integrali
j) Simpson 1/3 integrali
k) Gauss_legendre integrali
l) Romberg integrali integrali

Denklemlerini kullanarak integralini hesaplayınız.

PROBLEM 11

X	**0.99**	**2.10**	**3.22**	**4.40**	**5.70**	**7.12**	**8.01**	**8.37**	**9.32**	**9.98**
f(x)	**4.90**	**5.70**	**4.20**	**7.04**	**8.31**	**7.82**	**5.97**	**7.01**	**6.68**	**4.79**

Verisi verilmiştir. Bu veriyle tanımlanan fonksiyonun 0 ile 10 aralığındaki integralini bulunuz.

PROBLEM 12

Program 11 de verilen veriyi kullanarak x=5 noktasındaki türevini hesaplayınız.

PROBLEM 13

Bir kanalın derinlik profili şekilde gösterilmektedir. kanalın derinlik ve su hızı verisi aşağıdaki tabloda verilmiştir.

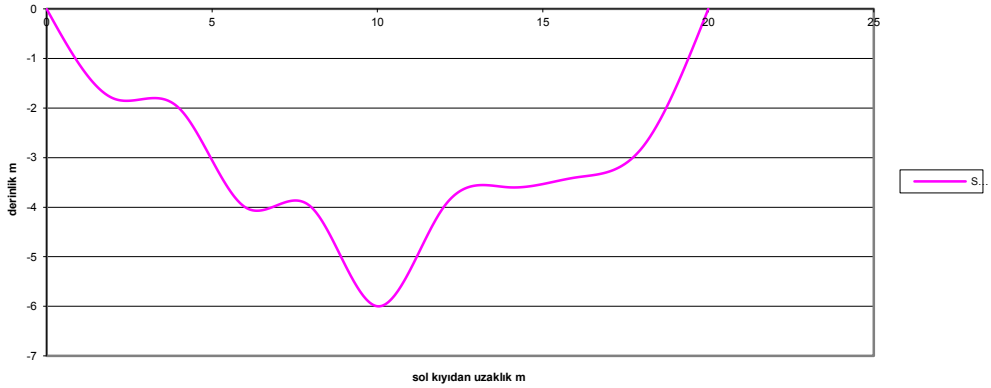
x, m	0	2	4	6	8	10	12	14	16	18	20
H, m	0	1.8	2	4	4	6	4	3.6	3.4	2.8	0
U m/s	0	0.03	0.045	0.055	0.065	0.12	0.07	0.06	0.05	0.04	0

Kanalın kesit alanı bulunmak istenilir ise :

$A_c = \int_0^B H(x)dx$ olacaktır. Kanalın debisi bulunmak istenilir ise :

$Q = \int_0^B U(x)H(x)dx$ olacaktır. Burada U suyun x noktasında ölçülen hızıdır(m/s). Tüm kanal için suyun debisini (Q) ve kesit alanını (A_c) hesaplayınız.

Not : veriyi arttırmak için kübik şerit interpolasyonu uygulanabilir.



(bolum7_010.jpg, **Problem 13 kanal kesiti**)

PROBLEM 14

Katı bir cisimdeki sıcaklığın dağılımı

$\frac{dT}{dt} = -k(T - T_a)$ denklemiyle ifade edilebilir. Burada T cismin sıcaklığı, T_a = yüzey sıcaklığı (sabit). Olarak verilmiştir. T=90 C sıkaklığındaki küçük bir metal cisim, T_a =20 C sabit sıcaklığında bir su banyosuna atılırsa zaman içindeki sıcaklık değişimi

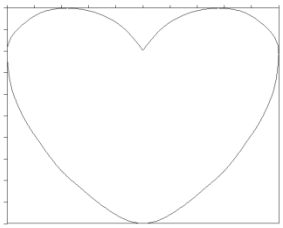
Zaman , t, s	0	5	10	15	20	25	30	35
Sıcaklık T C	90	55	40	28	24	22	21.5	20.6

Tablosundaki değerlere göre değişmektedir. k değerini hesaplayınız.
Not: k değeri sabit mi olacaktır? yoksa zamana göre değişecek midir?

PROBLEM 15

Bir kalp şeklinin koordinatları tabloda verilmiştir. Verilen şeklin alanınını bulunuz.

Xi	Yi	xi	yi	xi	yi	xi	Yi
241	127	331	157	215	246	149	120
258	107	323	178	199	232	160	106
283	96	312	197	184	217	175	97
307	97	298	217	170	197	199	96
322	106	283	232	159	178	224	107
333	120	267	246	151	157	241	127
334	138	241	258	148	138		



(bolum7_011.jpg, **Problem 15 kalp şekli**)

PROBLEM 16

$I = \int_{0.1x^3}^{0.5x^2} \int e^{y/x} dy dx$ integralini Gauss-Legendre Metodu kullanarak çözünüz.

PROBLEM 17

$$I = \int_{0}^{1} \int_{x}^{2x} (x^2 + y^3) dy dx$$

a) 2 boyutlu Gauss-Legenre integrali kullanarak çözünüz
b) 2 boyutlu simpson 1/3 integrali kullanarak çözünüz
c) 2 boyutlu simpson. 3/8 integralini kuralını kullanan bir program geliştiriniz ve çözünüz

PROBLEM 18

$$I = \int_{x=0}^{1} \int_{y=0}^{1} \frac{dy\,dx}{(1+y)^2}$$

integralini monte-carlo metodu kullanarak çözünüz, 2 boyutlu Gauss-Legenre integrali kullanarak sonucu kontrol ediniz.

PROBLEM 19

$\pi = \int_{0}^{2} (4 - x^2)^{1/2} dx$ denkleminin doğruluğunu gösteriniz.

PROBLEM 20

$I = \int_{0}^{1} e^x dx$ integralini m=25,50,100,500,1000,10000,100000 için monte karlo metodu ile çözünüz ve analitik sonuçla karşılaştırınız.

PROBLEM 21

Kütlesi 5400 kg, hızı 30 m/saniye olan bir otomobilin motoru aniden durdurulmaktadır. Motorun durdurulduğu t=0 anında hareket denklemi

$$5400 * v * (dv/dx) = -8{,}276 * v^2 - 2000$$

olarak verilmektedir. Burada v(m/sn), t anında otomobilin hızıdır. Hız 15 m/sn olana dek aracın gideceği yolu 4 nokta Gauss yöntemiyle hesaplayınız.

PROBLEM 22

Durmakta olan bir aracın, hareket ettirildikten sonraki ilk 5 dakikalık zaman aralığında hızının değişimi tabloda verilmiştir. Bu sürede aracın kaç metre yol alacağını çoklu yamuk yöntemi ile hesaplayınız.

t(dk)	0	0,5	1,0	1,5	2,0	2,5	3,0	3,5	4,0	4,5	5,0
v (km/h)	0	4	7	11	15	21	25	30	32	35	40

Not : $v = \frac{dx}{dt} \qquad x - x_0 = \int_{t0}^{t} v(t)dt$

PROBLEM 23

Bir roketin yukarı doğru hızı aşağıdaki formülle hesaplanabilir.

$$V = \frac{dz}{dt} = U * \ln[m_o / (m_o - q*t)] - g*t$$

Burada V yukarı doğru hız, U yakıtın rokete göre dışarı atıldığı hız, m_o roketin t = 0 sn anındaki başlangıç kütlesi, q = yakıt tüketimi hızı, g = aşağı yönlü yerçekimi ivmesi (sabit varsayılmıştır, 9,8 m^2/s). Eğer U = 2000 m/s, m_o = 150 000 kg, q = 2600 kg/s ise 30 saniyede roketin çıkacağı z yüksekliği Romberg integralini kullanarak k=4 için [$O(h^8)$] hata mertebesi ile hesaplayınız.

PROBLEM 24

Aşağıdaki verileri kullanarak t=5 saniyedeki hızı ve ivmeyi, a) merkezi sonlu fark [$O(h^4)$ hata mertebesi ile], b) ileri doğru sonlu fark [$O(h^2)$ hata mertebesi ile], c) geriye doğru sonlu fark [$O(h^2)$ hata mertebesi ile] yöntemleri ile hesaplayınız.

Zaman, t (s)	0	1	2	3	4	5	6	7	8
Konum, x (m)	0	0.7	1.8	3.4	5.1	6.5	7.3	8.0	8.4

PROBLEM 25

Aşağıdaki verileri yamuk(trapez) yöntemini kullarak integre ediniz.

x	-3	-1	1	3	5	7	9	11
f(x)	1	-4	-9	2	4	2	6	-3

PROBLEM 26

Bir nehirde nehir tabanı ile köprü alt yüzeyi arasındaki mesafe [f(X)], akış kesitine dik doğrultuda her x=12,5 metrede ölçülmüştür. Nehirde su köprü tabanına temas ederek akmaktadır. Su debisi Q = ∫ V*dA integrali alınarak hesaplanmaktadır. Burada Q: su debisi (m^3/s), V: su hızı (m/s), A: akış kesit alanı (m^2)'dır. Ortalama su hızı 0,4 m/s ise Simpson yöntemini kullanarak su debisini hesaplayınız.

X(m)	0	12,5	25	37,5	50	62,5	75	87,5	100
f(X) m	11	7,5	5,5	0	1,5	4	5,5	8	11

PROBLEM 27

$\int_0^{10} f(x)dx$ integralini aşağıda veri olarak verilen değerleri kullanarak tahmin ediniz.

X	0.99	2.10	3.22	4.40	5.70	7.12	8.01	8.37	9.32	9.98
f(x)	4.90	5.70	4.20	7.04	8.31	7.82	5.97	7.01	6.68	4.79

PROBLEM 28

f(x) fonksiyonunun x=3,4,5,6,7,8 noktalarındaki türevlerini Program 26da verilen verilen değerleri kullanarak tahmin ediniz.

PROBLEM 29

Gauss-Legendre integral denklemi kullanarak $I = \int_{-4}^{4} \frac{dx}{1+x^2}$ integralini n=2,3,4,5 için hesaplayınız. Sonucu tam değeriyle karşılaştırınız.

PROBLEM 30

Ayarlanabilir simpson 1/3 integrali kullanarak $I = \int_{-4}^{4} \frac{dx}{1+x^2}$ integralini

1.0e-5 hata miktarına kadar hesaplayınız.

PROBLEM 31

Gauss-Chebychev integral denklemi kullanarak $I = \int_{-1}^{1} \frac{\cos(x)dx}{\sqrt{(1-x^2)}}$ = 2.40394 integralini

çözünüz ve tam değeriyle karşılaştırınız.

PROBLEM 32

Trapezoidal integral denklemi kullanarak $I = \int_{0}^{1} \frac{2dx}{2+\sin(10\pi x)} = 1.15470054$ integralini n=2,4,8 ve 16 değerleri için hesaplayınız ve bu değerlere Richardson extrapolasyon formülünü uygulayınız.

PROBLEM 33

Bir borudaki akışkanın hız dağılımı biliniyorsa birim zamanda borudan geçen suyun hacmi (Q), $Q = \int v\, dA$ eşitliği ile hesaplanır. Burada v akışkanın hızı, A borunun akış kesit alanıdır. Dairesel kesitli bir boruda $A = (2\ \pi\ r)$ ve $dA = (2\ \pi\ r\ dr)$'dir. Buradan
$Q = \int v(2\pi r)\, dr$ eşitliği elde edilir. Burada r boru merkezinden dışa doğru yarıçap doğrultusunda uzaklıktır. Eğer hız dağılımı $v = 3[1-(r/r_0)]^{(1/7)}$ olarak ifade edilirse boru yarıçapı r_0= 10 cm için çoklu yamuk(trapez) yöntemi ile hacımsal akış debisini (Q) hesaplayınız.
Not: En az 10 aralık seçilecektir.

PROBLEM 34

Aşağıdaki tam değerleri verilmiş integralleri Ayarlanabilir integral formülleri kullanarak çözüm hata seviyelerinizi kontrol ediniz

a) $I1 = \int_{0}^{1} \frac{4dx}{1+256(x-0.375)^2} = \frac{1}{4}[\tan^{-1}(10) + \tan^{-1}(6)] = 0.7191938309210011$

b) $I2 = \int_{0}^{1} x^2\sqrt{x}dx = \frac{2}{7}$

c) $I3 = \int_{0}^{1} \sqrt{x}dx = \frac{2}{3}$

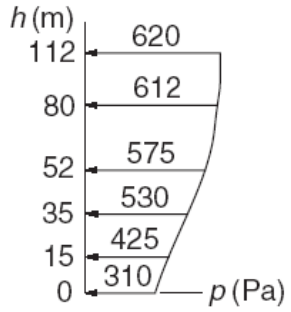
d) $I4 = \int_{0}^{1} \log(x)dx = -1$

e) $I5 = \int_{0}^{1} \log(|x-0.7|)dx = 0.3*\log(0.3) + 0.7*\log(0.7) - 1.0$

f) $I6 = \int_0^{2\pi} e^{-x}\sin(50x)dx = \frac{50}{2501}(1-e^{-2\pi})$

g) $I6 = \int_{-9}^{10000} \frac{dx}{\sqrt{|x|}} = 206$

PROBLEM 35



(bolum7_012.jpg, **Problem 35 rüzgar basıncı**)

Düz bir duvardaki rüzgar basıncı diyagramda görüldüğü gibi ölçülmüştür. Bu duvardaki basınç etki merkezini bulmak istiyoruz. Basınç etki merkezi :

$\bar{h} = \frac{\int_0^{112} hp(h)dh}{\int_0^{112} p(h)dh}$ formülüyle hesaplanır.

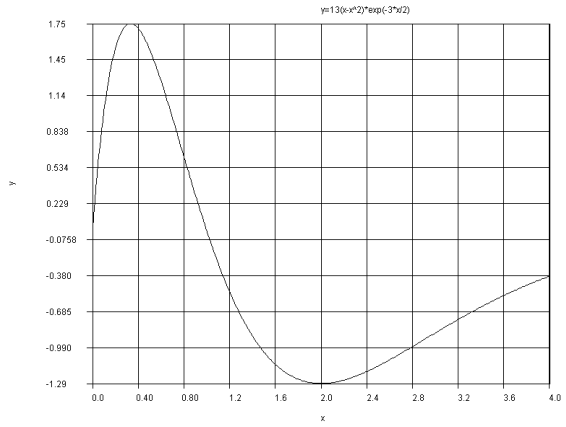
PROBLEM 36

Aşağıda analitik çözümü verilen integrali

a) Ayarlanabilir hata miktarlı Simpson integral formülünü kullanarak çözünüz ve hata kontrolu yapınız

b) Ayarlanabilir Gauss-Kronrod integral formülünü kullanarak çözünüz ve hata kontrolunu yapınız.

$$\int_0^4 13(x-x^2)e^{-3x/2}\,dx = \frac{4108e^{-6}-52}{27}$$
$$= -1.5487883725279481333$$



(bolum7_013.jpg, **Problem 36**)

PROBLEM 37

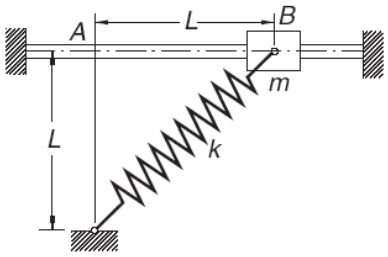
v(m/s)	0	1	1.8	2.5	3.5	4.4	5.1	6

P(kW)	0	4.7	12.2	19.0	31.8	40.1	43.8	43.2

Tabloda hızın fonksiyonu olarak bir arabanın tekerleklerine iletilen güç görülmektedir. Eğer arabanın kütlesi m=2000 kg ise, arabanın 0 m/s^2 den 6 m/s^2 ye hızlanması için gerekli olan süreyi hesaplayınız.

Not : $\Delta t = m\int_{0s}^{6s}(v/P)dv$

PROBLEM 38



(bolum7_014.jpg, **Problem 38**)

kayarak hareket edebilen bir m kütlesi şekilde görüldüğü gibi yay katsayısı k olan bir yayla bağlanmıştır. B noktasında sistem serbest bırakıldığında a noktasına ulaşması için gerekli zaman

$t = \left\{\int_0^1 \left[\left(\sqrt{2}-1\right)^2 - \left(\sqrt{2+z^2}-1\right)^2\right]^{-\frac{1}{2}} dz\right\}\sqrt{\frac{m}{k}}$ denklemi ile hesaplanabilir. Bu problemin çözümü için bir program geliştiriniz. M, k ve L değerleri değişken olarak tanımlanacaktır.

PROBLEM 39

$I = \int_0^1\int_1^2\int_0^y e^{x+y+z}dzdydx$ integralini 3 boyutlu Gauss-Legendre integrali kulanarak çözünüz.

PROBLEM 40

$I = \int_0^{\pi}\int_0^{x}\int_0^{xy}\frac{1}{y}\sin(\frac{z}{y})dzdydx$ integralini 3 boyutlu Gauss-Legendre integrali kulanarak çözünüz.

PROBLEM 41

Siyah cisim radyasyon fonksiyonu F(0- λT), λ(μm) T(derece Kelvin) cinsinden

$$x = \frac{C_2}{\lambda T} = \frac{14387.69\ \mu mK}{\lambda T}$$

$$F(0-\lambda T) = 1 - \frac{15}{\pi^4}\int_0^x \frac{x^3}{e^x - 1}dx$$

İntegral formülü ile hesaplanır. Bu değer aynı zamanda

$$F(0-\lambda T) = \frac{15}{\pi^4}\sum_{n=1}^{\infty}\left[\frac{e^{-nx}}{n}(x^3 + \frac{3x^2}{n} + \frac{6x}{n^2} + \frac{6}{n^3})\right]$$

serisiyle de hesaplanabilir. Bu seri ve integral formülünü hesaplayan metodlar yazınız. Çıktı değerlerini seri ve integral çözümüyle ve tablo değerleriyle karşılaştırınız. İntegral yöntemi size bırakılmıştır.

(Tablo referans amacıyla verilmiştir.)

λ T	**F(0-λT)**	**λ T**	**F(0-λT)**	**λ T**	**F(0-λT)**	**λ T**	**F(0-λT)**	**λ T**	**F(0-λT)**	**λ T**	**F(0-λT)**
μ mK		**μ mK**		**μ mK**		**μ mK**		**μ mK**		**μ mK**	
1000	0.00032	2900	0.25056	4800	0.60754	6900	0.8022	10700	0.92709	18000	0.98081
1050	0.00056	2950	0.26191	4850	0.61428	7000	0.80808	10800	0.92872	18200	0.98137
1100	0.00091	3000	0.27323	4900	0.62089	7100	0.81374	10900	0.93031	18400	0.98191

1150	0.00142	3050	0.28453	4950	0.62737	7200	0.81918	11000	0.93185	18600	0.98243
1200	0.00213	3100	0.29578	5000	0.63373	7300	0.82443	11200	0.9348	18800	0.98293
1250	0.00308	3150	0.30697	5050	0.63996	7400	0.82949	11400	0.93758	19000	0.98341
1300	0.00432	3200	0.3181	5100	0.64608	7500	0.83437	11600	0.94021	19200	0.98387
1350	0.00587	3250	0.32915	5150	0.65207	7600	0.83907	11800	0.9427	19400	0.98431
1400	0.00779	3300	0.34011	5200	0.65795	7700	0.8436	12000	0.94505	19600	0.98474
1450	0.01011	3350	0.35097	5250	0.66371	7800	0.84797	12200	0.94728	19800	0.98516
1500	0.01285	3400	0.36173	5300	0.66937	7900	0.85219	12400	0.94939	20000	0.98555
1550	0.01605	3450	0.37238	5350	0.67491	8000	0.85625	12600	0.95139	20500	0.98649
1600	0.01972	3500	0.38291	5400	0.68034	8100	0.86018	12800	0.95329	21000	0.98735
1650	0.02388	3550	0.39332	5450	0.68566	8200	0.86397	13000	0.95509	21500	0.98814
1700	0.02854	3600	0.4036	5500	0.69089	8300	0.86763	13200	0.95681	22000	0.98886
1750	0.03369	3650	0.41375	5550	0.696	8400	0.87116	13400	0.95843	22500	0.98952
1800	0.03934	3700	0.42377	5600	0.70102	8500	0.87457	13600	0.95998	23000	0.99014
1850	0.04549	3750	0.43364	5650	0.70594	8600	0.87787	13800	0.96145	23500	0.9907
1900	0.05211	3800	0.44338	5700	0.71077	8700	0.88105	14000	0.96285	24000	0.99123
1950	0.0592	3850	0.45297	5750	0.7155	8800	0.88413	14200	0.96419	24500	0.99172
2000	0.06673	3900	0.46241	5800	0.72013	8900	0.88711	14400	0.96546	25000	0.99217
2050	0.07469	3950	0.47172	5850	0.72468	9000	0.88999	14600	0.96667	26000	0.99297
2100	0.08306	4000	0.48087	5900	0.72914	9100	0.89278	14800	0.96783	27000	0.99368
2150	0.0918	4050	0.48987	5950	0.73351	9200	0.89547	15000	0.96893	28000	0.99429
2200	0.10089	4100	0.49873	6000	0.73779	9300	0.89808	15200	0.96999	29000	0.99482
2250	0.11031	4150	0.50744	6050	0.74199	9400	0.9006	15400	0.971	30000	0.99529
2300	0.12003	4200	0.516	6100	0.74611	9500	0.90305	15600	0.97196	32000	0.99607
2350	0.13002	4250	0.52442	6150	0.75015	9600	0.90541	15800	0.97289	34000	0.99669
2400	0.14026	4300	0.53269	6200	0.75411	9700	0.9077	16000	0.97377	36000	0.99719
2450	0.15071	4350	0.54081	6250	0.758	9800	0.90992	16200	0.97461	38000	0.99759
2500	0.16136	4400	0.54878	6300	0.76181	9900	0.91207	16400	0.97542	40000	0.99792
2550	0.17217	4450	0.55662	6350	0.76554	10000	0.91416	16600	0.9762	45000	0.99851
2600	0.18312	4500	0.56431	6400	0.76921	10100	0.91618	16800	0.97694	50000	0.9989
2650	0.19419	4550	0.57186	6450	0.7728	10200	0.91814	17000	0.97765	55000	0.99917
2700	0.20536	4600	0.57927	6500	0.77632	10300	0.92004	17200	0.97834	60000	0.99935
2750	0.2166	4650	0.58654	6600	0.78317	10400	0.92188	17400	0.97899		
2800	0.22789	4700	0.59367	6700	0.78976	10500	0.92367	17600	0.97962		
2850	0.23922	4750	0.60067	6800	0.7961	10600	0.9254	17800	0.98023		

PROBLEM 42

Siyah cismin tüm dalga boyları için integrali

$$E_b = \left[\frac{C_1}{C_2^4} \int_{x=0}^{\infty} \frac{x^3}{e^x - 1} dx \right] T^4 = \sigma T^4$$ ifadesiyle hesaplanabilir. Buradaki C_1=3.74177489x10^8 W□m^4/m^2 ve C_2=1.438769x10^4 μm K dir. Denklemdeki σ Stefan_Boltzman sabiti adını alır. Bu integrali Gauss-Leguerre integral formülü integral formülünü kullanarak hesaplayınız. Stefan – Boltzman sabitinin değeri kitapta σ =5.670x10^{-8} W/m^{2}K^4 olarak verilmiştir.

PROBLEM 41 Besel fonksiyonu $J_n(x)$:

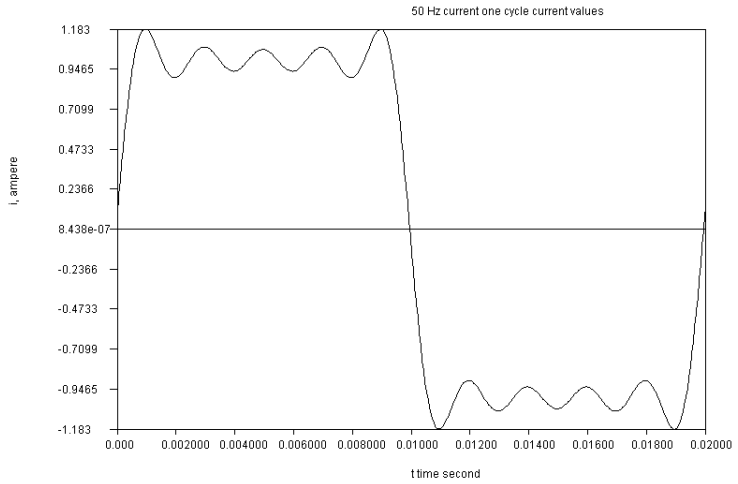
$$J_n(x) = \frac{1}{\pi} \int_0^{\pi} \cos(x \sin\theta - n\theta) d\theta$$

İfadesiyle tanımlanabilir. Bu denklemi verilen bağıntı üzerinden tanımlayan bir foksiyon yazınız.

PROBLEM 42 Bir doğru akım-alternatif akım (AC-DC) dönüştürücünün sinyali aşağıdaki grafikte ve java fonksiyonu olarak verilmektedir. Sinyal frekansı 50 hertz(periodu T=1.0/50=0.02 saniye) converter the signal defined with the following function is obtained. Signal frequency is 50 hertz (period T=1.0/50=0.02 second). Sinüs dalgası eşdeğeri (true RMS) Akım şiddeti aşağıdaki denklemle hesaplanabilir. $I_{RMS} = \sqrt{\frac{1}{T}\int_0^T i^2(t)dt}$

Akım şiddetini hesaplayınız.

```
class ff extends f_x
{
double a[]={-8.673617379884035E-17,
-0.02000000000000008,
-3.8163916471489756E-17,
-0.01999999999999792,
-7.45931094670027E-17,
-0.01999999999999994,
-3.122502256758253E-17,
-0.01999999999999858,
1.9255430583342559E-16,
-0.01999999999999983,
9.020562075079397E-17};

double b[]={
0.0,
-1.2731348232574313,
-7.773729576721067E-17,
-0.42409897579377515,
1.1232334506949826E-16,
-0.2541240947234941,
3.447762908503904E-17,
-0.18115773372477864,
4.336808689942018E-18,
-0.14052732458082803,
-2.5110122314764283E-16};
int n=10;
double T=1.0/50.0;
double x0=0;
double xn=T;

public double z_x(double x)
{double zz=2.0*Math.PI*(x-x0)/(xn-x0)-Math.PI;
 return zz;
}

public  double func(double x)
{
double cc=a[0]/2.0+a[n]*Math.cos(n*z_x(x));
for(int k=1;k<n;k++)
{cc+=a[k]*Math.cos(k*z_x(x))+b[k]*Math.sin(k*z_x(x));}
return cc;
}}
```



(bolum7_015.jpg, **Problem 38**)

PROBLEM 43 İş tanımı **İş=Kuvvet*Mesafe** şeklinde yapılabilir. Bu yüzden değişken bir kuvvet uygulanan sistemler için iş denklemi

$W = \int_{x0}^{xn} F(x)dx$ formunu alır. Bir sistemde Kuvvet aşağıdaki verilen tablodakigibi ölçülmüştür.

x(metre)	**0**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**
F(Newton)	**0**	**9**	**12**	**15**	**18**	**23**	**16**	**13**	**11**

Sistem iş çıkışını hesaplayınız.

PROBLEM 31

Bir makinanın titreşimi (x) farklı zamanlarda (t) ölçülerek tablodaki değerler belirlenmiştir.

İ	**1**	**2**	**3**	**4**
t_i	0	2	4	6
x_i	5,0	3,7	2,7	2,0

Bu makinanın titreşimi için:

$$x(t) = ae^{-bt}$$

biçiminde zamana bağlı bir fonksiyon uydurunuz.

PROBLEM 32

Yeni bir cihazın geliştirme çalışmasında, iki paralel levha arasındaki kayma geriliminin (τ) ölçülmesi gerekmektedir. Ölçme deneyleri sırasında üstteki levha 2 m/s hızla hareket ettirilirken, alttaki levha ise sabit tutulmaktadır. İki levha arasındaki uzaklık (h) 0,025 m'dir. Kayma gerilimi (τ)

$$\tau = \mu \times (V / h)$$

eşitliği ile hesaplanmaktadır. Viskozite (μ) birimi N-s/m^2 olup, sıcaklığın fonsiyonudur. Laboratuvar çalışması sonucu viskozite ile sıcaklık arasındaki ilişki tablodaki gibi elde edilmiştir.

T (°C)	5	20	30	50	55
μ (N-s/m^2)	0,08	0,015	0,009	0,006	0,005

a) Lagrange interpolasyon polinomlarını kullanarak 38 °C'daki viskoziteyi ve kayma gerilimini hesaplayınız.
b) Newton interpolasyon polinomlarını kullanarak birinci, ikinci, üçüncü, dördüncü denklemleri ayrı ayrı elde ediniz. En uygun denklemi belirleyiniz. Bu denklemi kullanarak 38 °C'daki viskoziteyi ve kayma gerilimini hesaplayınız.

PROBLEM 33

Bir makina yağının değişik sıcaklıklarda ölçülen viskozitesi aşağıdaki tabloda verilmiştir. Vizkozitenin sıcaklıkla değişimi için:

$\mu = a \cdot e^{b/T}$ biçiminde bir eğri uydurunuz.

T, (K)	253	293	333	353	373
μ, (Pa·s)	4,0	0,1	0,018	0,0086	0,005

PROBLEM 34

Bir pompa imalatçısı , imal ettiği pompaları deneyerek çeşitli debilerde (Q) pompaların sağladığı basınç (P) değerlerini ölçmüş ve aşağıdaki değerleri elde etmiştir.

Q (lt/s)	0,5	1,0	1,5	2,0	2,5	3,0	3,5
P (kPa)	275	290	280	255	229	205	178

a-)Bu değerlere ikinci dereceden bir polinomu[$P=f(Q)=aQ^2+bQ+c$] en küçük kareler yöntemi ile uydurunuz.
b-) Pompa 2,2 lt/s debide su basarken oluşacak basıncı tahmin ediniz.
c-) Bu pompa ile en yüksek basınç hangi debide su basılırken oluşur?

PROBLEM 35
Yanma gazları yoğunluğunun (ρ: kg/m^3), sıcaklıkla (T:°C) değişimi aşağıda verilmiştir.

ρ: kg/m^3	0,950	0,748	0,617	0,525
T:°C	100	200	300	400

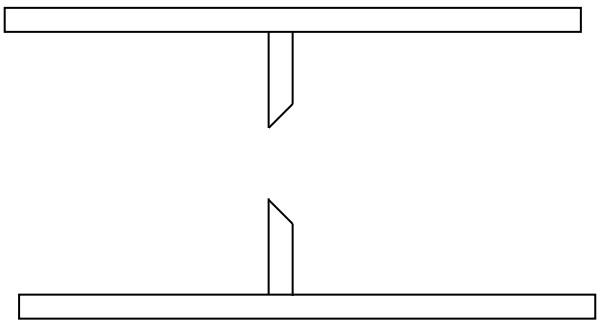
Newton enterpolasyon yöntemini kullanarak 1'den 4'ü dereceye kadar polinom Denklemlerini bulunuz. 285 °C'da yanma gazlarının yoğunluğunu hesaplayınız.

PROBLEM 35
Keskin uçlu dairesel bir orifis borular içindeki akışkan debisini ölçmek için kullanılmaktadır. Böyle bir cihaz şekilde gösterilmektedir. Aşağıdaki veriler bu orifis tasarımında kullanılmıştır.

A : Orifis kesit alanı.
A_1 : Boru kesit alanı.
A_2 : C_cA (Orifis arkasında büzülme alanı.

Burada C_c büzülme katsayısı (A/A_1) oranının bir fonksiyonudur. Bu katsayı için deneysel veriler aşağıda verilmiştir. Bu deneysel verilere uygun bulabileceğiniz en küçük kareler eğrisini [$C_c=f(A/A_1)$] uydurunuz.

A/A_1	C_c
0.1	0.62
0.2	0.63
0.3	0.64
0.4	0.66
0.5	0.68



0.6	0.71
0.7	0.76
0.8	0.81
0.9	0.89
1.00	1.00

(bolum6_047.jpg,Resimaltı: Problem 35 orifis**)**

PROBLEM 36

Bir çeliğin talaş kaldırmalı işlenmesinde kullanılan makinadaki sert kesme ucunun ömrü, T (dak) ile, kesme hızı, V (m/dak) arasında gerçekleştirilen ömür testlerinde, sabit derinlikte ve ilerleme hızında aşağıdaki tablodaki deneysel sonuçlar belirlenmiştir.

Kesme hızı, V (m/dak)	Sert kesme ucu ömrü,T (dakika)
140	8,0
160	7,0
180	5,5
200	5,0
220	3,5

Sert kesme ucu ömrü için:

$$\mathbf{V=(b/T)^{(1/a)}}$$

Biçiminde bir eşitliği, en küçük kareler yöntemi ile belirleyiniz.

PROBLEM 37

Bir boru içinden akan sıcak akışkandan, ısı kaybını belirlemek için, h_i : boru içi ısı taşınım katsayısının bilinmesine gereksinim duyulur. h_i'nin belirlenmesi için gerçekleştirilen deneylerde,

$$Nu = C\,\mathrm{Re}^{m}\,\mathrm{Pr}^{0,3}$$

biçiminde, Nu : Nusselt sayısı,
Re : Reynolds sayısı,
Pr : Prandtl sayısı,

Boyutsuz sayıları arasındaki ilişkiler araştırılır.

Burada *C* ve *m* belirli aralıklardaki Re ve Pr sayıları için sabitlerdir.

Aşağıdaki deney sonuçları için C ve m sabitlerini en küçük kareler yöntemi ile (çoklu regresyon) belirleyiniz.

Nu	**Re**	**Pr**
34.06	10500	0.70
43.05	14000	0.71
49.65	15000	0.95
51.03	17500	0.69

Motor yağının yoğunluğunun [ρ (kg/m^3)], sıcaklıkla [T(K)] değişimi tabloda verilmiştir. Bu verileri kullanarak yoğunluğun sıcaklıkla değişimini veren denklemleri

a-) Newton enterpolasyon yöntemi ve

b-) Lagrange yöntemi ile bulunuz.

ρ (kg/m^3)	859.9	847.8	836.0	825.1	812.1
T(K)	340	360	380	400	420

PROBLEM 38

Aşağıdaki verilere en küçük kareler yaklaşımını uygulayarak,

$$y=ax^b$$

eğrisini uydurunuz.

X	1	2	3	4	5
Y	1	1	2	4	4

PROBLEM 39

Suyun doymuş sıvı entalpisinin sıcaklıkla değişimini belirlemek için, t oC biriminde sıcaklık olmak üzere

$$h_f = A + Bt + Ct^2 + Dt^3$$

biçiminde bir kübik eşitlik uydurulmak istenmektedir. Aşağıdaki veriler termodinamik tablolardan alınmıştır. Bu verilerle belirtilen eşitlikteki sabitleri, en küçük kareler eğri uydurma yöntemi ile

t, ^oC	10	30	50	70	90
h_f , kJ/kg	41.99	125.66	209.26	292.97	376.94

PROBLEM 40

Tablo verilerini kullanarak aşağıdaki soruları yanıtlayınız.

X	1	2	3	5	6
f(X)	4,75	4	5,25	19,75	36

a) Newton enterpolasyon yöntemini kullanarak 4'cü derece polinom denklemini elde ediniz. X=4 için bu denklemden fonksiyon değerini elde ediniz.

b) Üçüncü derece spline fonksiyonunu ikinci aralık için elde ediniz. Bu denklemi kullanarak X=4 için fonksiyon değerini elde ediniz.

PROBLEM 41

Aşağıdaki veriler verilmiştir. En küçük kareler yöntemi kullanarak düz bir doğru ($f(x)=a_0+a_1x$) geçiriniz.

x	5	10	15	20	25	30	35	40	45	50
y	16	25	32	33	38	36	3	40	42	42

PROBLEM 42

SAE30 yağının kinematik vizkozite verileri aşağıda verilmiştir.

Sıcaklık T(C)	5	10	15	20	25	30	35	40	45	50
Vizkozite □ (m^2/s)	16	25	32	33	38	36	3	40	42	42

a) $\square = ae^{(b/T)}$ denkleminine göre vizkozite için bir denklem uydurunuz.

b) $\mu = \dfrac{a\sqrt{T}}{1+\dfrac{b}{T}}$ denkleminine göre vizkozite için bir denklem uydurunuz

PROBLEM 43

$f(x) = \frac{20}{1+2x^2}$ fonksiyonunu -3<=x<=3 aralığında x_i=-3,-2,-1,0,1,2,3 moktalarını kullanarak

a) kübik şerit interpolasyonu
b) quadratik B-şerit interpolasyonu kullanarak hesaplayınız
c) her iki yaklaşım için asıl foksiyondan hata miktarlarını belirleyiniz.

PROBLEM 44

Demirin ısıl iletim katsayısı(k) sıcaklığın fonksiyonu olarak aşağıda verilmiştir.

Sıcaklık T(K)	200	600	1000	1400
k (W/m^2K)	1	0.4	0.3	0.25

$K=(b/T)^{(1/a)}$ denkleminine göre ısıl iletim katsayısı için bir denklem uydurunuz.

PROBLEM 45

Düzgün yüzeyli bir kürede Reynold sayısı(Re) ile sürtünme katsayısı(C) arasında alttaki bağıntı bulunmuştur..

Re	0.1	1	10	100	1000	10000
C	210	30	4	1	0.5	0.4

Bu değişkenler arasında veriyi yeterince doğru olarak verecek bir denklem oluşturunuz.

PROBLEM 46

Örnek bir problem olarak yukarıda elma kurutma problemi verilmişti.

PROBLEM 47

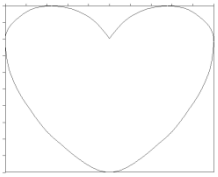
Bezier eğrisi kullanarak α şeklini oluşturunuz.

PROBLEM 48

Bezier eğrisi kullanarak γ şeklini oluşturunuz.

PROBLEM 49

Bir kalp şeklini iki tane simetrik Bezier eğrisi kulanarak oluşturunuz.



(bolum6_048.jpg,Resimaltı: Problem 49 kalp şekli**)**

9 Diferansiyel denklem sistemleri

9.1 Euler denklemi

$\frac{dy}{dx} = f(x,y)$ denkleminin çözümünü veren y denklemini bulmak istiyoruz. Bunun için ilk kullanacağımız ilk metod Euler yöntemidir. Bu yöntem

$y^0_{i+1}=y_i + f(x_i,y_i)h$ olarak yazılabilir. Burada h adım büyüklüğüdür. $h = \Delta x = x_{i+1} - x_i$
Örnek problemimizde önce h=1, h=0.5 e h=0.1 için çözüldü. Çıktılardan da görüldüğü gibi h değeri küçüldükçe çözüm gerçek çözme yaklaşmaktadır.

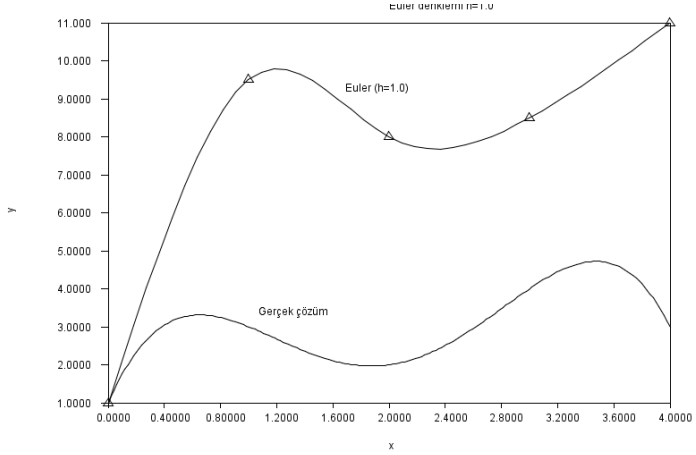
Program 8.1.1 Euler denklemi

```
//=================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =================================================
// Basit diferansiyel denklemler dy/dx=f(x,y)

import java.io.*;

abstract class f_xy
{
  //single function single independent variable
  // example f=x*x
  abstract double func(double x,double y);
}

class fxy extends f_xy
{ //diferansiyeli alınan fonksiyon
  double func(double x,double y)
  {return -2*x*x*x+12*x*x-20*x+8.5;}
}

class fxyKC extends f_x
{ //Gerçek çözüm fonksiyonu
  double func(double x)
  {return -0.5*x*x*x*x+4.0*x*x*x-10.0*x*x+8.5*x+1.0;}
}

class diferansiyel
{
  public static double[][] euler(f_xy f,double x0,double y0,double xn,double h)
  {
  int n=(int)((xn-x0)/h);
  System.out.println("n="+n);
  double a[][]=new double[2][n+1];
  int i=0;
  a[0][0]=x0;
  a[1][0]=y0;
  for(double xx=x0;xx<xn;xx+=h)
   {a[0][i+1]=xx+h;
   a[1][i+1]=a[1][i]+f.func(a[0][i],a[1][i])*h;
   i++;
   }
  return a;
  }

  public static void main(String args[]) throws IOException
  {
   double x0=0;
   double y0=1.0;
   fxy f1=new fxy();
   fxyKC fKC=new fxyKC();
   double a[][]=euler(f1,x0,y0,4.0,0.1);
   Text.print(Text.T(a));
   Plot p1=new Plot(a);
   p1.setPlotType(0,24);
   p1.addSpline(a,5);
   p1.addFunction(fKC,0.0,4.0,200);
```

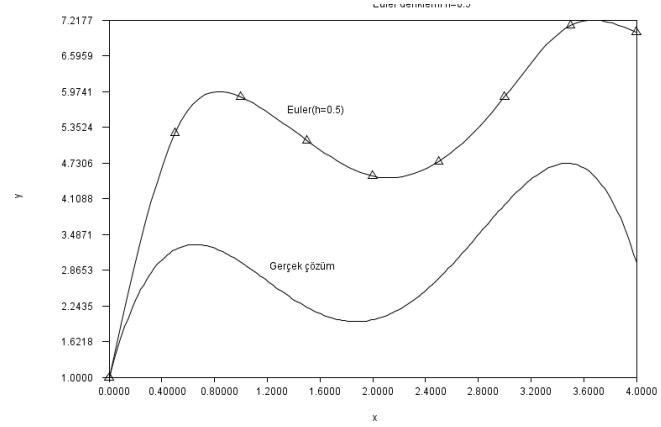
```
  p1.plot();
 }
}
```

Euler denklemi h=0.1

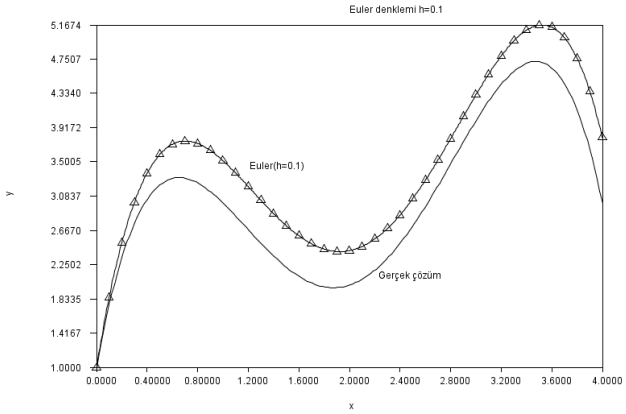
0.0	1.0
0.1	1.85
0.2	2.5118
0.30000000000000004	3.0082
0.4	3.3608000000000002
0.5	3.5900000000000003
0.6	3.7150000000000003
0.7	3.7538000000000005
0.7999999999999999	3.7232000000000003
0.8999999999999999	3.6388000000000003
0.9999999999999999	3.515
1.0999999999999999	3.365
1.2	3.2008000000000005
1.3	3.0332000000000003
1.4000000000000001	2.8718000000000004
1.5000000000000002	2.7250000000000005
1.6000000000000003	2.600000000000001
1.7000000000000004	2.502800000000001
1.8000000000000005	2.4382000000000015
1.9000000000000006	2.4098000000000015
2.0000000000000004	2.4200000000000017
2.1000000000000005	2.4700000000000024
2.2000000000000006	2.5598000000000027
2.3000000000000007	2.688200000000002
2.400000000000001	2.852800000000002
2.500000000000001	3.050000000000003
2.600000000000001	3.2750000000000044
2.700000000000001	3.5218000000000047
2.800000000000001	3.7832000000000057
2.9000000000000012	4.050800000000007
3.0000000000000013	4.315000000000005
3.1000000000000014	4.565000000000003
3.2000000000000015	4.788800000000003
3.3000000000000016	4.973200000000004
3.4000000000000017	5.103800000000005
3.5000000000000018	5.1650000000000045
3.600000000000002	5.140000000000003
3.700000000000002	5.010800000000003
3.800000000000002	4.7582



(bolum8_000.jpg,Resimaltı: **Euler denklemi h=1**)



(bolum8_001.jpg,Resimaltı: **Euler denklemi h=0.5**)



(bolum8_002.jpg,Resimaltı: **Euler denklemi h=0.1**)

9.2 Heun denklemi

Heun yöntemi euler yöntemin düzeltme kullanılan bir şeklidir.

$y^0_{i+1}=y_i+f(x_i,y_i)h$

$y_{i+1}^{k+1} = y_i + \frac{f(x_i, y_i^k) + f(x_{i+1}, y_{i+1}^k)}{2}$ $,k = 0...n$ şeklinde hesaplanır.

Burada bulduğumuz y^k_{i+1}iteratif olarak hesaplanabilecek bir terimdir. İterasyon stepinin uygulanma sayısı değişebilir.

Program 8.2-1 Heun denklemi

```
//==================================================
// Numerical Analysis package in java
```

```
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// ==================================================
// Basit diferansiyel denklemler dy/dx=f(x,y)

import java.io.*;

abstract class f_xy
{
 //single function single independent variable
 // example f=x*x
 abstract double func(double x,double y);
}


class fxy extends f_xy
{
 double func(double x,double y)
 {return -2*x*x*x+12*x*x-20*x+8.5;}
}

class fxyKC extends f_x
{
 double func(double x)
 {return -0.5*x*x*x*x+4.0*x*x*x-10.0*x*x+8.5*x+1.0;}
}

class fxy1 extends f_xy
{
 double func(double x,double y)
 {return 4.0*Math.exp(0.8*x)-0.5*y;}
}

class diferansiyelA
{

 public static double[][] heun(f_xy f,double x0,double y0,double xn,double h)
 {
 int n=(int)((xn-x0)/h);
 System.out.println("n="+n);
 double a[][]=new double[2][n+1];
 int i=0;
 a[0][0]=x0;
 a[1][0]=y0;
 double yi0=0;
 for(double xx=x0;xx<xn;xx+=h)
  {a[0][i+1]=xx+h;
   yi0=a[1][i]+f.func(a[0][i],a[1][i])*h;
   for(int j=0;j<5;j++)
   {yi0=a[1][i]+(f.func(a[0][i],a[1][i])+f.func(a[0][i+1],yi0) )/2.0; }
    a[1][i+1]=a[1][i]+(f.func(a[0][i],a[1][i])+f.func(a[0][i+1],yi0) )/2.0;
  i++;
  }
 return a;
 }

 public static void main(String args[]) throws IOException
 {
  double x0=0;
  double y0=1.0;
  fxy f1=new fxy();
```

```
  fxyKC fKC=new fxyKC();
  double a[][]=heun(f1,x0,y0,4.0,0.1);
  Text.print(Text.T(a));
  Plot p1=new Plot(a);
  p1.setPlotType(0,24);
  p1.addSpline(a,8);
  p1.addFunction(fKC,0.0,4.0,200);
  p1.plot();
 }
}
```

Çıktı 9.1-1 Heun denklemi, step h=1 için

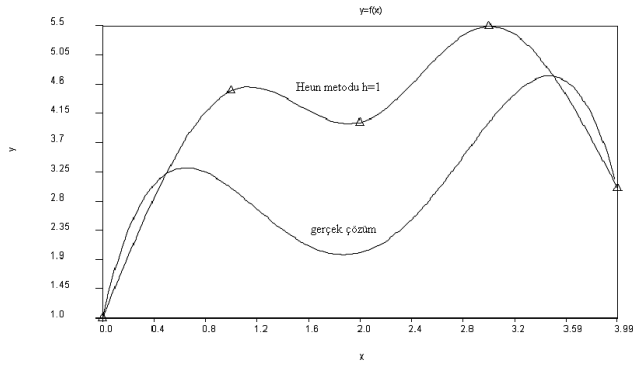
0.0 1.0

1.0 4.5

2.0 4.0

3.0 5.5

4.0 3.0



(bolum8_003.jpg,Resimaltı:**Heun denklemi h=1**)

9.3 Orta nokta veya geliştirilmiş poligon denklemi

Bu yöntemde önce orta noktadaki y değeri bulunur, sonra orta noktadaki eğim kullanılarak gerçek noktadaki çözüm bulunur.

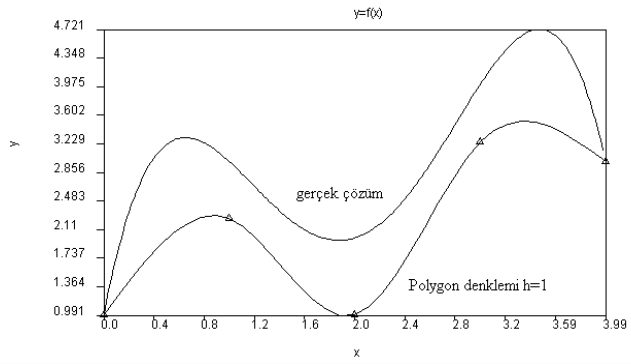
$y_{i+1/2} = y_i + f(x_i , y_i)h/2$

$y_{i+1} = y_i + f(x_{i+1/2} , y_{i+1/2})h$

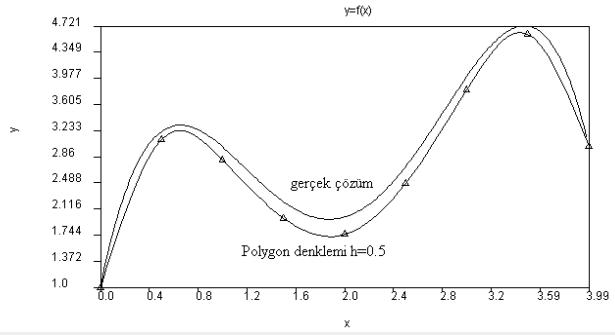
Program 8.3.1 Orta nokta ve geliştirilmiş polinom denklemi

```
//=====================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
// Basit diferansiyel denklemler dy/dx=f(x,y)

import java.io.*;

abstract class f_xy
{
```

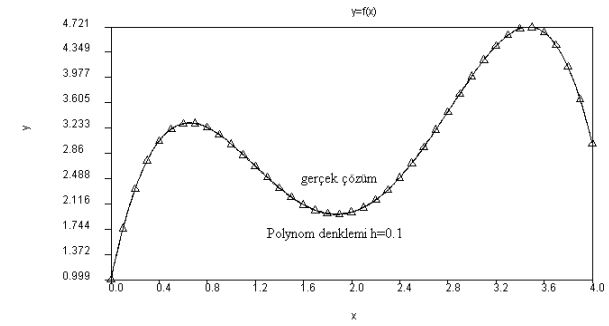
```
  //single function single independent variable
  // example f=x*x
  abstract double func(double x,double y);
}

class fxy extends f_xy
{
  double func(double x,double y)
  {return -2*x*x*x+12*x*x-20*x+8.5;}
}

class fxy1 extends f_xy
{
  double func(double x,double y)
  {return 4.0*Math.exp(0.8*x)-0.5*y;}
}

class diferansiyelB
{
  public static double[][] polygon(f_xy f,double x0,double y0,double xn,double h)
  {
  int n=(int)((xn-x0)/h);
  System.out.println("n="+n);
  double a[][]=new double[2][n+1];
  int i=0;
  a[0][0]=x0;
  a[1][0]=y0;
  double yi0=0;
  double xi0;
  for(double xx=x0;xx<xn;xx+=h)
   {a[0][i+1]=xx+h;
    xi0=xx+h/2.0;
   yi0=a[1][i]+f.func(a[0][i],a[1][i])*h/2.0;
   a[1][i+1]=a[1][i]+f.func(xi0,yi0)*h;
   i++;
   }
  return a;
  }

  public static void main(String args[]) throws IOException
  {
   double x0=0;
   double y0=1.0;
   fxy f1=new fxy();
   fxyKC fKC=new fxyKC();
   double a[][]=polygon(f1,x0,y0,4.0,1.0);
   Text.print(Text.T(a));
   Plot p1=new Plot(a);
   p1.setPlotType(0,24);
   p1.addSpline(a,8);
   p1.addFunction(fKC,0.0,4.0,200);
   p1.plot();
  }
}
```



(bolum8_004.jpg,Resimaltı: **Orta nokta ve geliştirilmiş polinom denklemi, h=1**)



(bolum8_005.jpg,Resimaltı: **Orta nokta ve geliştirilmiş polinom denklemi, h=0.5**)



(bolum8_006.jpg,Resimaltı: **Orta nokta ve geliştirilmiş polinom denklemi, h=0.5**)

9.4 Runge-Kutta denklemleri

Diferansiyel denklem çözüm yöntemi olarak Runge-Kutta denklemini genel şekilde

$y_{i+1} = y_i + \phi(x_i,y_i,h)h$

$$\phi(x_1, y_1, h) = \sum_{i=1}^{n} a_i k_i$$

burada a lar sabit değerler olup k değerleri

$k_1=f(x_i,y_i)$

$k_2=f(x_i+p_1h,y_i+q_{11}k_1h)$

$k_3=f(x_i+p_2h,y_i+q_{21}k_1h+ q_{22}k_2h)$

….

$k_n=f(x_i+p_{n-1}h,y_i+q_{n-1,1}k_1h+ …+ q_{n-1,n-1}k_{n-1}h)$

şeklinde ifade edilebilir. Denklemimizi daha genel bir formda yazacak olursak:

$$x_i = x_n + c_i h \quad \text{i=1…s}$$

$$y_i = y_n + h\sum_{j=1}^{i-1} a_{ij} k_j$$

$$k_i = f(x_i, y_i)$$

$$y_{n+1} = y_n + h\sum_{j=1}^{s} b_i k_i$$

Burada s katsayıların derecesidir. Sayısal analiz kitaplarında bu Runge-Kutta katsayıları genellikle Butcher tablosu denilen bir formda verilir.

$$\begin{array}{c|cccccc} c_1 & a_{11} & a_{12} & a_{13} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & a_{23} & \dots & a_{2s} \\ c_3 & a_{31} & a_{32} & a_{33} & \dots & a_{3s} \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ c_s & a_{s1} & a_{s2} & a_{s3} & \dots & a_{s4} \\ \hline & b_1 & b_2 & b_3 & \dots & b_s \end{array}$$

Görüldüğü gibi yöntem çok geneldir ve katsayıların aldığı değerlere göre hassasiyetleri artmaktadır. Örneğin ikinci derece Runge – Kutta (RK2)

$y_{i+1} = y_i + (0.5k_1 + 0.5k_2)h$

$k_1=f(x_i,y_i)$

$k_2=f(x_i+h,y_i+k_1h)$

Aynı denklemi Butcher tablosu olarak yazarsak:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 0.5 & 0.5 \end{array}$$

İkinci dereceden Ralston yöntemi:

$y_{i+1} = y_i + (1.0/3.0k_1 + 2.0/3.0k_2)h$

$k_1=f(x_i,y_i)$

$k_2=f(x_i+(3.0/4.0)h,y_i+(3.0/4.0)k_1h)$

Butcher tablosu eşdeğeri:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 3/4 & 3/4 & 0 \\ \hline & 1/3 & 2/3 \end{array}$$

Üçüncü dereceden RK yöntemi (RK3):

$y_{i+1} = y_i + (1/6)*(k_1 + 4k_2 +k_3)h$

$k_1=f(x_i,y_i)$

$k_2=f(x_i+0.5h,y_i+0.5k_1h)$

$k_3=f(x_i+h,y_i - k_1h +2k_2h)$

Butcher tablosu eşdeğeri:

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & 1/6 & 4/6 & 1/6 \end{array}$$

Dördüncü dereceden RK yöntemi (RK4):

$y_{i+1} = y_i + (1/6)*(k_1 + 2k_2 +2k_3+k_4)h$

$k_1=f(x_i,y_i)$

$k_2=f(x_i+0.5h,y_i+0.5k_1h)$

$k_3=f(x_i+0.5h,y_i+0.5k_2h)$

$k_4=f(x_i+h,y_i+k_3h)$

Butcher tablosu eşdeğeri:

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array}$$

4. dereceden Runge-Kutta denklemi en çok kullanılan diferansiyel denklem formüllerinin başını çeker.

Altıncı dereceden RK yöntemi:

$y_{i+1} = y_i + (1/90)*(7k_1 + 32k_3 +12k_4+32k_5+7k_6)h$

$k_1=f(x_i,y_i)$

$k_2=f(x_i+0.25h , y_i+0.25k_1h)$

$k_3=f(x_i+0.25h , y_i+0.125k_1h+0.125k_2h)$

$k_4=f(x_i+0.5h , y_i – 0.5k_2h+k_3h)$

$k_5=f(x_i+0.75h , y_i + (3/16)k_1h+(9/16)k_4h)$

$k_6=f(x_i+h , y_i - (3/7)k_1h+(2/7)k_2h+(12/7)k_3h - (12/7)k_4h+(8/7)k_5h)$

Butcher tablosu eşdeğeri:

$$\begin{array}{c|cccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 1/4 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 1/8 & 1/8 & 0 & 0 & 0 & 0 \\ 2/4 & 0 & -1/2 & 1 & 0 & 0 & 0 \\ 3/4 & 3/16 & 0 & 0 & 9/16 & 0 & 0 \\ 1 & -3/7 & 2/7 & 12/7 & -12/7 & 8/7 & 0 \\ \hline & 7/90 & 0 & 32/90 & 12/90 & 32/90 & 7/90 \end{array}$$

9 dereceden Runge-Kutta Verner denklemi

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" dif2
Buthcher tablosu n = 10
   0.00000000 |
   0.00000000 |   0.00000000
   0.06000000 |   0.00000000   0.06000000
   0.09593333 |   0.00000000   0.01923996   0.07669337
   0.14390000 |   0.00000000   0.03597500   0.00000000   0.10792500
   0.49730000 |   0.00000000   1.31868342   0.00000000  -5.04205806   4.22067465
   0.97250000 |   0.00000000 -41.87259166   0.00000000 159.43256216 -122.11921357   5.53174307
   0.99950000 |   0.00000000 -54.43015694   0.00000000 207.06725137 -158.61081378   6.99181659  -0.01859723
   1.00000000 |   0.00000000 -54.66374179   0.00000000 207.95280626 -159.28895747   7.01874374  -0.01833879  -0.00051195
   1.00000000 |   0.00000000   0.03438958   0.00000000   0.00000000   0.25826246   0.42093712   4.40539647 -176.48311902  172.36413340
-----------------------------------------------------------------------------------------------------------------------------------------
                0.00000000   0.03438958   0.00000000   0.00000000   0.25826246   0.42093712   4.40539647 -176.48311902  172.36413340   0.00000000
                0.00000000   0.04909968   0.00000000   0.00000000   0.22511122   0.46946823   0.80657922   0.00000000   -0.60711949   0.05686114

> Terminated with exit code 0.
```

Görüldüğü gibi çok çeşitli şekillerde katsayılar verilebileceği için Runge-Kutta denklemlerini genel formda ve Buthcher tablolarının fonksiyonu olarak vermeyi yeğledik. Bunun için ilk önce Buthcher tablolarını girebilecek bir sınıf oluşturduk, sonra bu sınıftan katsayıları okuyan Runge-Kutta Metodunu oluşturduk.

Program 8.4.1 Buthcher tablosu sınıfı

```
public class Buthcher
{public double a[][];
 public double b[][];
 public double c[];
 public Buthcher(double ai[][],double bi[][],double ci[])
 {set(ai,bi,ci); }

 public void set(double ai[][],double bi[][],double ci[])
 {a=ai;b=bi;c=ci;}

 public Buthcher(String isim)
 {if(isim.equals("RK2"))
  {double a2[][]={{0,0},{1,0}};
   double b2[][]={{0.5,0.5}};
   double c2[]={0,1};
   set(a2,b2,c2);
   }
  else if(isim.equals("Raltston"))
  {double a2[][]={{0,0},{3.0/4.0,0}};
   double b2[][]={{1.0/3.0,2.0/3.0}};
   double c2[]={1.0/6.0,4.0/6.0,1.0/6.0};
   set(a2,b2,c2);
  }
  else if(isim.equals("RK3"))
  {double a2[][]={{0,0,0},{0.5,0,0},{-1,2,0}};
   double b2[][]={{0,0.5,1}};
   double c2[]={1.0/6.0,4.0/6.0,1.0/6.0};
   set(a2,b2,c2);
  }
  else if(isim.equals("RK4"))
  {double a2[][]={{0,0,0,0},{0.5,0,0,0},{0,0.5,0,0},{0,0,1,0}};
   double b2[][]={{1.0/6.0,2.0/6.0,2.0/6.0,1.0/6.0}};
   double c2[]={0,0.5,0.5,1};
   set(a2,b2,c2);
  }
  else if(isim.equals("RK6"))
  {double a2[][]={{0,0,0,0,0,0},{0.25,0,0,0,0,0},{0.125,0.125,0,0,0,0},{0,-0.5,1,0,0,0},{3.0/16.0,0,0,9.0/16.0,0,0},{-3.0/7.0,2.0/7.0,12.0/7.0,-
12.0/7.0,8.0/7.0,0}};
   double b2[][]={{7.0/90.0,0,32.0/90.0,12.0/90.0,32.0/90.0,7.0/90.0}};
   double c2[]={0,0.25,0.25,0.5,0.75,1.0};
   set(a2,b2,c2);
  }
  else if(isim.equals("RKV9"))
  { //Runge-Kutta-Verner 9uncu derece sabit adımlı metodu
    int s=10;
    double c[]=new double[s];
    double a[][]=new double[s][s];
    double bh[]=new double[s];
    double bh1[]=new double[s];
c[0]=0;
c[1] =  0.;
c[2] =  .6e-1;
c[3] =  .95933333333333333333333333333333333333333333e-1;
c[4] =  .1439;
c[5] =  .4973;
c[6] =  .9725;
c[7] =  .9995;
c[8] =  1.;
c[9] =  1.;
a[2][1] =  .6e-1;
a[3][1] =  .19239962962962962962962962962962962962962296e-1;
a[3][2] =  .76693370370370370370370370370370370370370037e-1;
a[4][1] =  .35975e-1;
a[4][2] =  0.;
```

```
a[4][3] =  .107925;
a[5][1] =  1.3186834152331482609197472764317356128 61;
a[5][2] =  0.;
a[5][3] = -5.042058063628562225427761634715637693344;
a[5][4] =  4.220674648395413964508014358283902080483;
a[6][1] = -41.872591664327514618037577806443468129 05;
a[6][2] =  0.;
a[6][3] =  159.432562163137491770036566907034683045 3;
a[6][4] = -122.119213565010030920251620338924214066 3;
a[6][5] =  5.531743066200053768252631238332999150076;
a[7][1] = -54.430156935316504332506420512941424612 71;
a[7][2] =  0.;
a[7][3] =  207.067251365018464427365717386650983598 7;
a[7][4] = -158.610813784589999182874242436505859946 9;
a[7][5] =  6.991816585950242321992597280791793907096;
a[7][6] = -.185972310622032339776517179954929462369 2e-1;

a[8][1] = -54.663741787281976802412156480503869593 51;
a[8][2] =  0.;
a[8][3] =  207.952806255389373451582481669983424423 8;
a[8][4] = -159.288957474499507150895980587142665421 6;
a[8][5] =  7.018743740796944434698170760964252490817;
a[8][6] = -.183387859050457230647278200514173826836 1e-1;
a[8][7] = -.511948499788209907787543249724516839584 0e-3;

a[9][1] =  .343895786835703600927882012472832238652 0e-1;
a[9][2] =  0.;
a[9][3] =  0.;
a[9][4] =  .258262455563350340465955809858612085876 7;
a[9][5] =  .420937118967353715064255151406980196703 2;
a[9][6] =  4.405396469669310170148836816197095664891;
a[9][7] = -176.483119024298657615174094249900212502 9;
a[9][8] =  172.364133401415073029402258271190241331 5;

bh1[1] =  .343895786835703600927882012472832238652 0e-1;
bh1[2] =  0.;
bh1[3] =  0.;
bh1[4] =  .258262455563350340465955809858612085876 7;
bh1[5] =  .420937118967353715064255151406980196703 2;
bh1[6] =  4.405396469669310170148836816197095664891;
bh1[7] = -176.483119024298657615174094249900212502 9;
bh1[8] =  172.364133401415073029402258271190241331 5;
bh1[9] =  0.;

bh[ 1] =  .490996764838248973090685492797122583647 9e-1;
bh[ 2] =  0.;
bh[ 3] =  0.;
bh[ 4] =  .225111222951652415340139532053987532948 5;
bh[ 5] =  .469468225302956203943194852504738741255 3;
bh[ 6] =  .806579224998886770763416180899521798144 3;
bh[ 7] =  0.;
bh[ 8] = -.607119489177795979767295146525621712248 8;
bh[ 9] =  .568611394404756924114760317876613815359 4e-1;
double b[][]={bh1,bh};
set(a,b,c);
  }
  else if(isim.equals("RKCK45"))
  { // Runge-Kutta-Fehlberg, Cash-Karp, Dormant-Price, Verner
  int s=7;
double dydx=1.0;
double c[]=new double[s];
double a[][]=new double[s][s];
double bh1[]=new double[s];
double bh2[]=new double[s];
c[2] =  0.2;
c[3] =  0.3;
c[4] =  0.6;
c[5] =  1;
c[6] = 7.0/8.0;

a[1][1] = 0;
```

```
a[2][1] =  0.2;

a[3][1] =  3.0/40.0;
a[3][2] =  9.0/40.0;

a[4][1] =  0.3;
a[4][2] =  -9.0/10.0;
a[4][3] =  6.0/5.0;

a[5][1] =  -11.0/54.0;
a[5][2] =  0.25;
a[5][3] =  -70.0/27.0;
a[5][4] =  35.0/27.0;

bh1[1] =  37.0/378.0;
bh1[2] =  250.0/621.0;
bh1[3] =  125.0/591.0;
bh1[4] =  512.0/1771.0;
bh1[5] =  0.0;

bh2[1] =  2825.0/27648.0;
bh2[2] =  18575.0/48384.0;
bh2[3] =  13525.0/55296.0;
bh2[4] =  277.0/14336.0;
bh2[5] =  1.0/4.0;
double b[][]={bh1,bh2};
set(a,b,c);
 }
 else if(isim.equals("RKDP45"))
 {// Runge-Kutta Dormant-Prince order 4-5
 int s=8;
double dydx=1.0;
double c[]=new double[s];
double a[][]=new double[s][s];
double bh1[]=new double[s];
double bh2[]=new double[s];
c[1]=0.0e0;
    c[2]=0.2;
    c[3]=0.3;
    c[4]=0.8;
    c[5]=8.0/9.0;
    c[6]=1;
    c[7]=1;
    a[2][1]=0.2;
    a[3][1]=3.0/40.0;
    a[3][2]=9.0/40.0;
    a[4][1]=44.0/45.0;
    a[4][2]=-56.0/15.0;
    a[4][3]=32.0/9.0;
    a[5][1]=19372.0/6561.0;
    a[5][2]=-25360.0/2187.0;
    a[5][3]=64448.0/6561.0;
    a[5][4]=-212.0/729.0;
    a[6][1]=9017.0/3168.0;
    a[6][2]=-355.0/33.0;
    a[6][3]=46732.0/5247.0;
    a[6][4]=49.0/176.0;
    a[6][5]=-5103.0/18656.0;
    a[7][1]=35.0/384.0;
    a[7][3]=500.0/1113.0;
    a[7][4]=125.0/192.0;
    a[7][5]=-2187.0/6784.0;
    a[7][6]=11.0/84.0;
    bh1[1]=5179.0/57600.0;
    bh1[2]=0;
    bh1[3]=7571.0/16695.0;
    bh1[4]=393.0/640.0;
    bh1[5]=-92097.0/339200.0;
    bh1[6]=187.0/2100.0;
    bh1[7]=1.0/40.0;
```

```
    bh2[1]=35.0/384.0;
    bh2[2]=0;
    bh2[3]=500.0/1113.0;
    bh2[4]=125.0/192.0;
    bh2[5]=-2187.0/6784.0;
    bh2[6]=11.0/84.0;
    bh2[7]=0.0;
    double b[][]={bh1,bh2};
    set(a,b,c);
  }
  else if(isim.equals("RKV13"))
  { //Runge-Kutta-Verner 13üncü derece sabit adımlı metodu
int s=14;
double c[]=new double[s];
double a[][]=new double[s][s];
double bh[]=new double[s];
c[0]=0;
c[1] =  0.;
c[2] =  .5e-1;
a[2][1] =  .5e-1;
c[3] =  .1065625;
a[3][1] = -.69931640625e-2;
a[3][2] =  .1135556640625;
c[4] =  .15984375;
a[4][1] =  .399609375e-1;
a[4][2] =  0.;
a[4][3] =  .1198828125;
c[5] =  .39;
a[5][1] =  .36139756280045751240529407211840283451 29;
a[5][2] =  0.;
a[5][3] = -1.34152406670049277181998778820271583491 7;
a[5][4] =  1.37012650390003525941469371608431300040 4;
c[6] =  .465;
a[6][1] =  .49047202797202797202797202797202797202 80e-1;
a[6][2] =  0.;
a[6][3] =  0.;
a[6][4] =  .23509720422144047398629883354934271431 22;
a[6][5] =  .18085559298135672881090396365345448841 50;
//
c[7] =  .155;
a[7][1] =  .61692890442890442890442890442890442890 44e-1;
a[7][2] =  0.;
a[7][3] =  0.;
a[7][4] =  .11236568314640276622625570351301544230 3;
a[7][5] = -.38850460714513667670490481081112445674 56e-1;
a[7][6] =  .19791887125220458553791887125220458553 79e-1;
c[8] =  .943;
a[8][1] = -1.76763024022232687573559711957214558671 4;
a[8][2] =  0.;
a[8][3] =  0.;
a[8][4] = -62.5;
a[8][5] = -6.06188937737666910082136145965933199975 8;
a[8][6] =  5.65082319822276313856129803060084017420 1;
a[8][7] =  65.62169641937623283799566054863063741227;
c[9] =  .90180204173585695825970794067837214995 60;
a[9][1] = -1.18094506655497079982511628262829795788 2;
a[9][2] =  0.;
a[9][3] =  0.;
a[9][4] = -41.50473441114320841606641502701994225874;
a[9][5] = -4.43443831910372501122516922984610021177 6;
a[9][6] =  4.26040818858613302481219371074469324076 1;
a[9][7] =  43.75364022446171584987676829438379303004;
a[9][8] =  .78714254899123106874464750442263075508 60e-2;
c[10] =  .909;
a[10][1] = -1.28140599944148840545951029118205424626 6;
a[10][2] =  0.;
a[10][3] =  0.;
a[10][4] = -45.04713996013986630220754257136007322267;
a[10][5] = -4.73136206944957647731146426549128281094 3;
a[10][6] =  4.51496701659380784118585158459724099621 4;
a[10][7] =  47.44909557172985134869022392235929015114;
```

```
a[10][8] =  .10592282971116611356873939555165428752 28e-1;
a[10][9] = -.57468422638446162544323184782862962320 21e-2;
c[11] =  .94;
a[11][1] = -1.72447013426248519175670981748448186 1731;
a[11][2] =  0.;
a[11][3] =  0.;
a[11][4] = -60.92349008483054016518434619253765246063;
a[11][5] = -5.95151837622239245520283276706185486829 0;
a[11][6] =  5.55652373069845623597979165084359249683 9;
a[11][7] =  63.98301198033305336837536378635995939281;
a[11][8] =  .14642028250414961592759213917594526760 03e-1;
a[11][9] =  .64604087723582036036218651449776507148 92e-1;
a[11][10] = -.79303231690088789840244525486933732914 47e-1;
c[12] =  1.;
a[12][1] = -3.30162266774707901635399478979098362556 9;
a[12][2] =  0.;
a[12][3] =  0.;
a[12][4] = -118.01127235975250856669233039578988685 10;
a[12][5] = -10.14142238845611248642783916034510897595;
a[12][6] =  9.13931133223205792354401227355682700061 9;
a[12][7] =  123.37594282840426836848471809865018943 64;
a[12][8] =  4.62324437887458047483980762506763092479 2;
a[12][9] = -3.38327773806820192365255097153681124081 4;
a[12][10] =  4.52759210032461818945126533935112903532 5;
a[12][11] = -5.82849548581162296319308801916298570375 5;
c[13] =  1.;
a[13][1] = -3.03951503376630903004010285182120025105 6;
a[13][2] =  0.;
a[13][3] =  0.;
a[13][4] = -109.26086808941762546864441923221646233 52;
a[13][5] = -9.29064249740029344971766554265689754915 8;
a[13][6] =  8.43050498176491114213429925383616780345 4;
a[13][7] =  114.20100103783313135574240410955234274 76;
a[13][8] = -.96372713421454793581623756589879016527 62;
a[13][9] = -5.03488408880218979119868033618333232311 8;
a[13][10] =  5.95813082400292317754040216538817207279 4;
a[13][11] =  0.;
a[13][12] =  0.;
bh[1] =  .44312615229089792124864365102090297648 93e-1;
bh[2] =  0.;
bh[3] =  0.;
bh[4] =  0.;
bh[5] =  0.;
bh[6] =  .35460956423432264478631793508950550388 55;
bh[7] =  .24784804313666530696199867215044586600 16;
bh[8] =  4.44813473247578449272512831715964887131 2;
bh[9] =  19.84688636611873369930932399297687935291;
bh[10] = -23.58162337746561841969517960870394965085;
bh[11] =  0.;
bh[12] =  0.;
bh[13] = -.36016794372897751621245367377462024091 10;
double b[][]={bh};
set(a,b,c);
  }
}
}
```

Rınge_Kutta sınıfı bir çok gaye için kullanabileceğimiz genel bir sınıf olarak oluşturuldu. Bu sınıfa daha sonraki bölümlerde göreceğimiz birden fazla diferansiyel denklemi çözebilecek metodları da ilave ettik.

Program 8.4.2 Runge_Kutta sınıfı

```
 import java.io.*;
import java.text.*;
import java.util.Locale;

abstract class f_xy
{
  //single function single independent variable
  // example f=x*x
  abstract double func(double x,double y);
```

```
}
abstract class f_xi
{
  // çok fonksiyonlu ve çok denklemli fonksiyon
  // denklemlerin sadece bir tanesi denklem_ref
  // tarafından seçilir.
  // örnek   f[0]=x[0]+sin(x[1])
  //         f[1]=x[0]*x[0]-x[1]
  // func(x,1)  f[1] değerini verir
  // func(x,0)  f[0] değerini verir

  abstract double func(double x[],int denklem_ref);
}
public class Runge_Kutta
{
double a[][];
double b[][];
double c[];

public Runge_Kutta(double ai[][],double bi[][],double ci[])
{a=ai;b=bi;c=ci;}

public Runge_Kutta(Buthcher bu)
{a=bu.a;b=bu.b;c=bu.c;}
public Runge_Kutta(String isim)
{Buthcher bu=new Buthcher(isim);
 a=bu.a;b=bu.b;c=bu.c;
}
public double[][] hesapla(f_xy f,double x0,double xn,double y0,double h)
{return hesapla(f,x0,xn,y0,h,0);}

public double[][] hesapla(f_xy f,double x0,double xn,double y0,double h,int bk)
{
  int n=(int)((xn-x0)/h);
  double x[]=new double[n+1];
  double y[]=new double[n+1];
  int s=c.length;
  double xi;
  double yi;
  double ki[]=new double[s];
  x[0]=x0;y[0]=y0;
  for(int ni=0;ni<n;ni++)
  {for(int i=0;i<s;i++)
   { xi=x[ni]+c[i]*h;
     yi=y[ni];
    for(int j=0;j<i;j++)
    {yi+=h*a[i][j]*ki[j];//System.out.println("xi(0)="+xi+"xi(j)="+yi+"i="+i+"j="+j+"ki="+ki[j]);
    }
    ki[i]=f.func(xi,yi);
    //System.out.println("ki="+ki[i]);
   }
   y[ni+1]=y[ni];
   for(int i=0;i<s;i++)
    {y[ni+1]+=h*b[bk][i]*ki[i];}
   x[ni+1]=x[ni]+h;
  }
  double XY[][]=new double[2][n+1];
  XY[0]=x;
  XY[1]=y;
  return XY;
}
public static double[] tasi(double x1,double y1[])
{
int n=y1.length;
double z[]=new double[n+1];
z[0]=x1;
for(int i=1;i<=n;i++)
{z[i]=y1[i-1];}
return z;
}
//çok boyutlu RK girişler N(toplam veri sayısı) veya h(step boyu) üzeriinden verilebilmektedir.
```

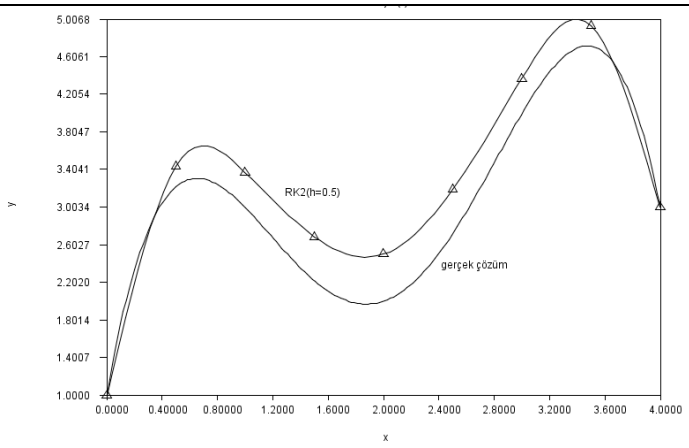
```
public double[][] hesapla(f_xi f,double x0,double xn,double y0[],double h)
{return hesapla(f,x0,xn,y0,h,0);}
public double[][] hesapla(f_xi f,double x0,double xn,double y0[],int ni)
{double h=(xn-x0)/ni;
return hesapla(f,x0,xn,y0,h,0);
}
public double[][] hesapla(f_xi f,double x0,double xn,double y0[],int ni,int bk)
{double h=(xn-x0)/ni;
return hesapla(f,x0,xn,y0,h,bk);
}
public double[][] hesapla(f_xi f,double x0,double xn,double y0[],double h,int bk)
{
  int n=(int)((xn-x0)/h);
  int m=y0.length;
  double x[]=new double[n+1];
  double y[][]=new double[n+1][m];
  int s=c.length;
  double xi;
  double yi;
  double z[]=new double[m+1];
  double ki[]=new double[s];

  x[0]=x0;
  for(int ni=0;ni<=n;ni++)
  {x[ni]=x0+ni*h;}
  y[0]=y0;
  for(int ni=0;ni<n;ni++)
  { for(int k=0;k<m;k++)
                {
     for(int i=0;i<s;i++)
     { xi=x[ni]+c[i]*h;
       z=tasi(xi,y[ni]);
       for(int j=0;j<i;j++)
         {z[k+1]+=h*a[i][j]*ki[j];//System.out.println("xi(0)="+z[0]+"xi(j)="+z[k+1]+"i="+i+"j="+j+"ki="+ki[j]);
         }
       ki[i]=f.func(z,k);
       //System.out.println("ki="+ki[i]);
     }
   y[ni+1][k]=y[ni][k];
    for(int i=0;i<s;i++)
    {y[ni+1][k]+=h*b[bk][i]*ki[i];}
  }
  }
  double XY[][]=new double[m+1][n+1];
  XY[0]=x;
  for(int k=0;k<m;k++)
   { for(int ni=0;ni<=n;ni++)
     {XY[k+1][ni]=y[ni][k];}
   }
  return XY;
}

public static String toString(double left, int w, int d)
// converts a double to a string with given width and decimals.
{
                NumberFormat df=NumberFormat.getInstance(Locale.US);
                df.setMaximumFractionDigits(d);
    df.setMinimumFractionDigits(d);
    df.setGroupingUsed(false);
    String s = df.format(left);
    while (s.length() < w)
      s = " " + s;
    if (s.length() > w)
    {
       s = "";
       for (int i=0; i<w; i++)
         s = s + "-";
    }
    return s;
}
public String toString()
```

```
 {int m=b[0].length;
  int n=c.length;
  int k=b.length;
  String s="Buthcher tablosu n = "+n+"\n";
  for(int i=0;i<n;i++)
  {s+=toString(c[i],15,8)+" | ";
  for(int j=0;j<i;j++)
  {s+=toString(a[i][j],15,8)+" ";}
  s+="\n";
  }
  for(int j=0;j<m;j++)
  {s+="----------";}
  s+="\n                ";
  for(int i=0;i<k;i++)
  {
  for(int j=0;j<m;j++)
  {s+=toString(b[i][j],15,8)+" ";}
   s+="\n                ";
  }
   return s;
 }
}
```

Program 8.4.3 ikinci dereceden Runge-Kutta denklemi

```
 import java.io.*;

abstract class f_xy
{
  //single function single independent variable
  // example f=x*x
  abstract double func(double x,double y);
}

class fxy extends f_xy
{ //diferansiyeli alınan fonksiyon
  public double func(double x,double y)
  {return -2*x*x*x+12*x*x-20*x+8.5;}
}

class fxyKC extends f_x
{ //Gerçek çözüm fonksiyonu
  public double func(double x)
  {return -0.5*x*x*x*x+4.0*x*x*x-10.0*x*x+8.5*x+1.0;}
}

class dif1
{
  public static void main(String args[]) throws IOException
  {
   double x0=0;
   double y0=1.0;
   double xn=4.0;
   fxy f1=new fxy();
   fxyKC f2=new fxyKC();
   Runge_Kutta rk=new Runge_Kutta("RK2");
   System.out.println(rk.toString());
   double XY[][]=rk.hesapla(f1,x0,xn,y0,0.5);
   System.out.println(Matrix.toString(Matrix.Transpose(XY)));
   Plot p1=new Plot(XY);
   p1.addSpline(XY,10);
   p1.addFunction(f2,x0,xn,200);
   p1.setPlotType(0,24);
   //p1.setPlotType(1,0);
```

```
  p1.plot();
 }
}
```



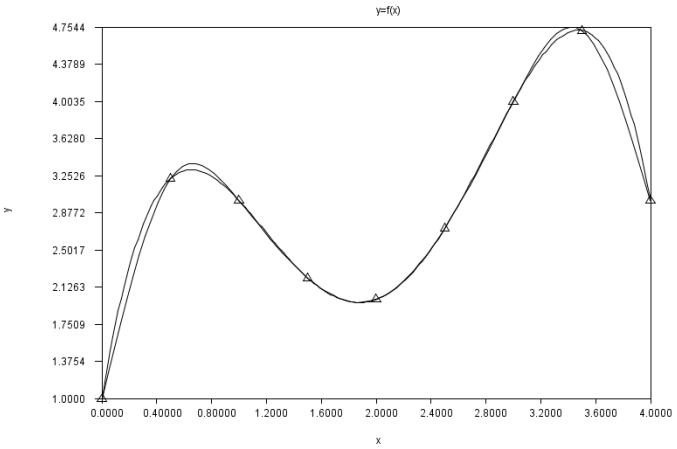
(bolum8_007.jpg,Resimaltı: **ikinci dereceden Runge-Kutta denklemi, h=0.5)**

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" dif1
Buthcher tablosu n = 2
   0.00000000 |
   1.00000000 |     1.00000000
-------------------------------------
                0.50000000     0.50000000

     0.000000000000000      1.000000000000000
     0.500000000000000      3.437500000000000
     1.000000000000000      3.375000000000000
     1.500000000000000      2.687500000000000
     2.000000000000000      2.500000000000000
     2.500000000000000      3.187500000000000
     3.000000000000000      4.375000000000000
     3.500000000000000      4.937500000000000
     4.000000000000000      3.000000000000000
```

Program 8.4.4 Üçüncü dereceden Runge-Kutta denklemi h=0.5

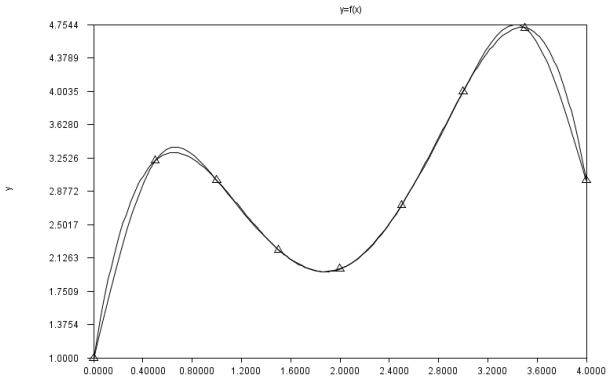
```
 import java.io.*;

abstract class f_xy
{
  //single function single independent variable
  // example f=x*x
  abstract double func(double x,double y);
}

class fxy extends f_xy
{ //diferansiyeli alınan fonksiyon
  public double func(double x,double y)
  {return -2*x*x*x+12*x*x-20*x+8.5;}
}

class fxyKC extends f_x
{ //Gerçek çözüm fonksiyonu
  public double func(double x)
  {return -0.5*x*x*x*x+4.0*x*x*x-10.0*x*x+8.5*x+1.0;}
}
```

```
class dif1
{
 public static void main(String args[]) throws IOException
 {
  double x0=0;
  double y0=1.0;
  double xn=4.0;
  fxy f1=new fxy();
  fxyKC f2=new fxyKC();
  Runge_Kutta rk=new Runge_Kutta("RK3");
  System.out.println(rk.toString());
  double XY[][]=rk.hesapla(f1,x0,xn,y0,0.5);
  System.out.println(Matrix.toString(Matrix.Transpose(XY)));
  Plot p1=new Plot(XY);
  p1.addSpline(XY,10);
  p1.addFunction(f2,x0,xn,200);
  p1.setPlotType(0,24);
  //p1.setPlotType(1,0);
  p1.plot();
 }
}
```



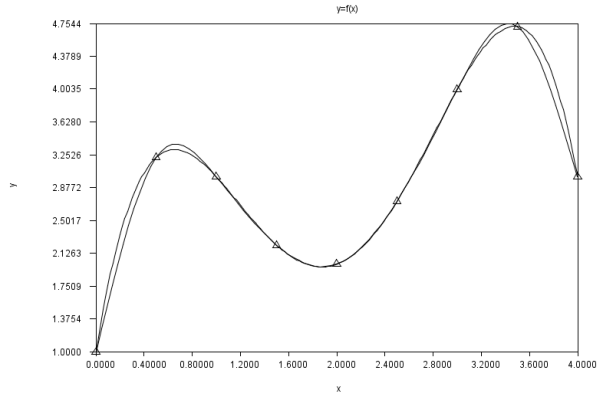
(bolum8_008.jpg,Resimaltı: **Üçüncü dereceden Runge-Kutta denklemi,h=0.5)**

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" dif1
Buthcher tablosu n = 3
   0.00000000 |
   0.50000000 |     0.50000000
   1.00000000 |    -1.00000000     2.00000000
------------------------------------------------------------
                0.16666667     0.66666667     0.16666667

     0.000000000000000      1.000000000000000
     0.500000000000000      3.218750000000000
     1.000000000000000      3.000000000000000
     1.500000000000000      2.218750000000000
     2.000000000000000      2.000000000000000
     2.500000000000000      2.718750000000000
     3.000000000000000      4.000000000000000
     3.500000000000000      4.718750000000000
     4.000000000000000      3.000000000000000
```



(bolum8_009.jpg,Resimaltı: **Dördüncü dereceden Runge-Kutta denklemi h=0.5)**

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" dif1
Buthcher tablosu n = 4
   0.00000000 |
   0.50000000 |    0.50000000
   0.50000000 |    0.00000000    0.50000000
   1.00000000 |    0.00000000    0.00000000    1.00000000
--------------------------------------------------------------------------
             0.16666667    0.33333333    0.33333333    0.16666667

    0.000000000000000     1.000000000000000
    0.500000000000000     3.218750000000000
    1.000000000000000     2.999999999999999
    1.500000000000000     2.218750000000000
    2.000000000000000     2.000000000000000
    2.500000000000000     2.718750000000000
    3.000000000000000     4.000000000000000
    3.500000000000000     4.718750000000000
    4.000000000000000     3.000000000000001
```



(bolum8_010.jpg,Resimaltı: **Altıncı dereceden Runge-Kutta (Ralston) denklemi, h=0.5)**

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" dif1
Buthcher tablosu n = 6
   0.00000000 |
   0.25000000 |    0.25000000
   0.25000000 |    0.12500000    0.12500000
   0.50000000 |    0.00000000   -0.50000000    1.00000000
   0.75000000 |    0.18750000    0.00000000    0.00000000    0.56250000
   1.00000000 |   -0.42857143    0.28571429    1.71428571   -1.71428571    1.14285714
-------------------------------------------------------------------------------------------------
                0.07777778    0.00000000    0.35555556    0.13333333    0.35555556    0.07777778

    0.000000000000000     1.000000000000000
    0.500000000000000     3.218750000000000
    1.000000000000000     3.000000000000001
```

1.500000000000000	2.218750000000001
2.000000000000000	2.000000000000001
2.500000000000000	2.718750000000001
3.000000000000000	4.000000000000002
3.500000000000000	4.718750000000002
4.000000000000000	3.000000000000002

ÖRNEK PROBLEM :

Aşağıdaki diferansiyel denklemi

$\frac{dy}{dx} = yx^2 - y$ x=0 ve x=2 aralığında, y(0)=1. Başlangıç koşulu ile

a) Analitik olarak çözünüz
b) h=0.5 adım aralığıyla 4th derece Runge_Kutte metoduyla elde çözünüz
c) h=0.1 adım aralığıyla 4th derece Runge_Kutte metoduyla elde çözünüz
d) grafiğini c.zerek sonuçları karşılaştırınız

$$\frac{dy}{dx} = yx^2 - y = y(x^2 - 1)$$

$$\frac{dy}{y} = (x^2 - 1)dx$$

$$\int \frac{dy}{y} = \int (x^2 - 1)dx$$

$$\ln(y) = (\frac{x^3}{3} - x) + C$$

Boundary condition x=0 y=1➔ C=0

$$y = \exp(\frac{x^3}{3} - x)$$

çözüm (excel):

h	**0.5**														
	x1	y1	k1	x2	y2	k2	x3	y3	k3	x4	y4	k4	y	y exact	error
	0	1	-1	0.25	0.75	-0.703	0.25	0.8242	-0.773	0.5	0.6136	-0.46	1	1	0
	0.5	0.6323	-0.474	0.75	0.5138	-0.225	0.75	0.5761	-0.252	1	0.5063	0	0.6323	0.6323	-5.4E-06
	1	0.5133	0	1.25	0.5133	0.2888	1.25	0.5855	0.3294	1.5	0.678	0.8475	0.5133	0.5134	7.02E-05
	1.5	0.687	0.8587	1.75	0.9017	1.8597	1.75	1.1519	2.3758	2	1.8749	5.6248	0.687	0.6873	0.000294
	2	1.9332	5.7996	2.25	3.3831	13.744	2.25	5.3692	21.812	2.5	12.839	67.407	1.9332	1.9477	0.01452

h	**0.1**														
	x1	y1	k1	x2	y2	k2	x3	y3	k3	x4	y4	k4	y	y exact	error
	0	1	-1	0.05	0.95	-0.948	0.05	0.9526	-0.95	0.1	0.905	-0.896	1	1	0
	0.1	0.9051	-0.896	0.15	0.8603	-0.841	0.15	0.8631	-0.844	0.2	0.8208	-0.788	0.9051	0.9051	-7E-08
	0.2	0.8209	-0.788	0.25	0.7815	-0.733	0.25	0.7843	-0.735	0.3	0.7474	-0.68	0.8209	0.8209	-1.2E-07
	0.3	0.7475	-0.68	0.35	0.7135	-0.626	0.35	0.7162	-0.628	0.4	0.6847	-0.575	0.7475	0.7475	-1.5E-07
	0.4	0.6848	-0.575	0.45	0.656	-0.523	0.45	0.6586	-0.525	0.5	0.6322	-0.474	0.6848	0.6848	-1.6E-07
	0.5	0.6323	-0.474	0.55	0.6086	-0.425	0.55	0.6111	-0.426	0.6	0.5897	-0.377	0.6323	0.6323	-1.6E-07
	0.6	0.5898	-0.377	0.65	0.5709	-0.33	0.65	0.5733	-0.331	0.7	0.5567	-0.284	0.5898	0.5898	-1.5E-07
	0.7	0.5567	-0.284	0.75	0.5425	-0.237	0.75	0.5449	-0.238	0.8	0.5329	-0.192	0.5567	0.5567	-1.4E-07
	0.8	0.5329	-0.192	0.85	0.5234	-0.145	0.85	0.5257	-0.146	0.9	0.5184	-0.098	0.5329	0.5329	-1.3E-07
	0.9	0.5184	-0.098	0.95	0.5135	-0.05	0.95	0.5159	-0.05	1	0.5134	0	0.5184	0.5184	-1.2E-07
	1	0.5134	0	1.05	0.5134	0.0526	1.05	0.516	0.0529	1.1	0.5187	0.1089	0.5134	0.5134	-1.2E-07
	1.1	0.5188	0.1089	1.15	0.5242	0.1691	1.15	0.5272	0.17	1.2	0.5358	0.2357	0.5188	0.5187	-1.2E-07

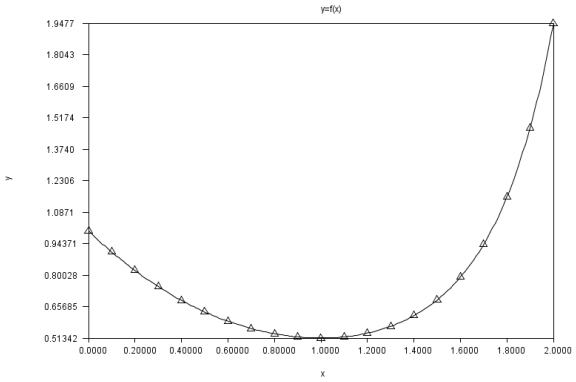
	1.2	0.5358	0.2358	1.25	0.5476	0.308	1.25	0.5512	0.31	1.3	0.5668	0.3911	0.5358	0.5358	-1.2E-07
	1.3	0.5668	0.3911	1.35	0.5864	0.4823	1.35	0.591	0.4861	1.4	0.6155	0.5908	0.5668	0.5668	-1.1E-07
	1.4	0.6155	0.5909	1.45	0.645	0.7112	1.45	0.651	0.7178	1.5	0.6873	0.8591	0.6155	0.6155	-6.9E-08
	1.5	0.6873	0.8591	1.55	0.7302	1.0242	1.55	0.7385	1.0357	1.6	0.7909	1.2337	0.6873	0.6873	7.08E-08
	1.6	0.7908	1.2337	1.65	0.8525	1.4685	1.65	0.8643	1.4887	1.7	0.9397	1.776	0.7908	0.7908	5.25E-07
	1.7	0.9396	1.7758	1.75	1.0284	2.121	1.75	1.0456	2.1566	1.8	1.1552	2.5877	0.9396	0.9396	1.9E-06
	1.8	1.1549	2.5869	1.85	1.2842	3.111	1.85	1.3104	3.1745	1.9	1.4723	3.8428	1.1549	1.1549	5.82E-06
	1.9	1.4716	3.8408	1.95	1.6636	4.6622	1.95	1.7047	4.7773	2	1.9493	5.8479	1.4716	1.4716	1.67E-05
	2	1.9477	5.8431	2.05	2.2398	7.1731	2.05	2.3063	7.3861	2.1	2.6863	9.1603	1.9477	1.9477	4.59E-05

Program 8.4.5 Örnek problemin çözümü Runge-Kutta (RK4) denklemi h=0.1

```
//=====================================================
// Numerical Analysis package in java
// example to differential equation solution
// Runge-Kutta metodları
// Dr. Turhan Coban
// =====================================================
// Basit diferansiyel denklemler dy/dx=f(x,y)

import java.io.*;

abstract class f_xy
{
 //single function single independent variable
 // example f=x*x
 abstract double func(double x,double y);
}

class fxy extends f_xy
{
 public double func(double x,double y)
 {return y*x*x-y;}
}

class fxyKC extends f_x
{
 public double func(double x)
 {return Math.exp(x*x*x/3.0-x);}
}


class dif1
{
 public static void main(String args[]) throws IOException
 {
  double x0=0;
  double y0=1.0;
  double xn=2.0;
  fxy f1=new fxy();
  fxyKC f2=new fxyKC();
  Runge_Kutta rk=new Runge_Kutta("RK4");
  System.out.println(rk.toString());
  double XY[][]=rk.hesapla(f1,x0,xn,y0,0.1);
  System.out.println(Matrix.toString(Matrix.Transpose(XY)));
  Plot p1=new Plot(XY);
  //p1.addSpline(XY,10);
  p1.addFunction(f2,x0,xn,200);
  p1.setPlotType(0,24);
  //p1.setPlotType(1,0);
  p1.plot();
 }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" dif1
Buthcher tablosu n = 4
```

```
   0.00000000 |
   0.50000000 |    0.50000000
   0.50000000 |    0.00000000    0.50000000
   1.00000000 |    0.00000000    0.00000000    1.00000000
------------------------------------------------------------------------
              0.16666667    0.33333333    0.33333333    0.16666667

    0.000000000000000    1.000000000000000
    0.100000000000000    0.905139151280990
    0.200000000000000    0.820917069099996
    0.300000000000000    0.747515827045378
    0.400000000000000    0.684773992072904
    0.500000000000000    0.632336819274610
    0.600000000000000    0.589783505270557
    0.700000000000000    0.556734717986668
    0.800000000000000    0.532947107134396
    0.900000000000000    0.518404336705643
    1.000000000000000    0.513417235958808
    1.100000000000000    0.518750050786687
    1.200000000000000    0.535797072713138
    1.300000000000000    0.566846645557574
    1.400000000000000    0.615492067374456
    1.500000000000000    0.687289208016086
    1.600000000000000    0.790833891701643
    1.700000000000000    0.939567747881861
    1.800000000000000    1.154878288457116
    1.900000000000001    1.471558461178070
    2.000000000000000    1.947688122952552
```



(bolum8_011.jpg,Resimaltı: **Örnek problemin çözümü Runge-Kutta (RK4) denklemi h=0.1)**

9.5 Çok adımlı yöntemler

Birden fazla denklemin bir arada kullanıldığı diferansiyel denklem çözüm yöntemlerine çok adımlı yöntemler denir. Bu tür metodların cözüm kararlılıklarının daha iyi olduğu yönünde görüşler mevcuttur.
İlk olarak **Milne** formülüne göz atalım

Milne 1. adım formülü (ilk tahmin)

$$y_{i+1}^{p} = y_{i-3} + \frac{4h}{3}\left[2f(x_i, y_i) - f(x_{i-1}, y_{i-1}) + 2f(x_{i-2}, y_{i-2})\right]$$

Milne 2. adım formülü (ilk düzeltme)

$$y_{i+1}^{(0)} = y_{j+1}^{p} + \frac{28}{29}\left[y_i - y_i^{p}\right]$$

Milne 3. adım formülü (düzeltme)

$$y_{i+1}^{(k+1)} = y_{j-1} + \frac{h}{3}\left[f(x_{i+1}, y_{i+1}^{(k)}) + 4f(x_i, y_i) + f(x_{i-1}, y_{i-1}) \right]$$ k=0,1,....

Bu metodları uygulayabilmek için daha önceden hesaplanmış noktalara ihtiyaç mevcuttur. Daha önceki noktalar tek adımlı metodlarla (örneğin Runge-Kutta metodlarıyla) hesaplanarak başlanabilir. Örnek olarak

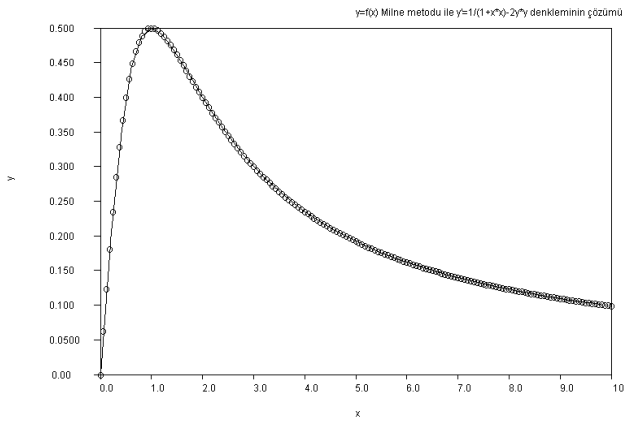
$y' = \frac{1}{1+x^2} - 2y^2$ denklemini y(0)=0 için çözelim. Bu diferansiyel denklemin tam analitik çözümü :

$y = \frac{x}{1+x^2}$ dir. Bu çözümü kullanarak hata miktarını da inceleyebiliriz.

Program 9.5.1 Milne Tahmin-düzeltme (Predictor-corrector) formülü

```
//=====================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
// Basit diferansiyel denklemler dy/dx=f(x,y)

import java.io.*;

abstract class f_xy
{
 //single function single independent variable
 // example f=x*x
 abstract double func(double x,double y);
}

class fxy extends f_xy
{
 double func(double x,double y)
 {return 1.0/(1.0+x*x)-2.0*y*y;}
}

class fxyKC extends f_x
{
 double func(double x)
 {return x/(1+x*x);}
}

class diferansiyelN
{
 public static double[][] RK6M(f_xy f,double x0,double y0,double xn,double h)
 {
 // Milne formülü
 int n=(int)((xn-x0)/h);
 double a[][]=new double[2][n+1];
 double yp[]=new double[n+1];
 double y[][]=new double[6][n+1];
 // Runge-Kutta 6
 int i=0;
 a[0][0]=x0;
 a[1][0]=y0;
 double x2=x0+3.0*h;
 double k0,k1,k2,k3,k4,k5,k6;
 for(double xx=x0;xx<x2;xx+=h)
  {a[0][i+1]=xx+h;
  k1=f.func(a[0][i],a[1][i]);
```

```
  k2=f.func(a[0][i]+0.25*h,a[1][i]+0.25*k1*h);
  k3=f.func(a[0][i]+0.25*h,a[1][i]+1.0/8.0*k1*h+1.0/8.0*k2*h);
  k4=f.func(a[0][i]+0.5*h,a[1][i]-0.5*k2*h+k3*h);
  k5=f.func(a[0][i]+3.0/4.0*h,a[1][i]+3.0/16.0*k1*h+9.0/16.0*k4*h);
  k6=f.func(a[0][i]+h,a[1][i]-3.0/7.0*k1*h+2.0/7.0*k2*h+12.0/7.0*k3*h-12.0/7.0*k4*h+8.0/7.0*k5*h);
  a[1][i+1]=a[1][i]+(7.0*k1+32.0*k3+12.0*k4+32.0*k5+7*k6)/90.0*h;
  i++;
  }
  for(double xx=x2;xx<xn;xx+=h)
  { a[0][i+1]=xx+h;
  //Milne formülü
  k1=f.func(a[0][i],a[1][i]);
  k2=f.func(a[0][i-1],a[1][i-1]);
  k3=f.func(a[0][i-2],a[1][i-2]);
  yp[i+1]=a[1][i-3]+4.0*h/3.0*(2.0*k1-k2+2.0*k3);
  y[0][i+1]=yp[i+1]+28.0/29.0*(a[1][i]- yp[i]);
  int k;
  for(k=0;k<5;k++)
  {
                k0=f.func(a[0][i+1],y[k][i+1]);
     y[k+1][i+1]=a[1][i-1]+h/3.0*(k0+4.0*k1+k2);
  }
  a[1][i+1]=y[k][i+1];
  i++;
  }
 return a;
 }

 public static void main(String args[]) throws IOException
 {double x0=0;
  double y0=0.0;
  double yn=10.0;
  double h=0.0625;
  fxy f2=new fxy();
  fxyKC fKC=new fxyKC();
  double a[][]=RK6M(f2,x0,y0,yn,h);
  Text.print(Text.T(a));
  Plot p1=new Plot(a);
  p1.setPlotType(0,24);
  p1.addFunction(fKC,0.0,10.0,50);
  p1.plot();
 }
}
```



(bolum8_012.jpg,Resimaltı: **Milne Tahmin-düzeltme (Predictor-corrector) formülü)**

Milne Tahmin-düzeltme (Predictor-corrector) formülü hesaplanan değerler

x	y_çözüm	tam çözüm	hata
0	**0**	**0**	**0**
0.0625	**0.062256809**	**0.062256809**	**1.29392E-10**
0.125	**0.123076922**	**0.123076923**	**6.07481E-10**
0.1875	**0.181132074**	**0.181132075**	**1.36894E-09**
0.25	0.235294614	0.235294118	-4.9588E-07
0.3125	0.284697577	0.284697509	-6.78819E-08
0.375	0.328767367	0.328767123	-2.43534E-07
0.4375	0.367212702	0.367213115	4.1304E-07
0.5	0.399999711	0.4	2.8879E-07
0.5625	0.427298769	0.427299703	9.34563E-07
0.625	0.449437527	0.449438202	6.75003E-07
0.6875	0.466842287	0.466843501	1.2147E-06
0.75	0.479999216	0.48	7.8413E-07
0.8125	0.489410522	0.489411765	1.24273E-06
0.875	0.495574537	0.495575221	6.83803E-07
0.9375	0.498959376	0.498960499	1.1231E-06
1	0.499999514	0.5	4.86142E-07
1.0625	0.499081613	0.499082569	9.56231E-07
1.125	0.496551452	0.496551724	2.72191E-07
1.1875	0.492705844	0.492706645	8.01101E-07
1.25	0.487804796	0.487804878	8.16887E-08
1.3125	0.482065316	0.482065997	6.81141E-07
1.375	0.475675749	0.475675676	-7.3537E-08
1.4375	0.468789209	0.468789809	5.99558E-07
1.5	0.461538657	0.461538462	-1.95371E-07
1.5625	0.454028961	0.454029512	5.51033E-07
1.625	0.446352222	0.446351931	-2.9049E-07
1.6875	0.438578152	0.43857868	5.28107E-07
1.75	0.430769597	0.430769231	-3.6604E-07
1.8125	0.422971217	0.422971741	5.23921E-07
1.875	0.415225342	0.415224913	-4.28026E-07
1.9375	0.40755904	0.407559573	5.33046E-07
2	0.400000481	0.4	-4.8098E-07

İkinci olarak olarak **Hamming** formülüne bakalım. Bu formül Milne formülüne oldukça yakındır. Bu formülün hata miktarı milne formülüne göre biraz daha düşüktür, ancak kararlılık yönünden biraz daha iyidir.

Hamming 1. adım formülü (ilk tahmin)

$$y_{i+1}^{p} = y_{i-3} + \frac{4h}{3}\left[2f(x_i, y_i) - f(x_{i-1}, y_{i-1}) + 2f(x_{i-2}, y_{i-2})\right]$$

Hamming 2. adım formülü (ilk düzeltme)

$$y_{i+1}^{(0)} = y_{j+1}^{p} + \frac{112}{121}\left[y_i - y_i^{p}\right]$$

Hamming 3. adım formülü (düzeltme)

$$y_{i+1}^{(k+1)} = \frac{1}{8}\left(9y_i - y_{i-2}\right) + \frac{3h}{8}\left[f(x_{i+1}, y_{i+1}^{(k)}) + 2f(x_i, y_i) - f(x_{i-1}, y_{i-1})\right] \quad \text{k=0,1,....}$$

Program 8.5.2 Hamming Tahmin-düzeltme (Predictor-corrector) formülü

```
//=======================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
```

```
// Dr. Turhan Coban
// ====================================================
// Basit diferansiyel denklemler dy/dx=f(x,y)

import java.io.*;

abstract class f_xy
{
  //single function single independent variable
  // example f=x*x
  abstract double func(double x,double y);
}

class fxy extends f_xy
{
  double func(double x,double y)
  {return 1.0/(1.0+x*x)-2.0*y*y;}
}

class fxyKC extends f_x
{
  double func(double x)
  {return x/(1+x*x);}
}

class diferansiyelP
{
  public static double[][] RK6M(f_xy f,double x0,double y0,double xn,double h)
  {
  // Hamming formülü
  int n=(int)((xn-x0)/h);
  double a[][]=new double[2][n+1];
  double yp[]=new double[n+1];
  double y[][]=new double[6][n+1];
  // Runge-Kutta 6
  int i=0;
  a[0][0]=x0;
  a[1][0]=y0;
  double x2=x0+3.0*h;
  double k0,k1,k2,k3,k4,k5,k6;
  for(double xx=x0;xx<x2;xx+=h)
   {a[0][i+1]=xx+h;
   k1=f.func(a[0][i],a[1][i]);
   k2=f.func(a[0][i]+0.25*h,a[1][i]+0.25*k1*h);
   k3=f.func(a[0][i]+0.25*h,a[1][i]+1.0/8.0*k1*h+1.0/8.0*k2*h);
   k4=f.func(a[0][i]+0.5*h,a[1][i]-0.5*k2*h+k3*h);
   k5=f.func(a[0][i]+3.0/4.0*h,a[1][i]+3.0/16.0*k1*h+9.0/16.0*k4*h);
   k6=f.func(a[0][i]+h,a[1][i]-3.0/7.0*k1*h+2.0/7.0*k2*h+12.0/7.0*k3*h-12.0/7.0*k4*h+8.0/7.0*k5*h);
   a[1][i+1]=a[1][i]+(7.0*k1+32.0*k3+12.0*k4+32.0*k5+7*k6)/90.0*h;
   i++;
   }
   for(double xx=x2;xx<xn;xx+=h)
   { a[0][i+1]=xx+h;
   //Milne formülü
   k1=f.func(a[0][i],a[1][i]);
   k2=f.func(a[0][i-1],a[1][i-1]);
   k3=f.func(a[0][i-2],a[1][i-2]);
   yp[i+1]=a[1][i-3]+4.0*h/3.0*(2.0*k1-k2+2.0*k3);
   y[0][i+1]=yp[i+1]+112.0/121.0*(a[1][i]- yp[i]);
   int k;
   for(k=0;k<5;k++)
   {
```

```
                k0=f.func(a[0][i+1],y[k][i+1]);
    y[k+1][i+1]=1.0/8.0*(9.0*a[1][i]-a[1][i-2])+3.0*h/8.0*(k0+2.0*k1-k2);
  }
  a[1][i+1]=y[k][i+1];
  i++;
  }
 return a;
 }

 public static void main(String args[]) throws IOException
 {double x0=0;
  double y0=0.0;
  double yn=10.0;
  double h=0.0625;
  fxy f2=new fxy();
  fxyKC fKC=new fxyKC();
  double a[][]=RK6M(f2,x0,y0,yn,h);
  Text.print(Text.T(a));
  Plot p1=new Plot(a);
  p1.setPlotType(0,24);
  p1.addFunction(fKC,0.0,10.0,50);
  p1.plot();
 }
}
```

Çıktı 9.5-2 Hamming Tahmin-düzeltme (Predictor-corrector) formülü hesaplanan değerler

x	y_çözüm	tam çözüm	hata
0	**0**	**0**	**0**
0.0625	**0.062256809**	**0.062256809**	**1.29392E-10**
0.125	**0.123076922**	**0.123076923**	**6.07481E-10**
0.1875	**0.181132074**	**0.181132075**	**1.36894E-09**
0.25	0.235295447	0.235294118	-1.32917E-06
0.3125	0.284699363	0.284697509	-1.85455E-06
0.375	0.328768754	0.328767123	-1.63052E-06
0.4375	0.367213781	0.367213115	-6.6612E-07
0.5	0.399999285	0.4	7.14995E-07
0.5625	0.427297497	0.427299703	2.20588E-06
0.625	0.449434654	0.449438202	3.5483E-06
0.6875	0.466838916	0.466843501	4.58534E-06
0.75	0.479994747	0.48	5.25346E-06
0.8125	0.489406203	0.489411765	5.56192E-06
0.875	0.495569657	0.495575221	5.56428E-06
0.9375	0.498955165	0.498960499	5.33375E-06
1	0.499995055	0.5	4.94536E-06
1.0625	0.499078103	0.499082569	4.46539E-06
1.125	0.496547778	0.496551724	3.94663E-06
1.1875	0.492703217	0.492706645	3.42766E-06
1.25	0.487801944	0.487804878	2.93423E-06
1.3125	0.482063515	0.482065997	2.48172E-06
1.375	0.475673598	0.475675676	2.07773E-06
1.4375	0.468788085	0.468789809	1.7244E-06
1.5	0.461537041	0.461538462	1.42031E-06
1.5625	0.45402835	0.454029512	1.16193E-06
1.625	0.446350987	0.446351931	9.44654E-07
1.6875	0.438577917	0.43857868	7.63473E-07
1.75	0.430768617	0.430769231	6.13446E-07
1.8125	0.422971251	0.422971741	4.89946E-07
1.875	0.415224525	0.415224913	3.88796E-07
1.9375	0.407559266	0.407559573	3.06316E-07
2	0.399999761	0.4	2.3933E-07

Çıktılarda Runge_kutta 6 metoduyla hesaplanan ilk değerler koyu olarak verilmiştir. Üçüncü olarak **Adams-Bashford-Moulton** formüllerini inceleyelim. Bu formüller birbirini tamamlıyan bir tahmin formülü (predictor) ve bir düzeltme formülünün (corrector) beraber kullanılmasıyla oluşur. Adams-Bashford-Moulton modelinde ilk tahmin için Adams-Bashford formüllerinden yararlanılır. Bu formüller

Adams-Bashford 1 adım formülü

$$y_{i+1} = y_i + hf(x_i, y_i)$$

Adams-Bashford 2 adım formülü

$$y_{i+1} = y_i + \frac{h}{2}\left(3f(x_i, y_i) - f(x_{i-1}, y_{i-1})\right)$$

Adams-Bashford 3 adım formülü

$$y_{i+1} = y_i + \frac{h}{12}\left(23f(x_i, y_i) - 16f(x_{i-1}, y_{i-1}) + 5f(x_{i-2}, y_{i-2})\right)$$

Adams-Bashford 4 adım formülü

$$y_{i+1} = y_i + \frac{h}{24}\left(55f(x_i, y_i) - 59f(x_{i-1}, y_{i-1}) + 37f(x_{i-2}, y_{i-2}) - 9f(x_{i-3}, y_{i-3})\right)$$

Adams-Bashford 5 adım formülü

$$y_{i+1} = y_i + \frac{h}{720}\left(1901f(x_i, y_i) - 2774f(x_{i-1}, y_{i-1}) + 2616f(x_{i-2}, y_{i-2}) - 1274f(x_{i-3}, y_{i-3}) + 251f(x_{i-4}, y_{i-4})\right)$$

Şeklindedir. Düzeltme terimi olarak Adams-Moulton formülünü kullanabiliriz. Bu formül değişik polinom kademeleri için :

Adams-Moulton 0 adım formülü

$$y_{i+1} = y_i + hf(x_{i+1}, y_{i+1})$$

Adams-Moulton 1 adım formülü

$$y_{i+1} = y_i + \frac{h}{2}\left(f(x_{i+1}, y_{i+1}) + f(x_i, y_i)\right)$$

Adams-Moulton 2 adım formülü

$$y_{i+1} = y_i + \frac{h}{12}\left(5f(x_{i+1}, y_{i+!}) + 8f(x_i, y_i) - f(x_{i-1}, y_{i-!})\right)$$

Adams-Moulton 3 adım formülü

$$y_{i+1} = y_i + \frac{h}{24}\left(9f(x_{i+1}, y_{i+1}) + 19f(x_i, y_i) - 5f(x_{i-1}, y_{i-1}) + f(x_{i-2}, y_{i-2})\right)$$

Adams-Moulton 4 adım formülü

$$y_{i+1} = y_i + \frac{h}{720}\left(251f(x_{i+1}, y_{i+1}) + 646f(x_i, y_i) - 264f(x_{i-1}, y_{i-1}) + 106f(x_{i-2}, y_{i-2}) - 19f(x_{i-3}, y_{i-3})\right)$$

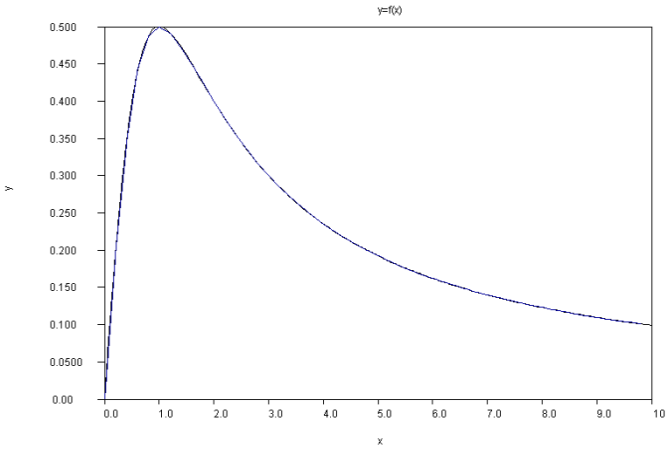
Program listesi aşağıda verilmiştir. Bu programda Adams Bashford 4 adım formülü ile Adams-Milton 3 adım formülünü kullanan bir metod ve Bashford 5 adım formülü ile Adams-Milton 4 adım formülünü kullanan bir metod verilmiştir.

Program 8.5.3 Adams-Bashford-Milton Tahmin-düzeltme (Predictor-corrector) formülü

```
//=====================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
// Basit diferansiyel denklemler dy/dx=f(x,y)
```

```
import java.io.*;

abstract class f_xy
{
 //single function single independent variable
 // example f=x*x
 abstract double func(double x,double y);
}

class fxy extends f_xy
{
 double func(double x,double y)
 {return 1.0/(1.0+x*x)-2.0*y*y;}
}

class fxyKC extends f_x
{
 double func(double x)
 {return x/(1+x*x);}
}

class diferansiyelL
{
 public static double[][] RK6AB4AM3(f_xy f,double x0,double y0,double xn,double h)
 {
 // Adams-Bashford (4 adım formülü)  Adams-Moulton (3 adım formülü) predictor corrector method
 //with RK6 initial step
 int n=(int)((xn-x0)/h);
 double a[][]=new double[2][n+1];
 // Runge-Kutta 6
 int i=0;
 a[0][0]=x0;
 a[1][0]=y0;
 double x3=x0+3.0*h;
 double k0,k1,k2,k3,k4,k5,k6;
 for(double xx=x0;xx<x3;xx+=h)
  {a[0][i+1]=xx+h;
  k1=f.func(a[0][i],a[1][i]);
  k2=f.func(a[0][i]+0.25*h,a[1][i]+0.25*k1*h);
  k3=f.func(a[0][i]+0.25*h,a[1][i]+1.0/8.0*k1*h+1.0/8.0*k2*h);
  k4=f.func(a[0][i]+0.5*h,a[1][i]-0.5*k2*h+k3*h);
  k5=f.func(a[0][i]+3.0/4.0*h,a[1][i]+3.0/16.0*k1*h+9.0/16.0*k4*h);
  k6=f.func(a[0][i]+h,a[1][i]-3.0/7.0*k1*h+2.0/7.0*k2*h+12.0/7.0*k3*h-12.0/7.0*k4*h+8.0/7.0*k5*h);
  a[1][i+1]=a[1][i]+(7.0*k1+32.0*k3+12.0*k4+32.0*k5+7*k6)/90.0*h;
  i++;
  }
  for(double xx=x3;xx<xn;xx+=h)
  {a[0][i+1]=xx+h;
  //Adams-Bashford
  k1=f.func(a[0][i],a[1][i]);
  k2=f.func(a[0][i-1],a[1][i-1]);
  k3=f.func(a[0][i-2],a[1][i-2]);
  k4=f.func(a[0][i-3],a[1][i-3]);
  a[1][i+1]=a[1][i]+h/24.0*(55.0*k1-59.0*k2+37.0*k3-9.0*k4);
  //Adams-Moulton j 1 den büyük olabilir
  for(int j=0;j<5;j++)
  {k0=f.func(a[0][i+1],a[1][i+1]);
  a[1][i+1]=a[1][i]+h/24.0*(9.0*k0+19.0*k1-5.0*k2+k3);
  }
  i++;
  }
 return a;
```

```
 }

 public static double[][] RK6AB5AM4(f_xy f,double x0,double y0,double xn,double h)
 {
 // Adams-Bashford (5 adım formülü)  Adams-Moulton (4 adım formülü) predictor corrector method
 //with RK6 initial step
 int n=(int)((xn-x0)/h);
 double a[][]=new double[2][n+1];
 // Runge-Kutta 6
 int i=0;
 a[0][0]=x0;
 a[1][0]=y0;
 double x4=x0+4.0*h;
 double k0,k1,k2,k3,k4,k5,k6;
 for(double xx=x0;xx<x4;xx+=h)
  {a[0][i+1]=xx+h;
  k1=f.func(a[0][i],a[1][i]);
  k2=f.func(a[0][i]+0.25*h,a[1][i]+0.25*k1*h);
  k3=f.func(a[0][i]+0.25*h,a[1][i]+1.0/8.0*k1*h+1.0/8.0*k2*h);
  k4=f.func(a[0][i]+0.5*h,a[1][i]-0.5*k2*h+k3*h);
  k5=f.func(a[0][i]+3.0/4.0*h,a[1][i]+3.0/16.0*k1*h+9.0/16.0*k4*h);
  k6=f.func(a[0][i]+h,a[1][i]-3.0/7.0*k1*h+2.0/7.0*k2*h+12.0/7.0*k3*h-12.0/7.0*k4*h+8.0/7.0*k5*h);
  a[1][i+1]=a[1][i]+(7.0*k1+32.0*k3+12.0*k4+32.0*k5+7*k6)/90.0*h;
  i++;
  }
  for(double xx=x4;xx<xn;xx+=h)
  {a[0][i+1]=xx+h;
  //Adams-Bashford
  k1=f.func(a[0][i],a[1][i]);
  k2=f.func(a[0][i-1],a[1][i-1]);
  k3=f.func(a[0][i-2],a[1][i-2]);
  k4=f.func(a[0][i-3],a[1][i-3]);
  k5=f.func(a[0][i-4],a[1][i-4]);
  a[1][i+1]=a[1][i]+h/720.0*(1901.0*k1-2774.0*k2+2616.0*k3-1274.0*k4+251*k5);
  //Adams-Moulton j 1 den büyük olabilir
  for(int j=0;j<5;j++)
  {k0=f.func(a[0][i+1],a[1][i+1]);
  a[1][i+1]=a[1][i]+h/720.0*(251.0*k0+646.0*k1-264.0*k2+106.0*k3-19*k4);
  }
  i++;
  }
 return a;
 }

 public static void main(String args[]) throws IOException
 {double x0=0;
  double y0=0.0;
  double yn=10.0;
  double h=0.0625;
  fxy f2=new fxy();
  fxyKC fKC=new fxyKC();
  double a[][]=RK6AB4AM3(f2,x0,y0,yn,h);
  Text.print(Text.T(a));
  Plot p1=new Plot(a);
  p1.setPlotType(0,24);
  //p1.addSpline(a,10);
  p1.addFunction(fKC,0.0,4.0,200);
  p1.plot();
 }
}
```



(bolum8_013.jpg,Resimaltı: **Adams-Bashford 4 adım Adams-Milton 3 adım Tahmin-düzeltme Predictor-corrector formülü)**

Hata miktarlarını görebilmek için Adams-Bashford 4 adım ve Adams-Milton3 adım Tahmin-düzeltme yöntemi ile çözdüğümüzde hata miktarları aşağıda verilmiştir. Runge-Kutta 6 adım yöntemi ile hesaplanan başlangıç değerleri koyu olarak işaretlenmiştir.

Çıktı 9.5-5 Adams-Bashford 4 adım Adams-Milton 3 adım Tahmin-düzeltme (Predictor-corrector) formülü Hesaplanan değerler

X	y_hesaplanan	y_tam çözüm	hata
0	**0**	**0**	**0**
0.0625	**0.062256809**	**0.062256809**	**1.29392E-10**
0.125	**0.123076922**	**0.123076923**	**6.07481E-10**
0.1875	**0.181132074**	**0.181132075**	**1.36894E-09**
0.25	**0.235295557**	**0.235294118**	**-1.43921E-06**
0.3125	0.284699363	0.284697509	-1.8537E-06
0.375	0.328768525	0.328767123	-1.40145E-06
0.4375	0.367213479	0.367213115	-3.64545E-07
0.5	0.399999069	0.4	9.31283E-07
0.5625	0.427297499	0.427299703	2.20423E-06
0.625	0.449434941	0.449438202	3.26136E-06
0.6875	0.466839495	0.466843501	4.0062E-06
0.75	0.479995578	0.48	4.42175E-06
0.8125	0.489407221	0.489411765	4.54356E-06
0.875	0.495570788	0.495575221	4.43364E-06
0.9375	0.498956338	0.498960499	4.16066E-06
1	0.499996212	0.5	3.78762E-06
1.0625	0.499079203	0.499082569	3.36586E-06
1.125	0.496548791	0.496551724	2.93349E-06
1.1875	0.492704129	0.492706645	2.51644E-06
1.25	0.487802747	0.487804878	2.13058E-06
1.3125	0.482064213	0.482065997	1.78422E-06
1.375	0.475674195	0.475675676	1.48028E-06
1.4375	0.468788591	0.468789809	1.21822E-06
1.5	0.461537466	0.461538462	9.95355E-07
1.5625	0.454028704	0.454029512	8.0791E-07
1.625	0.44635128	0.446351931	6.51661E-07
1.6875	0.438578158	0.43857868	5.22374E-07
1.75	0.430768815	0.430769231	4.16056E-07
1.8125	0.422971412	0.422971741	3.29088E-07
1.875	0.415224655	0.415224913	2.58278E-07

1.9375	0.407559372	0.407559573	2.00868E-07
2	0.399999845	0.4	1.54505E-07

Hata karşılaştırmayı görebilmek için aynı problemi Adams-Bashford 5 adım ve Adams-Milton 4 adım Tahmin-düzeltme yöntemi ile çözdüğümüzde hata miktarları aşağıda verilmiştir. Runge-Kutta 6 adım yöntemi ile hesaplanan başlangıç değerleri koyu olarak işaretlenmiştir.

Adams-Bashford 5 adım Adams-Milton 4 adım Tahmin-düzeltme (Predictor-corrector) formülü Hesaplanan değerler

x	y_hesaplanan	y_tam çözüm	hata
0.00000000000000000000	**0.00000000000000000000**	**0.00000000000000000000**	**0.00000000000000000000E+00**
0.06250000000000000000	**0.06225680920912990000**	**0.06225680933852140000**	**1.29391504455345000000E-10**
0.12500000000000000000	**0.12307692246944200000**	**0.12307692307692300000**	**6.07481079328842000000E-10**
0.18750000000000000000	**0.18113207410275700000**	**0.18113207547169800000**	**1.36894109936669000000E-09**
0.25000000000000000000	**0.23529411533906500000**	**0.23529411764705900000**	**2.30799382383573000000E-09**
0.31250000000000000000	0.28469685475736000000	0.28469750889679700000	6.54139437172852000000E-07
0.37500000000000000000	0.32876594499773400000	0.32876712328767100000	1.17828993723146000000E-06
0.43750000000000000000	0.36721161302808700000	0.36721311475409800000	1.50172601132148000000E-06
0.50000000000000000000	0.39999839935601100000	0.40000000000000000000	1.60064398901927000000E-06
0.56250000000000000000	0.42729819196822300000	0.42729970326409500000	1.51129587194765000000E-06
0.62500000000000000000	0.44943690583998300000	0.44943820224719100000	1.29640720797353000000E-06
0.68750000000000000000	0.46684247928276000000	0.46684350132626000000	1.02204349994306000000E-06
0.75000000000000000000	0.47999925820049200000	0.48000000000000000000	7.41799507986141000000E-07
0.81250000000000000000	0.48941127391795700000	0.48941176470588200000	4.90787925355729000000E-07
0.87500000000000000000	0.49557493472326900000	0.49557522123893800000	2.86515669023490000000E-07
0.93750000000000000000	0.49896036576082800000	0.49896049896049900000	1.33199671015216000000E-07
1.00000000000000000000	0.49999997320325300000	0.50000000000000000000	2.67967469880226000000E-08
1.06250000000000000000	0.49908260956179300000	0.49908256880733900000	-4.07544535341309000000E-08
1.12500000000000000000	0.49655180272928400000	0.49655172413793100000	-7.85913529810500000000E-08
1.18750000000000000000	0.49270674027182300000	0.49270664505672600000	-9.52150969180110000000E-08
1.25000000000000000000	0.48780497575024000000	0.48780487804878000000	-9.77014595338055000000E-08
1.31250000000000000000	0.48206608863152900000	0.48206599713056000000	-9.15009694679725000000E-08
1.37500000000000000000	0.47567575623482700000	0.47567567567567600000	-8.05595152942350000000E-08
1.43750000000000000000	0.46878987649136200000	0.46878980891719700000	-6.75741645417283000000E-08
1.50000000000000000000	0.46153851582225100000	0.46153846153846200000	-5.42837894634829000000E-08
1.56250000000000000000	0.45402955364521000000	0.45402951191827500000	-4.17269353514271000000E-08
1.62500000000000000000	0.44635196178861500000	0.44635193133047200000	-3.04581428967055000000E-08
1.68750000000000000000	0.43857870091491100000	0.43857868020304600000	-2.07118653339222000000E-08
1.75000000000000000000	0.43076924329127700000	0.43076923076923100000	-1.25220462332543000000E-08
1.81250000000000000000	0.42297174691765600000	0.42297174111212400000	-5.80553199958089000000E-09
1.87500000000000000000	0.41522491391281700000	0.41522491349481000000	-4.18007350955207000000E-10
1.93750000000000000000	0.40755956890978500000	0.40755957271980300000	3.81001780125345000000E-09
2.00000000000000000000	0.39999999294883700000	0.40000000000000000000	7.05116304233755000000E-09

9.6 Değişken adım boyutlu yöntemler

Hata azaltmanın bir yolu diferansiyel denklem adım boyutunu değiştirebilmektir. Bunun için önce iki değişik hata seviyesi verecek denkleme ihtiyacımız vardır. Örneğin 4. derece ve 5. derece Runge-KuttaFehlberg denklemlerini kullanabiliriz. Bu denklem setinin avantajı aynı k setlerini kullanmasıdır.

$$y_{i+1}= y_i +(\frac{16}{135}k_1+\frac{6656}{12825}k_3+\frac{28561}{56430}k_4-\frac{9}{50}k_5+\frac{2}{55}k_6)h$$ (5. derece Runge-Kutta- Fehlberg)

$$y_{i+1}= y_i +(\frac{25}{216}k_1+\frac{1408}{2565}k_3+\frac{2197}{4104}k_4-\frac{1}{5}k_5)h$$ (4. derece Runge-Kutta- Fehlberg)

$k_1=f(x_i,y_i)$

$k_2=f(x_i+0.25h\ ,\ y_i+0.25k_1h)$

$k_3=f(x_i+(3/8)h\ ,\ y_i+(3/32)k_1h+(9/32)k_2h)$

$k_4=f(x_i+(12/13)h\ ,\ y_i+(1932/2197)k_1h-(7200/2197)k_2h+(7296/2197)k_3h)$

$k_5=f(x_i+h\ ,\ y_i-(439/216)k_1h-8k_2h-(3680/513)k_3h-(845/4104)k_4h)$

$k_6=f(x_i+(1/2)h\ ,\ y_i-(8/27)k_1h+2k_2h-(3444/2565)k_3h+(1859/4104)k_4h-(11/40)k_5h)$

Seti Buthcher tablosu olarak gösterirsek

0	0	0	0	0	0	0
$1/4$	$1/4$	0	0	0	0	0
$3/8$	$3/32$	$9/32$	0	0	0	0
$12/13$	$1932/2197$	$-7200/2197$	$7296/2197$	0	0	0
1	$439/2197$	-8	$3680/513$	$-845/4104$	0	0
$1/2$	$-8/27$	2	$-3544/2565$	$1859/4104$	$-11/40$	0
	$25/216$	0	$1408/2565$	$2197/4104$	$-1/5$	0
	$16/135$	0	$6656/12825$	$28561/56430$	$-9/50$	$2/55$

$$h_{yeni}=h_{mevcut}\left|\frac{\Delta_{yeni}}{\Delta_{mevcut}}\right|^{\alpha}$$

$$\Delta_{yeni}=\varepsilon\, y_{ölçek}$$

ε genel bir tolerans seviyesidir. $y_{ölçek}$ hatanın nasıl ölçekleneceğini belirtir. Örneğin $y_{ölçek}$ = y alınabilir. Biz modelimizi hazırlarken

$y_{ölçek}=|y|+\left|y\frac{dy}{dx}\right|$ alacağız. α $(\Delta_{mevcut}\le\Delta_{yeni})$ ise değeri 0.2; $(\Delta_{mevcut}\ge\Delta_{yeni})$ ise değeri 0.25 olarak alınabilir.

Bu yaklaşımı Runge-Kutta_Fehlberg setinin yanı sıra aynı şekilde aynı katsayılar matrisiyle birden fazla denklem tanımlayan diğer setlerle de kullanabiliriz.

İkinci bir örnek set verelim:

Runge-Kutta-Cash-Karp method(RKCK)

Runge-Kutta-Cash-Karp metodu da , Runge-Kutta-Fehlberg metodunda olduğu gibi ikili diferansiyel denklem setine sahiptir. Bu setin Buther tablosu:

0	0	0	0	0	0	0
$1/5$	$1/5$	0	0	0	0	0
$3/10$	$3/40$	$9/40$	0	0	0	0
$3/5$	$3/10$	$-9/10$	$6/5$	0	0	0
1	$-11/54$	$5/2$	$-70/27$	$35/27$	0	0
$7/8$	$1631/55296$	$175/512$	$575/13824$	$44275/110592$	$253/4096$	0
	$37/378$	0	$250/621$	$125/594$	0	$512/1771$
	$2825/27648$	0	$18575/48384$	$13525/55296$	$277/14336$	$1/4$

Yine benzer olarak kullanılabilecek diğer bir sette Runge-Kutta Dormant-Prince setidir. Bu setin Buthcher tablosu altta verilmiştir.

Runge-Kutta Dormant-Prince 4-5 derece

```
Buthcher tablosu n = 8
   0.00000000 |
   0.00000000 |   0.00000000
   0.20000000 |   0.00000000   0.20000000
   0.30000000 |   0.00000000   0.07500000   0.22500000
   0.80000000 |   0.00000000   0.97777778  -3.73333333   3.55555556
   0.88888889 |   0.00000000   2.95259869 -11.59579332   9.82289285  -0.29080933
```

```
   1.00000000 |    0.00000000    2.84627525   -10.75757576    8.90642272    0.27840909   -0.27353130
   1.00000000 |    0.00000000    0.09114583    0.00000000    0.44923630    0.65104167   -0.32237618    0.13095238
-------------------------------------------------------------------------------------------------------------------------------------
              0.00000000    0.08991319    0.00000000    0.45348907    0.61406250   -0.27151238    0.08904762    0.02500000
              0.00000000    0.09114583    0.00000000    0.44923630    0.65104167   -0.32237618    0.13095238    0.00000000
```

örnek problem olarak

$\frac{dy}{dx} = -0.6y + 10e^{-(x-2)^2/[2(0.075)^2]}$ diferansiyel denkleminin çözümünü hesaplayalım. Fonksiyonda x=2 civarında bir atlama oluşmaktadır. Bu yüzden sabit step boyutu ile çözümü oldukça zordur. Program 9.6-1 de bu problemin çözümü değişken step boyutu kullanılarak gerçekleştirilmiştir. Programımızda Runge Kutta Fehlberg denklem setinin yanı sıra, Runge-Kutta Cash-Karp metodu ve Runge-Kutta Dormant-Price metodu da verilmiştir.

Program 9.6-1 Değişken h kullanan dördüncü ve beşinci dereceden Runge-Kutta_Fehlberg denklemi

```
 //=====================================================
// Numerical Analysis package in java
// example to show differential equation solution
// Runge-Kutta-Fehlberg, Cash-Karp, Dormant-Price, Verner
//  variable step methods
// Dr. Turhan Coban
// =====================================================
// dy/dx=f(x,y)

import java.io.*;
import java.util.*;

abstract class f_xy
{
 //single function single independent variable
 // example f=x*x
 abstract double func(double x,double y);
}

class fxy2 extends f_xy
{
 double func(double x,double y)
 {return -0.6*y+10.0*Math.exp(-(x-2)*(x-2)/(2*0.075*0.075));}
}

class NA92H1
{

public static double[][] RKCK45(f_xy f,double x0,double y0,double xn,double h,double eps)
{
  double hmax=(xn-x0)/40;
  double hmin=hmax/100;
// Runge-Kutta Cash-Karp order 4-5
//eps bağıl hata
int s=7;
double dydx=1.0;
double c[]=new double[s];
double a[][]=new double[s][s];
double bh[]=new double[s];
double b[]=new double[s];
c[2] =  0.2;
c[3] =  0.3;
c[4] =  0.6;
c[5] =  1;
c[6] = 7.0/8.0;
```

```
a[1][1] = 0;

a[2][1] =  0.2;

a[3][1] =  3.0/40.0;
a[3][2] =  9.0/40.0;

a[4][1] =  0.3;
a[4][2] =  -9.0/10.0;
a[4][3] =  6.0/5.0;

a[5][1] =  -11.0/54.0;
a[5][2] =  0.25;
a[5][3] =  -70.0/27.0;
a[5][4] =  35.0/27.0;

b[1] =  37.0/378.0;
b[2] =  250.0/621.0;
b[3] =  125.0/591.0;
b[4] =  512.0/1771.0;
b[5] =  0.0;

bh[1] =  2825.0/27648.0;
bh[2] =  18575.0/48384.0;
bh[3] =  13525.0/55296.0;
bh[4] =  277.0/14336.0;
bh[5] =  1.0/4.0;

 double alfa;
 //double a[][]=new double[2][n+1];
 int i=0;
 double delta=h;
 double deltayeni=eps*y0;
 double k1,k2,k3,k4,k5,k6;
 Vector<Double> xi=new Vector<Double>();
 Vector<Double> yi=new Vector<Double>();
 double x,y,yp1_5,yp1_6,xeski,yeski;
 x=x0;
 y=y0;
 xeski=x;
 yeski=y;
 Double Xi=new Double(x);
 Double Yi=new Double(y);
 xi.add(Xi);
 yi.add(Yi);
 while((x+h)<=xn)
  {
  xeski=x;
  yeski=y;
  k1=f.func(x,y);
  k2=f.func(x+c[2]*h,y+a[2][1]*k1*h);
  k3=f.func(x+c[3]*h,y+a[3][1]*k1*h+a[3][2]*k2*h);
  k4=f.func(x+c[4]*h,y+a[4][1]*k1*h+a[4][2]*k2*h+a[4][3]*k3*h);
  k5=f.func(x+c[5]*h,y+a[5][1]*k1*h+a[5][2]*k2*h+a[5][3]*k3*h+a[5][4]*k4*h);
  k6=f.func(x+c[6]*h,y+a[6][1]*k1*h+a[6][2]*k2*h+a[6][3]*k3*h+a[6][4]*k4*h+a[6][5]*k5*h);
  yp1_5=y+(b[1]*k1+b[2]*k2+b[3]*k3+b[4]*k4+b[5]*k5+b[6]*k6)*h;
  yp1_6=y+(bh[1]*k1+bh[2]*k2+bh[3]*k3+bh[4]*k4+bh[5]*k5+bh[6]*k6)*h;
  y=yp1_6;
  x=x+h;
  Xi=new Double(x);
  Yi=new Double(y);
```

```
  xi.add(Xi);
  yi.add(Yi);
  delta=Math.abs(yp1_6-yp1_5);
  dydx=f.func(x,y);
  deltayeni=Math.abs(eps*h*dydx);
  if(delta<=deltayeni) alfa=0.2;
  else                 alfa=0.25;
  h=h*0.9437436*Math.pow((deltayeni/delta),alfa);
  if(h>hmax) h=hmax;
  if(h<hmin) h=hmin;
  i++;
  }
  if(x<xn)//son bir step daha
  {
  h=xn-x;
  k1=f.func(x,y);
  k2=f.func(x+c[2]*h,y+a[2][1]*k1*h);
  k3=f.func(x+c[3]*h,y+a[3][1]*k1*h+a[3][2]*k2*h);
  k4=f.func(x+c[4]*h,y+a[4][1]*k1*h+a[4][2]*k2*h+a[4][3]*k3*h);
  k5=f.func(x+c[5]*h,y+a[5][1]*k1*h+a[5][2]*k2*h+a[5][3]*k3*h+a[5][4]*k4*h);
  k6=f.func(x+c[6]*h,y+a[6][1]*k1*h+a[6][2]*k2*h+a[6][3]*k3*h+a[6][4]*k4*h+a[6][5]*k5*h);
  yp1_5=y+(b[1]*k1+b[2]*k2+b[3]*k3+b[4]*k4+b[5]*k5+b[6]*k6)*h;
  yp1_6=y+(bh[1]*k1+bh[2]*k2+bh[3]*k3+bh[4]*k4+bh[5]*k5+bh[6]*k6)*h;
  y=yp1_6;
  //System.out.println("x="+x+"h="+h+"y="+yp1_6);
  x=xn;
  Xi=new Double(x);
  Yi=new Double(y);
  xi.add(Xi);
  yi.add(Yi);
  i++;
  }
  //vektöre aktarılan değerleri double matrise aktar
  int n=i;
     double aa[][]=new double[2][n+1];
     Enumeration nx=xi.elements();
     Enumeration ny=yi.elements();
     for(i=0;i<=n;i++)
     {
               //String ss1=(String)nx.nextElement();
               //String ss2=(String)ny.nextElement();
               Double ax=(Double)nx.nextElement();
               Double ay=(Double)ny.nextElement();
     aa[0][i]=ax.doubleValue();
     aa[1][i]=ay.doubleValue();
     }
  return aa;
 }


public static double[][] RKDP45(f_xy f,double x0,double y0,double xn,double h,double eps)
{
  double hmax=(xn-x0)/40;
  double hmin=hmax/100;
// Runge-Kutta Dormant-Price order 4-5
// eps relative error
int s=8;
double dydx=1.0;
double c[]=new double[s];
double a[][]=new double[s][s];
double bh[]=new double[s];
double b[]=new double[s];
```

```
c[1]=0.0e0;
    c[2]=0.2;
    c[3]=0.3;
    c[4]=0.8;
    c[5]=8.0/9.0;
    c[6]=1;
    c[7]=1;

    a[2][1]=0.2;
    a[3][1]=3.0/40.0;
    a[3][2]=9.0/40.0;
    a[4][1]=44.0/45.0;
    a[4][2]=-56.0/15.0;
    a[4][3]=32.0/9.0;
    a[5][1]=19372.0/6561.0;
    a[5][2]=-25360.0/2187.0;
    a[5][3]=64448.0/6561.0;
    a[5][4]=-212.0/729.0;
    a[6][1]=9017.0/3168.0;
    a[6][2]=-355.0/33.0;
    a[6][3]=46732.0/5247.0;
    a[6][4]=49.0/176.0;
    a[6][5]=-5103.0/18656.0;
    a[7][1]=35.0/384.0;
    a[7][3]=500.0/1113.0;
    a[7][4]=125.0/192.0;
    a[7][5]=-2187.0/6784.0;
    a[7][6]=11.0/84.0;

    b[1]=5179.0/57600.0;
    b[2]=0;
    b[3]=-7571.0/16695.0;
    b[4]=393.0/640.0;
    b[5]=-92097.0/339200.0;
    b[6]=187.0/2100.0;
    b[7]=1.0/40.0;

    bh[1]=35.0/384.0;
    bh[2]=0;
    bh[3]=500.0/1113.0;
    bh[4]=125.0/192.0;
    bh[5]=-2187.0/6784.0;
    bh[6]=11.0/84.0;
    bh[7]=0.0;

  double alfa;
  //double a[][]=new double[2][n+1];
  int i=0;
  //a[0][0]=x0;
  //a[1][0]=y0;
  double delta=h;
  double deltayeni=eps*y0;
  double k1,k2,k3,k4,k5,k6,k7;
  Vector<Double> xi=new Vector<Double>();
  Vector<Double> yi=new Vector<Double>();
  double x,y,yp1_5,yp1_6,xeski,yeski;
  x=x0;
  y=y0;
  xeski=x;
  yeski=y;
  Double Xi=new Double(x);
  Double Yi=new Double(y);
```

```
xi.add(Xi);
yi.add(Yi);
while((x+h)<=xn)
 {
 xeski=x;
 yeski=y;
 k1=f.func(x,y);
 k2=f.func(x+c[2]*h,y+a[2][1]*k1*h);
 k3=f.func(x+c[3]*h,y+a[3][1]*k1*h+a[3][2]*k2*h);
 k4=f.func(x+c[4]*h,y+a[4][1]*k1*h+a[4][2]*k2*h+a[4][3]*k3*h);
 k5=f.func(x+c[5]*h,y+a[5][1]*k1*h+a[5][2]*k2*h+a[5][3]*k3*h+a[5][4]*k4*h);
 k6=f.func(x+c[6]*h,y+a[6][1]*k1*h+a[6][2]*k2*h+a[6][3]*k3*h+a[6][4]*k4*h+a[6][5]*k5*h);
 k7=f.func(x+c[7]*h,y+a[7][1]*k1*h+a[7][2]*k2*h+a[7][3]*k3*h+a[7][4]*k4*h+a[7][5]*k5*h+a[7][6]*k6*h);
 yp1_5=y+(b[1]*k1+b[2]*k2+b[3]*k3+b[4]*k4+b[5]*k5+b[6]*k6+b[7]*k7)*h;
 yp1_6=y+(bh[1]*k1+bh[2]*k2+bh[3]*k3+bh[4]*k4+bh[5]*k5+bh[6]*k6+bh[7]*k7)*h;
 y=yp1_6;
 x=x+h;
 Xi=new Double(x);
 Yi=new Double(y);
 xi.add(Xi);
 yi.add(Yi);
 delta=Math.abs(yp1_6-yp1_5);
 dydx=f.func(x,y);
 deltayeni=Math.abs(eps*h*dydx);
 if(delta<=deltayeni) alfa=0.2;
 else                 alfa=0.25;
 h=h*0.9437436*Math.pow((deltayeni/delta),alfa);
 if(h>hmax) h=hmax;
 if(h<hmin) h=hmin;
 i++;
 }
 if(x<xn)//son bir step daha
 {
 h=xn-x;
 xeski=x;
 yeski=y;
 k1=f.func(x,y);
 k2=f.func(x+c[2]*h,y+a[2][1]*k1*h);
 k3=f.func(x+c[3]*h,y+a[3][1]*k1*h+a[3][2]*k2*h);
 k4=f.func(x+c[4]*h,y+a[4][1]*k1*h+a[4][2]*k2*h+a[4][3]*k3*h);
 k5=f.func(x+c[5]*h,y+a[5][1]*k1*h+a[5][2]*k2*h+a[5][3]*k3*h+a[5][4]*k4*h);
 k6=f.func(x+c[6]*h,y+a[6][1]*k1*h+a[6][2]*k2*h+a[6][3]*k3*h+a[6][4]*k4*h+a[6][5]*k5*h);
 k7=f.func(x+c[7]*h,y+a[7][1]*k1*h+a[7][2]*k2*h+a[7][3]*k3*h+a[7][4]*k4*h+a[7][5]*k5*h+a[7][6]*k6*h);
 yp1_5=y+(b[1]*k1+b[2]*k2+b[3]*k3+b[4]*k4+b[5]*k5+b[6]*k6+b[7]*k7)*h;
 yp1_6=y+(bh[1]*k1+bh[2]*k2+bh[3]*k3+bh[4]*k4+bh[5]*k5+bh[6]*k6+bh[7]*k7)*h;
 y=yp1_6;
 //System.out.println("x="+x+"h="+h+"y="+yp1_6);
 x=xn;
 Xi=new Double(x);
 Yi=new Double(y);
 xi.add(Xi);
 yi.add(Yi);
 i++;
 }
 //vektöre aktarılan değerleri double matrise aktar
 int n=i;
    double aa[][]=new double[2][n+1];
    Enumeration nx=xi.elements();
    Enumeration ny=yi.elements();
    for(i=0;i<=n;i++)
    {
                //String ss1=(String)nx.nextElement();
```

```
                //String ss2=(String)ny.nextElement();
                Double ax=(Double)nx.nextElement();
                Double ay=(Double)ny.nextElement();
    aa[0][i]=ax.doubleValue();
    aa[1][i]=ay.doubleValue();
    }
 return aa;
}

public static double[][] RKF45(f_xy f,double x0,double y0,double xn,double h,double eps)
{
 //  Runge-Kutta Fehlberg order 4-5
 //  eps relative error
 double hmax=(xn-x0)/40;
 double hmin=hmax/1000;
double sh=1;
double dydx=1.0;
double alfa;
int i=0;
double delta=h;
double deltayeni=eps*y0;
double k1,k2,k3,k4,k5,k6;
Vector<Double> xi=new Vector<Double>();
Vector<Double> yi=new Vector<Double>();
double x,y,yp1_5,yp1_6,xeski,yeski;
x=x0;
y=y0;
xeski=x;
yeski=y;
Double Xi=new Double(x);
Double Yi=new Double(y);
xi.add(Xi);
yi.add(Yi);
while((x+h)<=xn)
 {
 xeski=x;
 yeski=y;
 k1=f.func(x,y);
 k2=f.func(x+1.0/4.0*h,y+1.0/4.0*k1*h);
 k3=f.func(x+3.0/8.0*h,y+3.0/32.0*k1*h+9.0/32.0*k2*h);
 k4=f.func(x+12/13*h,y+1932.0/2197.0*k1*h-7200.0/2197.0*k2*h+7296/2197*k3*h);
 k5=f.func(x+h,y+439.0/216.0*k1*h-8.0*k2*h+3680.0/513.0*k3*h-845.0/4104.0*k4*h);
 k6=f.func(x+1.0/2.0*h,y-8.0/27.0*k1*h+2.0*k2*h+-3544.0/2565.0*k3*h+1859.0/4104.0*k4*h-11.0/40.0*k5*h);

 yp1_5=y+(25.0/216.0*k1+1408.0/2565.0*k3+2197.0/4104.0*k4-1.0/5.0*k6)*h;
 yp1_6=y+(16.0/135.0*k1+6656.0/12825.0*k3+28561.0/56430.0*k4-9.0/50.0*k5+2.0/55.0*k6)*h;
 y=yp1_6;
 x=x+h;
 Xi=new Double(x);
 Yi=new Double(y);
 xi.add(Xi);
 yi.add(Yi);
 delta=Math.abs(yp1_6-yp1_5);
 dydx=f.func(x,y);
 deltayeni=Math.abs(eps*h*dydx);
 if(delta<=deltayeni) alfa=0.25;
 else                 alfa=1.0/6.0;
 h=h*0.98*Math.pow((deltayeni/delta),alfa);
 if(h>hmax) h=hmax;
 if(h<hmin) h=hmin;
 i++;
 }
```

```
  if(x<xn)//son bir step daha
  {
  h=xn-x;
  k1=f.func(x,y);
  k2=f.func(x+1.0/4.0*h,y+1.0/4.0*k1*h);
  k3=f.func(x+3.0/8.0*h,y+3.0/32.0*k1*h+9.0/32.0*k2*h);
  k4=f.func(x+12/13*h,y+1932.0/2197.0*k1*h-7200.0/2197.0*k2*h+7296/2197*k3*h);
  k5=f.func(x+h,y+439.0/216.0*k1*h-8.0*k2*h+3680.0/513.0*k3*h-845.0/4104.0*k4*h);
  k6=f.func(x+1.0/2.0*h,y-8.0/27.0*k1*h+2.0*k2*h+-3544.0/2565.0*k3*h+1859.0/4104.0*k4*h-11.0/40.0*k5*h);

  yp1_5=y+(25.0/216.0*k1+1408.0/2565.0*k3+2197.0/4104.0*k4-1.0/5.0*k6)*h;
  yp1_6=y+(16.0/135.0*k1+6656.0/12825.0*k3+28561.0/56430.0*k4-9.0/50.0*k5+2.0/55.0*k6)*h;
  y=yp1_6;
  //System.out.println("x="+x+"h="+h+"y="+yp1_6);
  x=xn;
  Xi=new Double(x);
  Yi=new Double(y);
  xi.add(Xi);
  yi.add(Yi);
  i++;
  }
  //vektöre aktarılan değerleri double matrise aktar
  int n=i;
     double aa[][]=new double[2][n+1];
     Enumeration nx=xi.elements();
     Enumeration ny=yi.elements();
     for(i=0;i<=n;i++)
     {
               //String ss1=(String)nx.nextElement();
               //String ss2=(String)ny.nextElement();
               Double ax=(Double)nx.nextElement();
               Double ay=(Double)ny.nextElement();
     aa[0][i]=ax.doubleValue();
     aa[1][i]=ay.doubleValue();
     }
  return aa;
 }

public static double[][] RKV16(f_xy f,double x0,double y0,double xn,double h,double eps)
{  //Runge-Kutta-Verner 16th degree step constant step size version
double hmax=(xn-x0)/40;
double hmin=hmax/100;
double dydx=1.0;
double delta,deltayeni,alfa;
int s=17;
double c[]=new double[s];
double a[][]=new double[s][s];
double bh[]=new double[s];
double b[]=new double[s];
c[0]=0;
c[1] =  0.;
c[2] =  .6e-1;
c[3] =  .95933333333333333333333333333333333333333e-1;
c[4] =  .1439;
c[5] =  .4973;
c[6] =  .9725;
c[7] =  .9995;
c[8] =  1.;
c[9] =  1.;
a[2][1] =  .6e-1;
a[3][1] =  .19239962962962962962962962962962962962e-1;
a[3][2] =  .76693370370370370370370370370370370370e-1;
```

```
a[4][1] =  .35975e-1;
a[4][2] =  0.;
a[4][3] =  .107925;
a[5][1] =  1.3186834152331482609197472764317356128 61;
a[5][2] =  0.;
a[5][3] = -5.0420580636285622254277616347156376933 44;
a[5][4] =  4.2206746483954139645080143582839020804 83;
a[6][1] = -41.8725916643275146180375778064434681290 5;
a[6][2] =  0.;
a[6][3] =  159.4325621631374917700365669070346830453;
a[6][4] = -122.1192135650100309202516203389242140663;
a[6][5] =  5.5317430662000537682526312383329991500 76;
a[7][1] = -54.4301569353165043325064205129414246127 1;
a[7][2] =  0.;
a[7][3] =  207.0672513650184644273657173866509835987;
a[7][4] = -158.6108137845899991828742424365058599469;
a[7][5] =  6.9918165859502423219925972807917939070 96;
a[7][6] = -.1859723106220323397765171799549294623692e-1;

a[8][1] = -54.6637417872819768024121564805038695935 1;
a[8][2] =  0.;
a[8][3] =  207.9528062553893734515824816699834244238;
a[8][4] = -159.2889574744995071508959805871426654216;
a[8][5] =  7.0187437407969444346981707609642524908 17;
a[8][6] = -.1833878590504572306472782005141738268361e-1;
a[8][7] = -.5119484997882099077875432497245168395840e-3;

a[9][1] =  .3438957868357036009278820124728322386520e-1;
a[9][2] =  0.;
a[9][3] =  0.;
a[9][4] =  .2582624555633503404659558098586120858767;
a[9][5] =  .4209371189673537150642551514069801967032;
a[9][6] =  4.4053964696693101701488368161970956648 91;
a[9][7] = -176.4831190242986576151740942499002125029;
a[9][8] =  172.3641334014150730294022582711902413315;

b[1] =  .3438957868357036009278820124728322386520e-1;
b[2] =  0.;
b[3] =  0.;
b[4] =  .2582624555633503404659558098586120858767;
b[5] =  .4209371189673537150642551514069801967032;
b[6] =  4.4053964696693101701488368161970956648 91;
b[7] = -176.4831190242986576151740942499002125029;
b[8] =  172.3641334014150730294022582711902413315;
b[9] =  0.;

bh[ 1] =  .4909967648382489730906854927971225836479e-1;
bh[ 2] =  0.;
bh[ 3] =  0.;
bh[ 4] =  .2251112229516524153401395320539875329485;
bh[ 5] =  .4694682253029562039431948525047387412553;
bh[ 6] =  .8065792249988867707634161808995217981443;
bh[ 7] =  0.;
bh[ 8] = -.6071194891777959797672951465256217122488;
bh[ 9] =  .5686113944047569241147603178766138153594e-1;
 // Runge-Kutta Verner 16 variable step
 //eps relative error
 //double a[][]=new double[2][n+1];
 int i=0;
 double x1,y1;
 Vector<Double> xi=new Vector<Double>();
 Vector<Double> yi=new Vector<Double>();
```

```
double x,y,yp1=0,yp2=0,xeski,yeski;

x=x0;
y=y0;
xeski=x;
yeski=y;
Double Xi=new Double(x);
Double Yi=new Double(y);
double k[]=new double[s];
xi.add(Xi);
yi.add(Yi);
int n=1;
while((x+h)<=xn)
{
 xeski=x;
 yeski=y;
 double total=0;
 //yp1=y;
 yp2=y;
 double w=1;
 for(int ii=1;ii<s;ii++)
 {  x1=x+c[ii]*h;
   y1=y;
              for(int j=1;j<ii;j++)
   {y1+=h*a[ii][j]*k[j];}
    k[ii]=f.func(x1,y1);
    yp2+=h*bh[ii]*k[ii];
    yp1+=h*b[ii]*k[ii];
 }
 delta=Math.abs(yp2-yp1);
 dydx=f.func(x,y);
 deltayeni=Math.abs(eps*h*dydx);
 if(delta<=deltayeni) alfa=0.2;
 else              alfa=0.25;
 h=h*0.9437436*Math.pow((deltayeni/delta),alfa);
 if(h>hmax) h=hmax;
 if(h<hmin) h=hmin;
 y=yp2;
 x+=h;
 Xi=new Double(x);
 Yi=new Double(y);
 xi.add(Xi);
 yi.add(Yi);
 n++;
 } //end of while
 if(x<xn)//son bir step daha
 {
 h=xn-x;
 xeski=x;
 yeski=y;
 double total=0;
 yp1=y;
 yp2=y;
 for(int ii=1;ii<s;ii++)
 {  x1=x+c[ii]*h;
   y1=y;
              for(int j=1;j<ii;j++)
   {y1+=h*a[i][j]*k[j];}
   k[ii]=f.func(x1,y1);
   yp1+=h*b[ii]*k[ii];
   yp2+=h*bh[ii]*k[ii];
 }
```

```
  y=yp2;
  x+=h;
  Xi=new Double(x);
  Yi=new Double(y);
  xi.add(Xi);
  yi.add(Yi);
  }
     // vektöre aktarilan değerleri double matrise aktar
     double aa[][]=new double[2][n+1];
     Enumeration nx=xi.elements();
     Enumeration ny=yi.elements();
     for(i=0;i<=n;i++)
     {
               Double ax=(Double)nx.nextElement();
               Double ay=(Double)ny.nextElement();
     aa[0][i]=ax.doubleValue();
     aa[1][i]=ay.doubleValue();
     }
  return aa;
}

public static double[][] RKV13(f_xy f,double x0,double y0,double xn,double h,double eps)
{  //Runge-Kutta-Verner 13th degree step constant step size version
  double hmax=(xn-x0)/40;
  double hmin=hmax/1000;
  double dydx=1.0;
  double delta,deltayeni,alfa;
int s=14;
double c[]=new double[s];
double a[][]=new double[s][s];
double bh[]=new double[s];
double b[]=new double[s];
c[0]=0;
c[1] =  0.;
c[2] =  .5e-1;
a[2][1] =  .5e-1;
c[3] =  .1065625;
a[3][1] = -.69931640625e-2;
a[3][2] =  .1135556640625;
c[4] =  .15984375;
a[4][1] =  .399609375e-1;
a[4][2] =  0.;
a[4][3] =  .1198828125;
c[5] =  .39;
a[5][1] =  .36139756280045751240529407211840283451 29;
a[5][2] =  0.;
a[5][3] = -1.34152406670049277181998778820271583 4917;
a[5][4] =  1.37012650390003525941469371608431300 0404;
c[6] =  .465;
a[6][1] =  .49047202797202797202797202797202797202 80e-1;
a[6][2] =  0.;
a[6][3] =  0.;
a[6][4] =  .23509720422144047398629883354934271431 22;
a[6][5] =  .18085559298135672881090396365345448847 50;
//
c[7] =  .155;
a[7][1] =  .61692890442890442890442890442890442890 44e-1;
a[7][2] =  0.;
a[7][3] =  0.;
a[7][4] =  .11236568314640276622625570351300154423 03;
a[7][5] = -.38850460714513667670490481081112445674 56e-1;
```

```
a[7][6] =  .19791887125220458553791887125220458553 79e-1;
c[8] =  .943;
a[8][1] = -1.76763024022232687573559711957214558 6714;
a[8][2] =  0.;
a[8][3] =  0.;
a[8][4] = -62.5;
a[8][5] = -6.06188937737666910082136145965933199 9758;
a[8][6] =  5.65082319822276313856129803060084017 4201;
a[8][7] =  65.62169641937623283799566054863063741227;
c[9] =  .90180204173585695825970794067837214995 60;
a[9][1] = -1.18094506655497079982511628262828297957882;
a[9][2] =  0.;
a[9][3] =  0.;
a[9][4] = -41.50473441114320841606641502701994225874;
a[9][5] = -4.43443831910372501122516922984610021 1776;
a[9][6] =  4.26040818858613302481219371074469324 0761;
a[9][7] =  43.75364022446171584987676829438379303004;
a[9][8] =  .78714254899123106874464750442263075508 60e-2;
c[10] =  .909;
a[10][1] = -1.28140599944148840545951029118205424 6266;
a[10][2] =  0.;
a[10][3] =  0.;
a[10][4] = -45.04713996013986630220754257136007322267;
a[10][5] = -4.73136206944957647731146426549128281 0943;
a[10][6] =  4.51496701659380784118585158459724099 6214;
a[10][7] =  47.44909557172985134869022392235929015114;
a[10][8] =  .10592282971116611356873939555165428752 28e-1;
a[10][9] = -.57468422638446162544323184782862962320 21e-2;
c[11] =  .94;
a[11][1] = -1.72447013426248519175670981748448186 1731;
a[11][2] =  0.;
a[11][3] =  0.;
a[11][4] = -60.92349008483054016518434619253765246063;
a[11][5] = -5.95151837622239245520283276706185486829 0;
a[11][6] =  5.55652373069845623597979165084359249683 9;
a[11][7] =  63.98301198033305336837536378635995939281;
a[11][8] =  .14642028250414961592759213917594526760 03e-1;
a[11][9] =  .64604087723582036036218651449776507148 92e-1;
a[11][10] = -.79303231690088789840244525486933732914 47e-1;
c[12] =  1.;
a[12][1] = -3.30162266774707901635399478979098362556 9;
a[12][2] =  0.;
a[12][3] =  0.;
a[12][4] = -118.01127235975250856669233039578988685 10;
a[12][5] = -10.14142238845611248642783916034510897595;
a[12][6] =  9.13931133223205792354401227355682700061 9;
a[12][7] =  123.37594282840426836848471809865018943 64;
a[12][8] =  4.62324437887458047483980762506763092479 2;
a[12][9] = -3.38327773806820192365255097153681124081 4;
a[12][10] =  4.52759210032461818945126533935112903532 5;
a[12][11] = -5.82849548581162296319308801916298570375 5;
c[13] =  1.;
a[13][1] = -3.03951503376630903004010285182120025105 6;
a[13][2] =  0.;
a[13][3] =  0.;
a[13][4] = -109.26086808941762546864441923221646233 52;
a[13][5] = -9.29064249740029344971766554265689754915 8;
a[13][6] =  8.43050498176491114213429925383616780345 4;
a[13][7] =  114.20100103783313135574240410955234274 76;
a[13][8] = -.96372713421454793581623756589879016527 62;
a[13][9] = -5.03488408880218979119868033618333232311 8;
a[13][10] =  5.95813082400292317754040216538817207279 4;
```

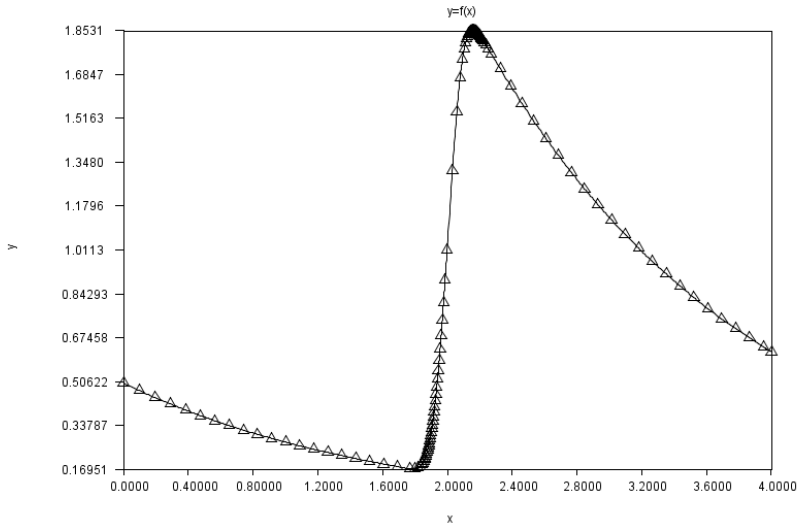
```
a[13][11] =  0.;
a[13][12] =  0.;

bh[1] =  .443126152290897921248643651020902976489 3e-1;
bh[2] =  0.;
bh[3] =  0.;
bh[4] =  0.;
bh[5] =  0.;
bh[6] =  .354609564234322644786317935089505503 8855;
bh[7] =  .247848043136665306961998672150445866 0016;
bh[8] =  4.448134732475784492725128317159648871312;
bh[9] =  19.846886366118733699309323992976879352 91;
bh[10] = -23.581623377465618419695179608703949650 85;
bh[11] =  0.;
bh[12] =  0.;
bh[13] = -.360167943728977516212453673774620240911 0;

 // Runge-Kutta Verner 16 variable step
 //eps relative error
 //double a[][]=new double[2][n+1];
 int i=0;
 double x1,y1;
 Vector<Double> xi=new Vector<Double>();
 Vector<Double> yi=new Vector<Double>();
 double x,y,yp1,yp2,xeski,yeski;
 x=x0;
 y=y0;
 xeski=x;
 yeski=y;
 Double Xi=new Double(x);
 Double Yi=new Double(y);
 double k[]=new double[s];
 xi.add(Xi);
 yi.add(Yi);
 int n=1;
 while((x+h)<=xn)
 {
  xeski=x;
  yeski=y;
  double total=0;
  yp1=y;
  yp2=y;
  double w=1;
  for(int ii=1;ii<s;ii++)
  {  x1=x+c[ii]*h;
    y1=y;
                for(int j=1;j<ii;j++)
    {y1+=h*a[ii][j]*k[j];}
     k[ii]=f.func(x1,y1);
     yp1+=h*b[ii]*k[ii];
     yp2+=h*bh[ii]*k[ii];
  }
  delta=Math.abs(yp2-yp1);
  dydx=f.func(x,y);
  deltayeni=Math.abs(eps*h*dydx);
  if(delta<=deltayeni) alfa=0.2;
  else               alfa=0.25;
  h=h*0.9437436*Math.pow((deltayeni/delta),alfa);
  if(h>hmax) h=hmax;
  if(h<hmin) h=hmin;

  y=yp2;
```

```
  x+=h;
  Xi=new Double(x);
  Yi=new Double(y);
  xi.add(Xi);
  yi.add(Yi);
  n++;
  } //end of while
  if(x<xn)//son bir step daha
  {
  h=xn-x;
  xeski=x;
  yeski=y;
  double total=0;
  yp1=y;
  yp2=y;
  for(int ii=1;ii<s;ii++)
  {  x1=x+c[ii]*h;
    y1=y;
              for(int j=1;j<ii;j++)
    {y1+=h*a[i][j]*k[j];}
    k[ii]=f.func(x1,y1);
    yp1+=h*b[ii]*k[ii];
    yp2+=h*bh[ii]*k[ii];
  }
  y=yp2;
  x+=h;
  Xi=new Double(x);
  Yi=new Double(y);
  xi.add(Xi);
  yi.add(Yi);
  }
  //vektöre aktarilan değerleri double matrise aktar
      double aa[][]=new double[2][n+1];
      Enumeration nx=xi.elements();
      Enumeration ny=yi.elements();
      for(i=0;i<=n;i++)
      {
                Double ax=(Double)nx.nextElement();
                Double ay=(Double)ny.nextElement();
      aa[0][i]=ax.doubleValue();
      aa[1][i]=ay.doubleValue();
      }
  return aa;
}

public static double[][] RKV9(f_xy f,double x0,double y0,double xn,double h,double eps)
{  //Runge-Kutta-Verner 9th degree step constant step size version
  double hmax=(xn-x0)/40.0;
  double hmin=hmax/100.0;int s=10;
  double dydx=1.0;
double c[]=new double[s];
double a[][]=new double[s][s];
double bh[]=new double[s];
double b[]=new double[s];
c[0]=0;
c[1] =  0.;
c[2] =  .6e-1;
c[3] =  .95933333333333333333333333333333333333333e-1;
c[4] =  .1439;
c[5] =  .4973;
c[6] =  .9725;
c[7] =  .9995;
```

```
c[8] =  1.;
c[9] =  1.;
a[2][1] =  .6e-1;
a[3][1] =  .192399629629629629629629629629629629629e-1;
a[3][2] =  .766933703703703703703703703703703703703e-1;
a[4][1] =  .35975e-1;
a[4][2] =  0.;
a[4][3] =  .107925;
a[5][1] =  1.31868341523314826091974727643173561286l;
a[5][2] =  0.;
a[5][3] = -5.04205806362856222542776163471563769334;
a[5][4] =  4.22067464839541396450801435828390208048;
a[6][1] = -41.8725916643275146180375778064434681290;
a[6][2] =  0.;
a[6][3] =  159.432562163137491770036566907034683045;
a[6][4] = -122.119213565010030920251620338924214066;
a[6][5] =  5.53174306620005376825263123833299915007;
a[7][1] = -54.4301569353165043325064205129414246127;
a[7][2] =  0.;
a[7][3] =  207.067251365018464427365717386650983598;
a[7][4] = -158.610813784589999182874242436505859946;
a[7][5] =  6.99181658595024232199259728079179390709;
a[7][6] = -.185972310622032339776517179954929462369e-1;

a[8][1] = -54.6637417872819768024121564805038695935;
a[8][2] =  0.;
a[8][3] =  207.952806255389373451582481669983424423;
a[8][4] = -159.288957474499507150895980587142665421;
a[8][5] =  7.01874374079694443469817076096425249081;
a[8][6] = -.183387859050457230647278200514173826836e-1;
a[8][7] = -.511948499788209907787543249724516839584e-3;

a[9][1] =  .343895786835703600927882012472832238652e-1;
a[9][2] =  0.;
a[9][3] =  0.;
a[9][4] =  .258262455563350340465955809858612085876;
a[9][5] =  .420937118967353715064255151406980196703;
a[9][6] =  4.40539646966931017014883681619709566489;
a[9][7] = -176.483119024298657615174094249900212502;
a[9][8] =  172.364133401415073029402258271190241331;

b[1] =  .343895786835703600927882012472832238652e-1;
b[2] =  0.;
b[3] =  0.;
b[4] =  .258262455563350340465955809858612085876;
b[5] =  .420937118967353715064255151406980196703;
b[6] =  4.40539646966931017014883681619709566489;
b[7] = -176.483119024298657615174094249900212502;
b[8] =  172.364133401415073029402258271190241331;
b[9] =  0.;

bh[ 1] =  .490996764838248973090685492797122583647e-1;
bh[ 2] =  0.;
bh[ 3] =  0.;
bh[ 4] =  .225111222951652415340139532053987532948;
bh[ 5] =  .469468225302956203943194852504738741255;
bh[ 6] =  .806579224998886770763416180899521798144;
bh[ 7] =  0.;
bh[ 8] = -.607119489177795979767295146525621712248;
bh[ 9] =  .568611394404756924114760317876613815359e-1;
 // Runge-Kutta Verner 16 variable step
 //eps relative error
```

```
double alfa;
//double a[][]=new double[2][n+1];
int i=0;
double x1,y1;
Vector<Double> xi=new Vector<Double>();
Vector<Double> yi=new Vector<Double>();
double x,y,yp1,yp2,xeski,yeski;
x=x0;
y=y0;
xeski=x;
yeski=y;
double delta,deltayeni;
Double Xi=new Double(x);
Double Yi=new Double(y);
double k[]=new double[s];
xi.add(Xi);
yi.add(Yi);
int n=1;
while((x+h)<=xn)
{
 xeski=x;
 yeski=y;
 double total=0;
 yp1=y;
 yp2=y;
 double w=1;
 for(int ii=1;ii<s;ii++)
 {  x1=x+c[ii]*h;
   y1=y;
              for(int j=1;j<ii;j++)
   {y1+=h*a[ii][j]*k[j];}
    k[ii]=f.func(x1,y1);
    yp2+=h*bh[ii]*k[ii];
 }
 delta=Math.abs(yp2-yp1);
 dydx=f.func(x,y);
 deltayeni=Math.abs(eps*h*dydx);
 if(delta<=deltayeni) alfa=0.2;
 else              alfa=0.25;
 h=h*0.9437436*Math.pow((deltayeni/delta),alfa);
 if(h>hmax) h=hmax;
 if(h<hmin) h=hmin;
 y=yp2;
 x+=h;
 Xi=new Double(x);
 Yi=new Double(y);
 xi.add(Xi);
 yi.add(Yi);
 n++;
 } //end of while
 if(x<xn)//son bir step daha
 {
 h=xn-x;
 xeski=x;
 yeski=y;
 double total=0;
 yp1=y;
 yp2=y;
 for(int ii=1;ii<s;ii++)
 {  x1=x+c[ii]*h;
   y1=y;
              for(int j=1;j<ii;j++)
```

```
    {y1+=h*a[i][j]*k[j];}
    k[ii]=f.func(x1,y1);
    yp1+=h*b[ii]*k[ii];
    yp2+=h*bh[ii]*k[ii];
   }
   y=yp2;
   x+=h;
   Xi=new Double(x);
   Yi=new Double(y);
   xi.add(Xi);
   yi.add(Yi);
   }
   //vektöre aktarilan değerleri double matrise aktar
      double aa[][]=new double[2][n+1];
      Enumeration nx=xi.elements();
      Enumeration ny=yi.elements();
      for(i=0;i<=n;i++)
      {
                Double ax=(Double)nx.nextElement();
                Double ay=(Double)ny.nextElement();
      aa[0][i]=ax.doubleValue();
      aa[1][i]=ay.doubleValue();
      }
   return aa;
}
  public static void main(String args[]) throws IOException
  {
   double x0=0;
   double y0=0.5;
   fxy2 f1=new fxy2();
   double a[][]=RKF45(f1,x0,y0,4.0,0.1,5e-3);
    Text.print(Text.T(a));
    Plot p1=new Plot(a);
    p1.setPlotType(0,24);
    p1.addSpline(a,8);
    //p1.addData(a);
    p1.plot();
  }
}
```



(bolum8_014.jpg,Resimaltı: **Değişken h kullanan dördüncü ve beşinci dereceden Runge-Kutta_Fehlberg**

denklemi)

9.7 Birden fazla değişkenli denklemler

$$\frac{dy_1}{dx} = f_1(x, y_1, y_2, ..., y_n)$$

$$\frac{dy_2}{dx} = f_2(x, y_1, y_2, ..., y_n)$$

....

$$\frac{dy_n}{dx} = f_n(x, y_1, y_2, ..., y_n)$$

denklem sisteminin çözümünü düşünelim. Bu tür bir sistem x'in başlangıç koşulunda verilen n tane başlangıç koşulu gerektirir. Örnek olarak

$$\frac{dy_1}{dx} = -0.5y_1$$

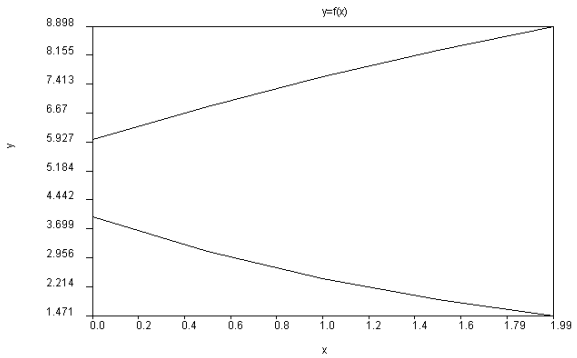
$$\frac{dy_2}{dx} = 4 - 0.3y_2 - 0.1y_1$$

setini düşünelim. x=0 için y_1=4 ve y_2=6 olduğunu varsayalım. Bu denklem sistemini 6. derece Runge kutta denklemi ile çözersek :

Program 8.7.1 iki değişkenli diferansiyel denklem setinin 6 derece Runge-Kutta metodu ile çözülmesi

```
 import java.io.*;
class fm1 extends f_xi
{
//multivariable function
 double func(double x[],int x_ref)
 {
  //x[0] is x, x[1] is y
  if(x_ref==0) return -0.5*x[1];
  if(x_ref==1) return 4-0.3*x[2]-0.1*x[1];
  else return 0;
 }}
class diferansiyel3
{
public static void main(String args[]) throws IOException
 {
```

```
 fm1 b3=new fm1();
 double y[]={4,6};
 Runge_Kutta rk=new Runge_Kutta("RK6");
 double a[][]=rk.hesapla(b3,0.0,2.0,y,4);
 Text.print(Text.T(a));
 Plot pp=new Plot(a);
 pp.plot();
}}
```



(bolum8_014.jpg,Resimaltı: **iki değişkenli diferansiyel denklem setinin 6 derece Runge-Kutta metodu ile çözülmesi grafik format çıktısı)**

iki değişkenli diferansiyel denklem setinin 6 derece Runge-Kutta metodu ile çözülmesi

0.0	4.0	6.0
0.5	3.115199788411459	6.835752376757813
1.0	2.426117430429699	7.596172963183511
1.5	1.889460126483987	8.282667565228241
2.0	1.4715114465587011	8.898456285684427

Üstteki birinci dereceden diferansiyel denklem seti çözümünü n inci derceden diferansiyel denklemleri çözmek için de uygulayabiliriz.

$$\frac{d^n y_1}{dx^n} = g(x, y, \frac{dy}{dx}, \frac{d^2 y}{dx^2}, ..., \frac{d^{n-1} y}{dx^{n-1}})$$

bu denklemde değişken değiştirirsek

$x=x_0$

$y=x_1$

$$\frac{dy}{dx} = x_2$$

$$\frac{d^2 y}{dx^2} = x_3$$

..

$$\frac{d^{n-1} y}{dx^{n-1}} = x_{n-1}$$

tanımını yaparsak, diferansiyel denklemimiz bir denklem sistemine dönüşür.

$$\frac{dx_k}{dx_0} = \frac{d^k y}{dx_0^k} = x_{k+1}\ ,\ 1 \le k < n$$

$$\frac{dx_n}{dx_0} = \frac{d^n y}{dx_0^n} = g(x_0, x_1, x_2, ..., x_n)$$

bu denklem sistemi de ilk başlangıç değerleri verildiğinde rahatlıkla çözebileceğimiz bir set oluşturur.
Örnek olarak

$$\frac{d^2 y}{dt^2} + 11\frac{dy}{dt} + 10y = 10u(t)$$

diferansiyel denklemini alalım. Bu denklemi u(t)=1,ve başlangıç sınır şartları t=0 da y(t)=0,dy(t)/dt=0 olursa çözüm seti y(t)=1-(10/9)exp(-t)+(1/9)exp(-10t) olarak bulunur. Aynı denklemi sayısal yöntemle çözmek için

$$\frac{dx_1}{dx_0} = x_2$$

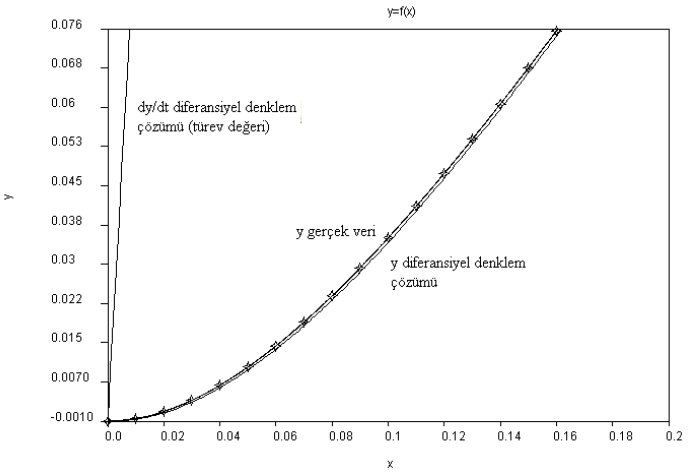
$$\frac{dx_2}{dx_0} = 10u - 10x_1 - 11x_2$$

$x_1(0)=0$
$x_2(0)=0$
olarak yazabiliriz.

Program 8.7.2 diferansiyel denklem setinin 6 derece Runge-Kutta metodu ile çözülmesi

```
import java.io.*;

class fm1 extends f_xi
{
//multivariable function
 double func(double x[],int x_ref)
 {
  //x[0] is x, x[1] is y
  if(x_ref==0) return x[2];
  if(x_ref==1) return 10-10.0*x[1]-11*x[2];
  else return 0;
 }
}

class fm2 extends f_x
{// gerçek çözüm
 public double func(double t)
 {return 1.0-10.0/9.0*Math.exp(-t)+1.0/9.0*Math.exp(-10.0*t);}
}

class diferansiyel3A
{

public static void main(String args[]) throws IOException
 {

  fm1 b1=new fm1();
  fm2 b2=new fm2();
  double y[];
  y=new double[2];
  y[0]=0;
  y[1]=0;
  Runge_Kutta rk=new Runge_Kutta("RK6");
  double a[][]=rk.hesapla(b1,0.0,0.2,y,200);
  Text.print(Text.T(a));
  Plot pp=new Plot(a);
```

```
 pp.addFunction(b2,0.0,0.2,20);
 pp.addSpline(2,100);
 pp.setPlotType(2,33);
 pp.plot();
 }
}
```



(bolum8_015.jpg,Resimaltı: **diferansiyel denklemin 6 derece Runge-Kutta metodu ile çözülmesi)**

İkinci bir örnek verelim.

Makine mühendisliği ve atmosfer bilimi (Meteoroloji)nin ilgilendiği konulardan birisi akışkanlar mekaniğidir. Akışkanlar mekaniğinin ana konularından birisi türbülanslı akış problemidir. Türbülans problemine klasik yaklaşımımızda türbülansın tesadüfi etkilerden oluştuğu kabul edilir. Türbülansın tesadüfi yapısının incelenmesi, bu yapının içinde tesadüfi olmayan bazı diferansiyel denklemlerin ufak değişimlerinin kaotik sonuçlar verebildiği görülmüştür. Bunun sonucunda kaos ayrı bir bilim dalı olarak gelişmiştir. Koas'u anlamamıza yardım eden ilk çıkış noktalarından birisi Lorentz denklemleridir. Bu denklem atmosfer bilimci Edward Lorentz tarafından geliştirilmiştir.

$$\frac{dx_1}{dt} = \sigma(x_2 - x_1)$$

$$\frac{dx_2}{dt} = (1 + \lambda - x_3)x_1 - x_2$$

$$\frac{dx_3}{dt} = x_1 x_2 - \gamma x_2$$

bu denklemlerdeki σ, λ, ve γ pozitif sabitlerdir. Denklemlerin davranışı

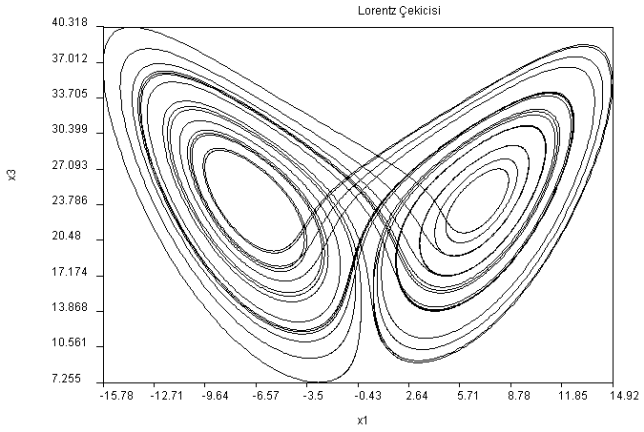
$\sigma > \gamma + 1$

$\lambda > \frac{(\sigma+1)(\sigma+\gamma+1)}{\sigma-\gamma-1}$ olduğunda kaotik davranış gösterecektir. Örneğin $\sigma = 10$, $\lambda = 24$ ve $\gamma = 2$ olduğunda denkleme kaotik davranacaktır. $0\ s \le t \le 30\ s$ arasında $x_1(0) = 1.0$; $x_2(0) = 0.0$; $x_3(0) = 20.0$ şartlarını kullanarak denklemi çözelim.

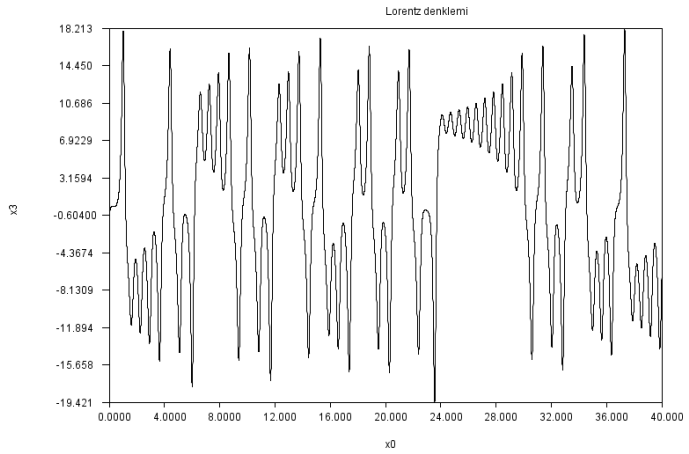
Program 8.7.3 Lorentz denklem setinin Runge-Kutta Verner 9 metodu ile çözülmesi

```
 //=====================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
```

```
// Dr. Turhan Coban
// ====================================================
import java.io.*;

class fm2 extends f_xi
{
//multivariable function
//değişik değerleri deneyiniz
 double sigma=10.0;
 double lambda=28.0;
 double gamma=8.0/3.0;
 double func(double x[],int x_ref)
 {
  //x[0] is x, x[1] is y

  if(x_ref==0) return sigma*(x[2]-x[1]);
  if(x_ref==1) return (1+lambda-x[3])*x[1]-x[2];
  else        return x[1]*x[2]-gamma*x[3];
 }
}

class diferansiyel3B
{

public static void main(String args[]) throws IOException
{
 fm2 b1=new fm2();
 double y[];
 y=new double[3];
 y[0]=-0.6;
 y[1]=0.0;
 y[2]=60.0;
  Runge_Kutta rk=new Runge_Kutta("RKV9");
 double a[][]=rk.hesapla(b1,0.0,40.0,y,10000);
 Plot p1=new Plot(a[1],a[3]);
 p1.setXlabel("x1");
 p1.setYlabel("x3");
 p1.setPlabel("Lorentz Çekicisi");
 p1.plot();
 Plot p2=new Plot(a[0],a[1]);
 p2.setXlabel("x0");
 p2.setYlabel("x3");
 p2.setPlabel("Lorentz denklemi");
 p2.plot();
 }
}
```

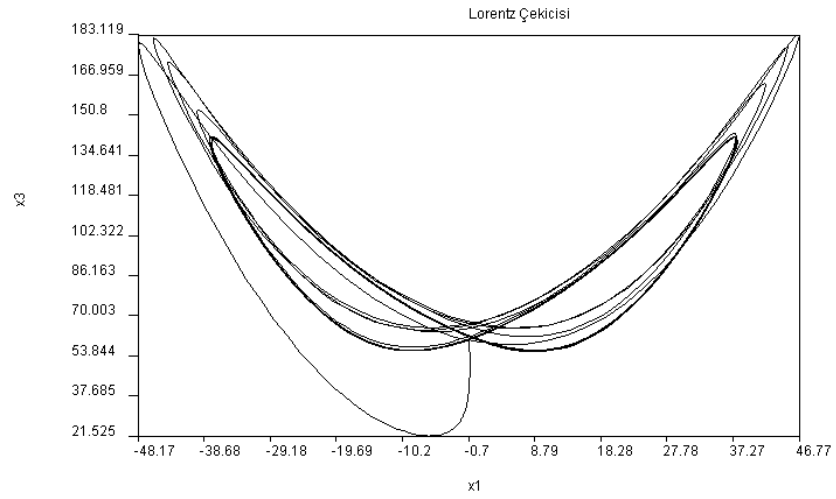


(bolum8_016.jpg,Resimaltı: **Lorentz denklem setinin 9 derece Runge-KuttaVerner metodu ile çözülmesi** $\sigma = 10$, λ=24 ve γ=2 x1-x2 seti **)**



(bolum8_017.jpg,Resimaltı: **Lorentz denklem setinin 9 derece Runge-KuttaVerner metodu ile çözülmesi** $\sigma = 10$, λ=24 ve γ=2 x1-x3 seti**)**

Parametreleri biraz değiştirirsek denklemlerin değişik bir davranış gösterdiğini gözleyebiliriz.



(bolum8_018.jpg,Resimaltı: **Lorentz denklem setinin 9 derece Runge-KuttaVerner metodu ile çözülmesi** $\sigma = 10$, λ=90 ve γ=5.5 x1-x3 seti**)**

Kaosla ilgili verebileceğimiz diğer bir ilginç örnekAv-Avcı modeli dediğimiz modeldir. Avcı sayısı çok arttığında avı aşırı tüketeceğinden av sayısı azalır ve bu da avcı sayısının da azalmasına yol açar. Av ve avcı sayısı arasındaki bu ilişkiyi İtalyan matematikçi Vito Voltera ve Amerikalı Bioloji bilgini Alfret J. Lotka şu denklemlerle ifade etmiştir :

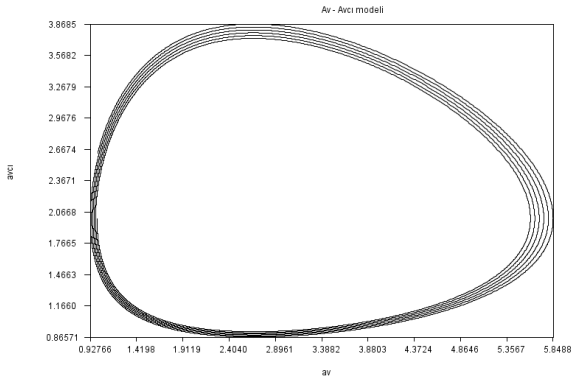
$$\frac{dy_1}{dt} = ay_1 - by_1y_2$$

$$\frac{dy_2}{dt} = -cy_2 + dy_1y_2$$

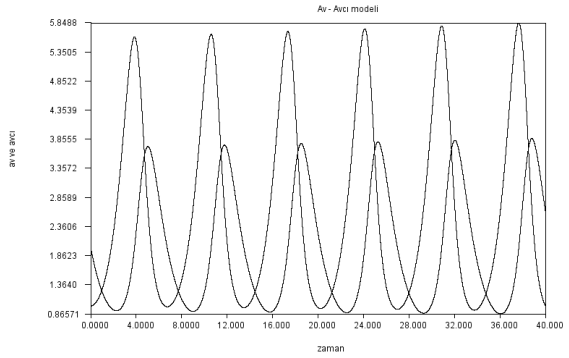
buradaki y_1 av sayısını, y_2 avcı sayısını belirtmektedir. a av doğum oranı, b avcı ölüm oranı, c av ölüm oranında av-avcı ilişkisinin etkisi ve d avcı çoğalma oranında av-avcı ilişkisinin etkisini belirtmektedir.

Program 8.7.4 Av-Avcı modeli (Lotka-Volterra denklemi) denklem setinin 9 derece Runge-KuttaVerner metodu ile çözülmesi

```
 import java.io.*;

class fm1 extends f_xi
{
//av avcı modeli
// Lotka-Volterra denklemi
 double a=1.2;  //av çoğalma oranı
 double b=0.6;  //avcı ölüm oranı
 double c=0.8;  //avın ölümündeki av avcı ilişkisinin etkisi
 double d=0.3;  //avcının yetişmesindeki av avcı ilişkisinin etkisi

 double func(double x[],int x_ref)
 {
  //x[0] is x, x[1] is y
  if(x_ref==0) return a*x[1]-b*x[1]*x[2];
  else return -c*x[2]+d*x[1]*x[2];
 }
}

class diferansiyel3H
{

public static void main(String args[]) throws IOException
{
  fm1 b1=new fm1();
  double y[];
  y=new double[2];
  y[0]=1.0;
  y[1]=2.0;
  Runge_Kutta rk=new Runge_Kutta("RKV9");
  double a[][]=rk.hesapla(b1,0.0,40.0,y,10000);
  Plot p1=new Plot(a[0],a[1]);
  p1.addData(a[0],a[2]);
  p1.setXlabel("zaman");
  p1.setYlabel("av ve avcı");
  p1.setPlabel("Av - Avcı modeli");
  p1.plot();
  Plot p2=new Plot(a[1],a[2]);
  p2.setXlabel("av");
  p2.setYlabel("avcı");
  p2.setPlabel("Av - Avcı modeli");
  p2.plot();
 }
}
```



(bolum8_019.jpg,Resimaltı: **Av-Avcı modeli denklem seti a=1.2 b= 0.6 c=0.8 d=0.3, Av-Avcı sayısının birbirine göre değişimi)**



(bolum8_020.jpg,Resimaltı: **Av-Avcı modeli denklem a=1.2 b= 0.6 c=0.8 d=0.3 Av-Avcı sayısının zamana göre değişimi)**

Bir başka örnek daha: Van der Pol kalp atışlarını incelerken kalp atış ritmi için Van der Pol denklemleri adını verdiğimiz denklem setini geliştirmiştir. Denklem

$\frac{d^2y}{dt^2} + \mu(y^2 - 1)\frac{dy}{dt} + y = 0$ şeklindedir. Denklem sistemine dönüştürürsek

$y=y_1$

$\frac{dy}{dt} = y_2$ olmak üzere

$\frac{dy_1}{dt} = y_2$

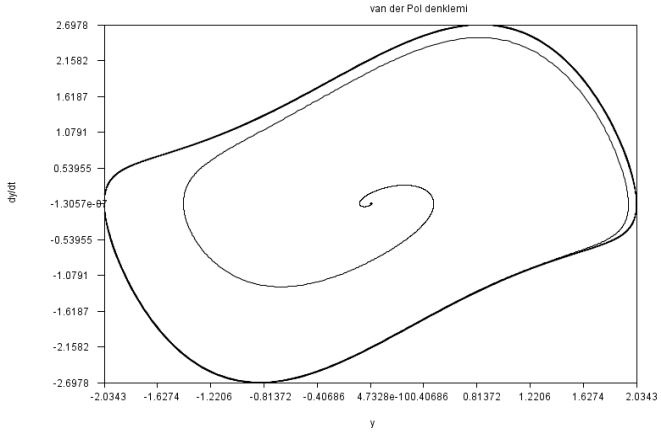
$\frac{dy_2}{dt} = -y_1 - \mu(y_1^2 - 1)y_2 = 0$

şekline dönüşür. Bu denklemi μ=1 için $y_1(0)$=0, $y_2(0)$=0.5 için çözersek :

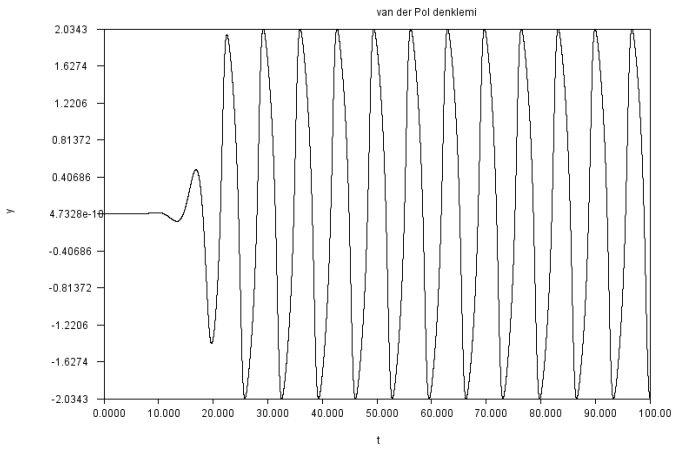
Program 8.7.5 Van der Pol denklem setinin 6 derece Runge-Kutta metodu ile çözülmesi

```
//=====================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class fm1 extends f_xi
{
//Van der Pol salınım deklemi
//kalp atış simulasyonu
```

```
 double mu=1.0;
 // d^2y/dt^2+mu(y^2-1)dy/dt=0
 double func(double x[],int x_ref)
 {
  //x[0] is x, x[1] is y
  if(x_ref==0) return x[2];
  else  return -x[1]-mu*(x[1]*x[1]-1.0)*x[2];
 }
}

class diferansiyel3D
{


public static void main(String args[]) throws IOException
 {

 fm1 b1=new fm1();
 double y[];
 y=new double[3];
 y[0]=0.0;
 y[1]=0.5;
 double a[][]=diferansiyel3.RK6(b1,0.0,40.0,y,10000);
 Plot p1=new Plot(a[1],a[2]);
 p1.setXlabel("y");
 p1.setYlabel("dy/dt");
 p1.setPlabel("van der Pol denklemi");
 p1.plot();
 Plot p2=new Plot(a[0],a[1]);
 p2.setXlabel("t");
 p2.setYlabel("y");
 p2.setPlabel("van der Pol denklemi");
 p2.plot();
 }
}
```



(bolum8_021.jpg,Resimaltı: **Van der Pol denklem seti y ve türevinin birbirine göre değişimi)**



(bolum8_022.jpg,Resimaltı: **Van der Pol denklem seti y zamana göre değişimi)**

Bir örnek daha verelim. Sönümlü harmonik hareket :

$\frac{d^2y}{dt^2} + b\frac{dy}{dt} + \omega^2 y = 0$ denklemiyle verilebilir.

Bu denklemi y(0)=1 $\frac{dy(0)}{dt} = 1$ sınır şartlarıyla ω =1 ve b=0.1 için çözeceğiz (b=0 sönümsüz basit harmonik harekettir.):

Denklemin dönüştürülmüş formu:

$$\frac{dy_1}{dt} = y_2$$

$$\frac{dy_2}{dt} = -by_2 - \omega^2 y$$

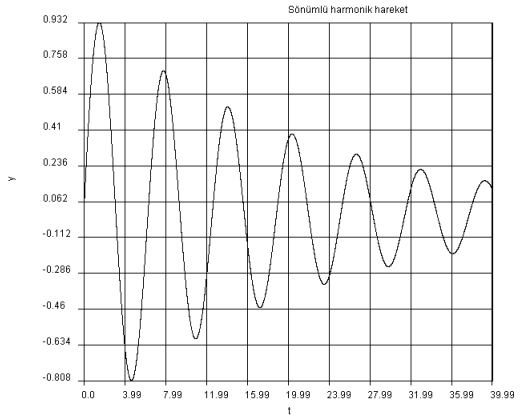
$y_1(0)=1$

$y_2(0)=1$

Program 8.7.6 Sönümlü harmonik hareket setinin 6 derece Runge-Kutta metodu ile çözülmesi

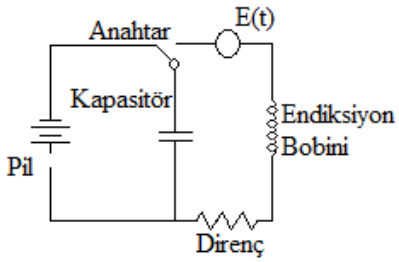
```
 //================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// ================================================
import java.io.*;


class fm1 extends f_xi
{
//Sönümlü harmonik hareket
 double b=0.1;
 double w=1.0;
 // d^2y/dt^2+b*dy/dt+w^2y=0
 double func(double x[],int x_ref)
 {
  //x[0] is x, x[1] is y
  if(x_ref==0) return x[2];
  else  return -b*x[2]-w*w*x[1];
 }
}
```

```
class diferansiyel3E
{

public static void main(String args[]) throws IOException
  {

  fm1 b1=new fm1();
  double y[];
  y=new double[3];
  y[0]=0.0;
  y[1]=1.0;
  Runge_Kutta rk=new Runge_Kutta("RK6");
  double a[][]=rk.hesapla(b1,0.0,40.0,y,10000);
  Plot p1=new Plot(a[1],a[2]);
  p1.setXlabel("y");
  p1.setYlabel("dy/dt");
  p1.setPlabel("Sönümlü harmonik hareket");
  p1.plot();
  Plot p2=new Plot(a[0],a[1]);
  p2.setXlabel("t");
  p2.setYlabel("y");
  p2.setPlabel("Sönümlü harmonik hareket");
  p2.plot();
 }
}
```



(bolum8_023.jpg ,Resimaltı: **Sönümlü harmonik hareket)**

Bir örnek daha verelim. Elektriksel bir devredeki harmonik hareket : Akımın sabit olmayıp zamana bağlı olarak değiştiği elektrik devrelerine sıkça rastlarız. Aşağıdaki şekilde böyle bir devre verilmiştir. Bu devre



(bolum8_024.jpg ,Resimaltı: **harmonik hareket oluşturan elektrik devresi)**
İçin diferansiyel denklem

$L\frac{d^2q}{dt^2}+R\frac{dq}{dt}+\frac{q}{C}-E(t)=0$ şeklinde verilebilir. Burada q kapasitörün elektrik enerjisi depolama miktarıdır (C, Coulomb). L endüktans (H, Henry), R direnç (□, Ohm), C kapasitans(F), $E(t)=E_0\sin(wt)$, w frekans , t zaman (s) dir. Bu denklemde akım şiddeti kapasitör yükünün türevidir.

$i=\frac{dq}{dt}$ i nin birimi Amperdir. Bu deklemin R=0 ve $E(t)=E_0\sin(wt)$ için analtik çözümü:

$q(t)=\frac{-E_0}{L(p^2-w^2)}\frac{w}{p}\sin(pt)+\frac{E_0}{L(p^2-w^2)}\sin(wt)$ şeklinde verilebilir. Veya akım şiddeti

$i=\frac{dq(t)}{dt}=\frac{-E_0 w}{L(p^2-w^2)}\sin(pt)+\frac{E_0 w}{L(p^2-w^2)}\sin(wt)$ şeklini alır. Burada

$p=\frac{1}{\sqrt{LC}}$ dir.

Bu denklemi q(0)=0 ve $i(0)=\frac{dq(0)}{dt}=0$ sınır şartlarıyla $w^2=3.5\ 1/s^2$ p=2, L=1 H, E_0=1 V, C=0.25 F, için çözeceğiz.

Denklemin dönüştürülmüş formu:

$$\frac{dy_1}{dt}=y_2$$

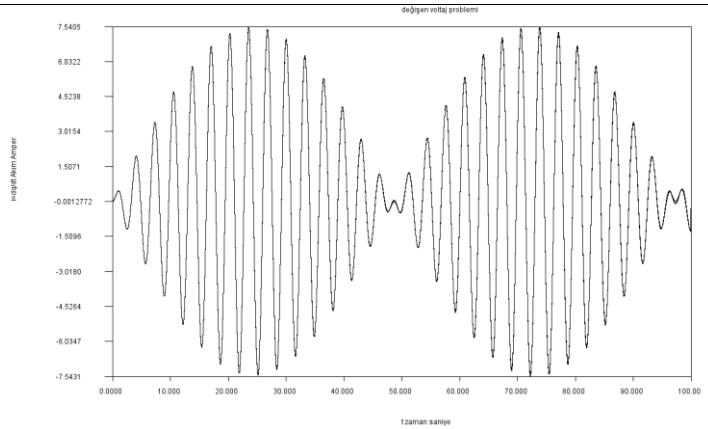
$$\frac{dy_2}{dt}=\frac{\left[-Ry_2-\frac{y_1}{C}+E_0\sin(wt)\right]}{L}$$

$y_1(0)=0$
$y_2(0)=0$

şeklinde yazılabilir. Çözüm :

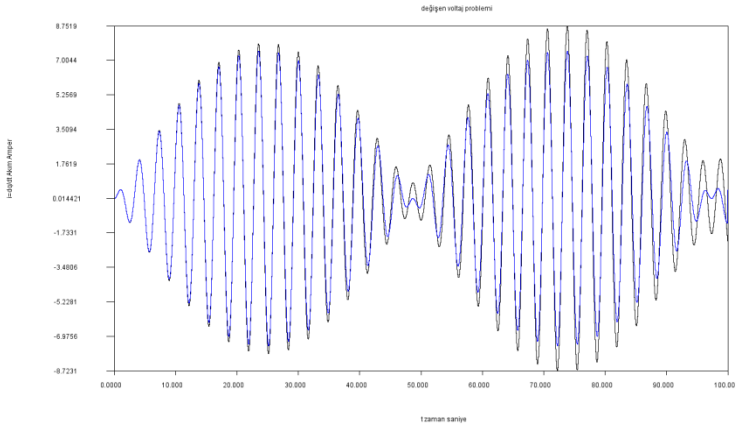
```
//=====================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class fm1 extends f_xi
{
 double E0=1.0;  //Volt
 double L=1.0;   // Henry
 double C=0.25;  // Coulomb
 double w2=3.5;  // s^2
 double R=0.0;     // Ohm
 double w=Math.sqrt(w2);
 double p=1.0/Math.sqrt(L*C);
 // Ld^2q/dt^2+R*dq/dt+q/C-E(t)=0
 double func(double x[],int x_ref)
```

```
 {
  //x[0] is t, x[1] is q x[2] is i
  if(x_ref==0) return x[2];
  else  return (-R*x[2]-x[1]/C+E0*Math.sin(w*x[0]))/L;
 }
 double i(double t)
 {
 return -E0/L/(p*p-w2)*w*Math.cos(p*t)+E0/L/(p*p-w2)*w*Math.cos(w*t);
 }
 double q(double t)
 {
 return -E0/L/(p*p-w2)*w/p*Math.sin(p*t)+E0/L/(p*p-w2)*Math.sin(w*t);
 }
}

class diferansiyel3J
{

public static void main(String args[]) throws IOException
  {

  fm1 b1=new fm1();
  double y[];
  y=new double[3];
  y[0]=0.0;
  y[1]=0.0;
  int n=800000;
  double a[][]=diferansiyel3.RK6(b1,0.0,100.0,y,n);
  Plot p1=new Plot(a[0],a[2]);
  double t=0;
  double dt=100.0/n;
  for(int i=0;i<(n-1);i++) {a[1][i]=b1.i(t);t+=dt;}
  p1.addData(a[0],a[1]);
  p1.setXlabel("t zaman saniye");
  p1.setYlabel("i=dq/dt Akım Amper");
  p1.setPlabel("değişen voltaj problemi");
  p1.plot();
 }
}
```



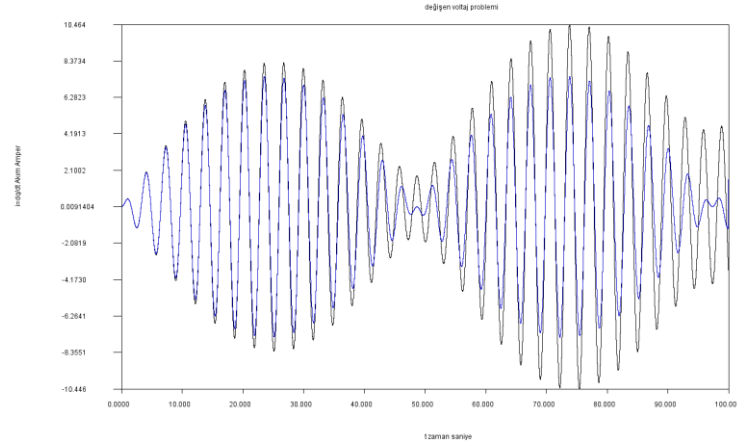
(bolum8_025.jpg ,Resimaltı: **elektrik devresi harmonik hareket N=800000)**

Akım değerleri için hem analitik hem de sayısal çözüm kulanılmış ve birlikte çizilmiştir. Değerler iyi bir uyum göstermiştir. Hesap için 800000 eleman alınmıştır. N=50000 aldığımızda hatanın oluşmasını gözlemleyebiliriz



(bolum8_026.jpg ,Resimaltı: **elektrik devresi harmonik hareket N=50000)**

n=25000 için hata daha da büyür.



(bolum8_027.jpg ,Resimaltı: **elektrik devresi harmonik hareket N=25000)**

bir örnek daha :Yerçekiminin olduğu sürtünmeli hava ortamında bir top mermisi attığımızı varsayalım. Kartezien koordinat sisteminde denklemimiz :

$$\frac{d^2x}{dt^2} = -\gamma(\sqrt{v_x^2 + v_y^2})v_x$$

$$\frac{d^2y}{dt^2} = -g - \gamma(\sqrt{v_x^2 + v_y^2})v_y$$

olur burada $\gamma = \frac{k}{m}$, k hava sürtünme katsayısı, v_x x doğrultusundaki hız bileşeni, v_y y doğrultusundaki hız bileşeni, m kütle, g yer çekimi ivmesi, t zamandır.

t=x[0] , x=x[1], y=x[2], v_x=x[3], v_y=x[4] değişken dönüşümü yapar isek, denklem sistemimiz

$$\frac{dx[1]}{dx[0]} = x[3]$$

$$\frac{dx[2]}{dx[0]} = x[4]$$

$$\frac{dx[3]}{dx[0]} = -\gamma(\sqrt{x^2[3] + x^2[4]})x[3]$$

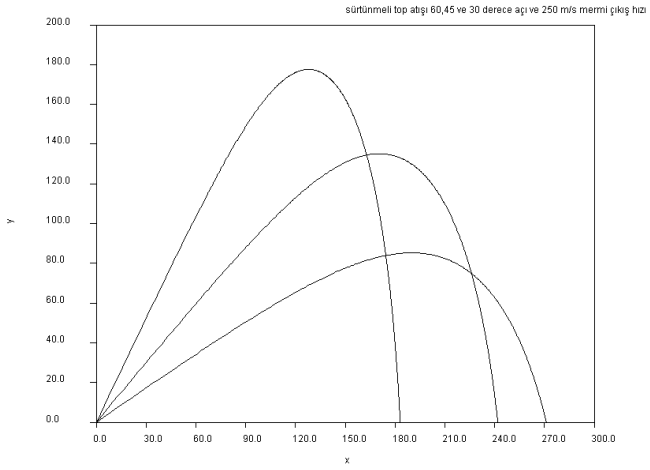
$$\frac{dx[4]}{dx[0]} = -g - \gamma(\sqrt{x^2[3] + x^2[4]})x[4]$$

olur sınır şartları olarak x[0]=0 da x[1]=0 x[2]=0,
V_0=250 m/s x[3]= V_0 sin(□□□m/s, x[4]= V_0 cos(□□□ m/s, g=9.806 m/s^2, □=0.01 ,□□ =30,45,60 derece alırsak

Program 9.7-7 sürtünmeli top atışı probleminin 6 derece Runge-Kutta metodu ile çözülmesi

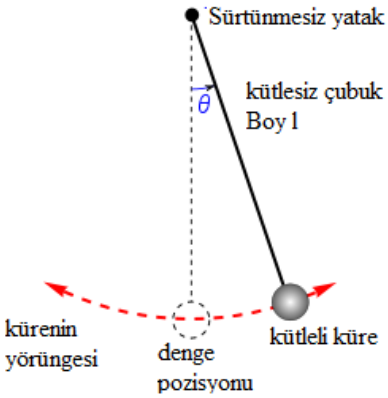
```
 //=====================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =====================================================
import java.io.*;


class fm1 extends f_xi
{
 double gamma;
              public fm1(double gammai)
   {gamma=gammai;}
 // sürtünmeli eğik atış problemi
 double func(double x[],int x_ref)
 {
  //x[0] is x, x[1] is y
  double g=9.806;
  if(x_ref==0) return x[3];
  else if(x_ref==1) return x[4];
  else if(x_ref==2) return -gamma*Math.sqrt(x[3]*x[3]+x[4]*x[4])*x[3];
  else            return -g - gamma*Math.sqrt(x[3]*x[3]+x[4]*x[4])*x[4];
 }
}

class diferansiyel3F
{

public static void main(String args[]) throws IOException
  {
  fm1 b1=new fm1(0.01);
  double y[];
  y=new double[4];
  double V0=250.0;//  m/s
  double alfa=Math.PI/3; //60 derece
  y[0]=0.0;
  y[1]=0.0;
  y[2]=V0*Math.sin(alfa);
  y[3]=V0*Math.cos(alfa);
    Runge_Kutta rk=new Runge_Kutta("RK6");
  double a[][]=rk.hesapla(b1,0.0,20.0,y,1000);
  Plot p1=new Plot(a[1],a[2]);
  alfa=Math.PI/4;  //45 derece
  y[2]=V0*Math.sin(alfa);
  y[3]=V0*Math.cos(alfa);
  double b[][]=rk.hesapla(b1,0.0,20.0,y,1000);
  p1.addData(b[1],b[2]);
  alfa=Math.PI/6;  //30 derece
  y[2]=V0*Math.sin(alfa);
  y[3]=V0*Math.cos(alfa);

  double c[][]=rk.hesapla(b1,0.0,20.0,y,1000);
  p1.addData(c[1],c[2]);
  double xa[]=new double[2];
  double ya[]=new double[2];
  xa[0]=0;xa[1]=12.0;
```

```
  ya[0]=0;ya[1]=0.0;
  p1.addData(xa,ya);
  p1.setXlabel("x");
  p1.setYlabel("y");
  p1.setPlabel("sürtünmeli top atışı 60,45 ve 30 derece açı ve 250 m/s mermi çıkış hızı");
  p1.setMinMax(0,300,0,200);
  p1.plot();
  }
}
```



(bolum8_028.jpg ,Resimaltı: **sürtünmeli top atışı problemi)**

Başka bir örnek olarak sarkaç problemine bakalım. Sarkaç mekanik sistemlerde sıkça karşımıza çıkan bir mekanizmadır. l uzunluğunda bir sarkaçın salınımı



(bolum8_029.jpg ,Resimaltı: **Sarkaç sistemi)**

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l}\sin(\theta)$$

denklemi ile ifade edilebilir. Buradaki sarkaç salınım açısı, t zaman, g yerçekimi ivmesi(m/s^2) değişkenleridir. Bu denklem lineer olmıyan bir denklemdir. Çok küçük açılar için sin() Taylor formülü açılımından

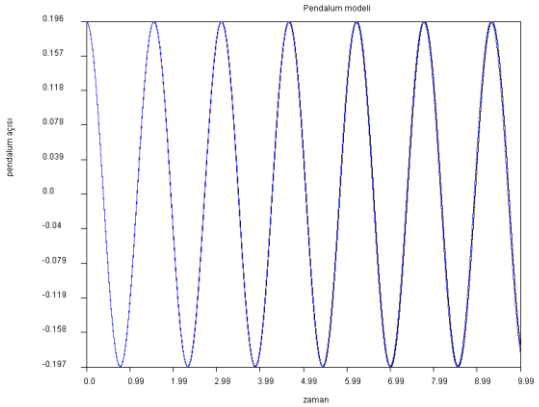
$$\sin(\theta) = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \frac{\theta^9}{9!} + ...$$

$\frac{d^2\theta}{dt^2} = -\frac{g}{l}\theta$ lineer diferansiyel denklemi yazılabilir.Bu denklemin analitik çözümü

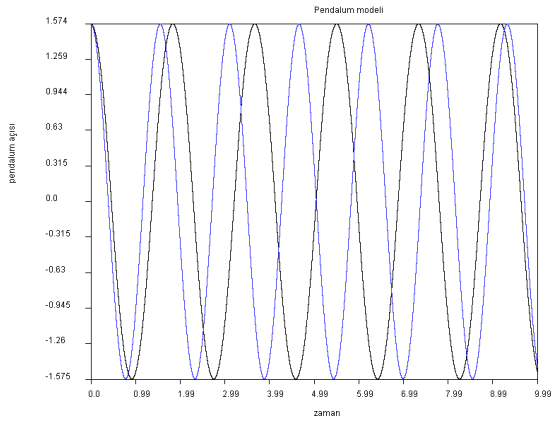
$\theta(t) = \theta_0 \cos(\sqrt{\frac{g}{l}}t)$ şeklinde yazılabilir. Denklemin lineer olmıyan orijinal formunun analitik çözümü mevcut değildir, fakat sayısal yöntemlerle çözülebilir. Programımızda hem lineer olmıyan denklem hem de lineer denklem birlikte çözülmüştür. Çıktılardan da görüleceği gibi faz açısı küçük olduğunda (□/16 radyan) iki çözüm birbirine oldukça yaklaşmaktadır. Ancak faz açısı büyüdüğünde (□/2 radyan) lineer yaklaşımın verdiği çözüm gerçek çözümden sapmaktadır. Periodlar arasında da farklar oluşmaktadır.

Program 9.7-8 sarkaç probleminin 6 derece Runge-Kutta metodu ile çözülmesi

```
 //=======================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =======================================================
import java.io.*;

class fm1 extends f_xi
{
// Sarkaç denklemi
 double g=9.806;  //yerçekimi ivmesi m/s2
 double l=0.6;   //sarkaç boyu m

 double func(double x[],int x_ref)
 {
   //x[0] is t, x[1] is teta(sarkaç açısı) x[2] v hız
   if(x_ref==0) return x[2];
   else return -g/l*Math.sin(x[1]);
 }
}

class diferansiyel3I
{

public static void main(String args[]) throws IOException
{
  fm1 b1=new fm1();
  double y[];
  y=new double[2];
  y[0]=Math.PI/2.0; //90 derece salınım açısı olduğunda
  y[1]=0.0;
  Runge_Kutta rk=new Runge_Kutta("RK6");
  double a[][]=rk.hesapla(b1,0.0,10.0,y,500000);
  int n=a[0].length;
  double teta[]=new double[n];
  for(int i=0;i<n;i++) {teta[i]=y[0]*Math.cos(Math.sqrt(b1.g/b1.l)*a[0][i]);}
  Plot p1=new Plot(a[0],a[1]);
  p1.addData(a[0],teta);
  p1.setXlabel("zaman");
  p1.setYlabel("pendalum açısı");
  p1.setPlabel("Pendalum modeli");
  p1.plot();
 }
}
```



(bolum8_030.jpg ,Resimaltı: **sarkaç probleminin 6 derece Runge-Kutta metodu ile çözülmesi $\theta=\pi/16$ radyan, yerçekimi ivmesi 9.806 m/s^2, sarkaç boyu λ = 0.6 m)**



(bolum8_031.jpg ,Resimaltı: **sarkaç probleminin 6 derece Runge-Kutta metodu ile çözülmesi $\theta=\pi/2$ radyan, yerçekimi ivmesi 9.806 m/s^2, sarkaç boyu λ = 0.6 m)**

9.8 Diferansiyel denklemlerde başlangıç noktası, atış problemi

Şu ana kadarki tüm diferansiyel denklem çözümlerinde başlangıç sınır değerlerinin verildiğini varsaydık. Gerçek problemlerde bu koşulu her zaman başlangıç sınır değeri olarak bulmak mümkün değildir, bazen verilen bölgenin son kısmı için de verilebilir. Böyle bir durumda şu stepler gerçekleştirilerek çözüm bulunur

Adım 1 : Başlangıç değeri için bir deger tahmini yapın ve diferansiyel denklemi çözün, h
Adım 2 : Adım 1 i 3-5 kere tekrar edin, her seferinde başlangıç değerine karşı gelen bitiş değerini saptayın
Adım 3 : Başlanıç ve bitiş değerleri arasında interpolasyon veya egri uydurma programlarından birisini kullanarak bir fonksiyon oluşturunuz. Bu fonksiyonu kullanarak bitiş değerinden gerçek başlangıç değerini hesaplayınız.
Adım 3 : Gerçek başlangıç değerini kullanarak denklemi son bir kere daha çözünüz.

Örnek denklem olarak akışkanlar mekaniğinde kullanılan Blassius düz levha problemine bakalım. Bu problem sayısal olarak çözülen ilk problemlerden birisidir. Denklem
df''/dν +0.5*f*f'' = 0 şeklinde verilmiştir. Denklemi ayırdığımızda
f'''=-0.5*x[1]*x[2]
f''=x[2]
f'=x[1]

şeklini alır.
Bu denklemin sınır şartları

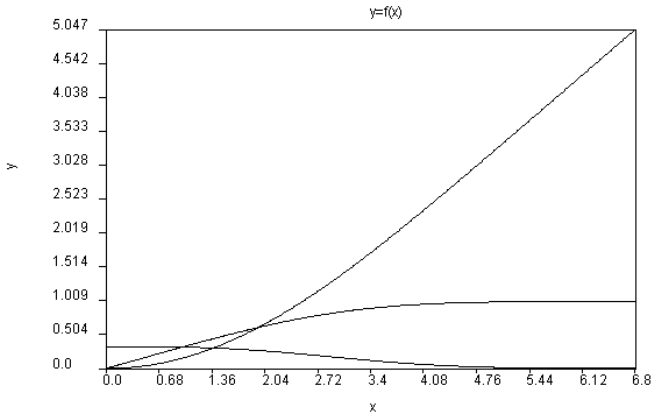
x[0]=0 da x[2]=0
x[0]= ∞ da x[2]=1
x[0]=0 da x[3]=0

olarak verilmiştir. x[1] in sınır şartı verilmediğinden, belli bir tahmin vererek başlarız (örnek problemde ilk başlangıç değeri 0.3 olarak verilmiştir). Daha sonra x[1] değeri değiştirilerek x[2] nin x[0]=6.8 (sonsuz yerine bu değer alınmıştır) aldığı değerler saptanmış ve kübik şerit interpolasyonuyla x[1]in alması gereken değer bulunmuştur (x[0]=0 da x[1]=0.3267065268657032, noktası için x[0]=6.8 de x[2] = 0.9992172933326781 değeri bulunmuştur.).

Program 9.8-1 Blassius düz levha probleminin çözümü

```
 //=======================================================
// Numerical Analysis package in java
// example to show differential equation solution
// and differentiation (derivative) functions
// Dr. Turhan Coban
// =======================================================
//Blasius flat plate problem

import java.io.*;

class fm1 extends f_xi
{
//multivariable function
 double func(double x[],int x_ref)
 {
  //Blasius flat plate diferential equation
  //f'''+0.5*ff'' = 0
  //f'''=-0.5*x[1]*x[2] f''=x[1] f'=x[2]
  double a=0;
  if(x_ref==0) a= -0.5*x[1]*x[3];
  if(x_ref==1) a= x[1];
  if(x_ref==2) a= x[2];
  return a;
 }
}

class diferansiyel1X
{
 public static void main(String args[]) throws IOException
 {
 fm1 b3=new fm1();
 double y[]=new double[3];
 y[0]=0.3; //y0 first guess for f''
 y[1]=0.0;
 y[2]=0.0;
 double a[][]=new double[3][501];
 //solution of the differential system :
 double xi[]=new double[20];
 double fi[]=new double[20];
 //shooting method
 Runge_Kutta RK6=new Runge_Kutta("RK6");
 for(int i=1;i<20;i++)
 {
 a=RK6.hesapla(b3,0.0,6.0,y,500);
 xi[i]=a[2][500];
 fi[i]=y[0];
```

```
 y[0]+=0.1;  //change the boundary value
 }
 //curve fit to find actual boundary value
 double S[][]=SCO11F.cubic_spline(xi,fi,0,0);
 y[0]=SCO11F.funcSpline(S,1.0);
 a=RK6.hesapla(b3,0.0,6.0,y,500);
 //Text.print(Text.T(a));
 System.out.println(Matrix.toString(Matrix.T(a)));
 }
}
```



(bolum8_032.jpg ,Resimaltı: **Blassius düz levha probleminin çözümü)**

9.9 Problemler

PROBLEM 1
dy/dx = y*sin(x)* sin(x) diferansiyel denklemini x=0;x=1 aralığında h=dx=0.2 için
x=0 y=2 başlangıç değerini kullanarak klasik dördücü derece runge kutta formülü ile çözünüz

PROBLEM 2
$d^2y/dx^2+6\ dy/dx= \sin(\pi x)$ diferansiyel denkleminin sınır değerleri x=0;x=1 aralığında x=0 da y=2 ve x=1 de y=-1 olarak verilmiştir. h=0.02 step boyutu için diferansiyel denklemi çözünüz.

PROBLEM 3
dy/dx= sin(πx) diferansiyel denkleminin sınır değerleri x=0;x=1 aralığında x=0 da y=0 olarak verilmiştir. h=0.02 step boyutu için diferansiyel denklemi çözünüz.

PROBLEM 4
Falkner-Scan akış denklemi
$f''' + f*f'' + \beta(1-f'^2) = 0$ şeklinde verilmiştir. Denklemi ayırdığımızda
Bu denklemin sınır şartları
$f(0)=0$
$f'(0)=1$
$f'(\propto)=1$
$\beta = 0.3$ ve $\beta = 1.0$ için denklemi çözünüz. $\propto$ değeri yerine 8 değerini alabilirsiniz.

PROBLEM 5
$y'' = x^2 - y^2 - 3y'$
diferansiyel denklemini y(0)=0, y'(0)=0 şartları için çözünüz.

PROBLEM 6
$y''' = y'^2 - 4y$
diferansiyel denklemini y(0)=0, y'(0)=1, y''(0)=0 şartları için çözünüz.

PROBLEM 7
x’ = 3x-y
y’=x+y
diferansiyel denklemini x(0)=0.2, ve y(0)=0.5 şartları için çözünüz.

PROBLEM 8
$2x''(t)-5x'(t)-3x(t)=45e^{2t}$ denklemini x(0)=0 ve x’(0)=1 şartları için çözünüz.

PROBLEM 9
x’(t)=x(t)-x(t)y(t)
y’(t)=-y(t)+x(t)y(t) denklem sistemini x(0)=4 ve y(0)=1 şartları t=[0,8] bölgesinde çözünüz.

PROBLEM 10
$x'(t)=\frac{2t}{1+t^2}x'(t)-\frac{2}{1+t^2}x(t)+1$ denklem sistemini x(0)=1.25 ve x(4)=-0.95 şartları t=[0,4] bölgesinde çözünüz.

PROBLEM 11
$y'' = -(y')^2 - y + \ln x$ denklem sistemini çözünüz.
Sınır şartları : 1<=x<=2 y(1)=0 y(2)=ln2

PROBLEM 12
$y' - 2(1+y^2) = 0$ diferansiyel denkleminde, *y*(0)=0 başlangıç değeri için, *h*=0,25 adım aralığı ile *y*(1) değerini Heun yöntemi ile çözünüz.

PROBLEM 13
$\frac{dy}{dx} = yx^2 - y$ Diferansiyel denkleminin x=0 ve x=2 aralığında çözümü araştırılmaktadır. Başlangıç değeri olarak y(0)=1 verilmiştir.

a) Başlangıç değer problemini analitik olarak x=0 dan x=2 ye çözümleyiniz. Çözümü ölçekli olarak çiziniz.

b) h=0.5 adım aralığı ile dördüncü derece Runge-Kunta yöntemi ile sayısal çözümleme ile y(2) ’yi hesaplayınız.

c) Çözümü analitik çözümle karşılaştırmak üzere yaklaşım değerlerini (a) şıkkında çizdiğiniz grafiğe taşıyınız. Adım aralıklarında bulduğunuz sonuçlardaki yüzde bağıl yanlış değerleri nedir?

PROBLEM 14

$\frac{dy}{dx} = -5y + e^{-2x}$ diferansiyel denkleminde başlangıç koşulu x_0=0 için y_0=1 olarak verilmektedir. *y*(1) ’i , *h*=0,2 alarak, simpson 1/3 yöntemi ile çözünüz.

PROBLEM 15
$\frac{dy}{dx} = \frac{1}{x+y}$ diferansiyel denkleminde başlangıç koşulu x_0=0 için y_0=2 olarak verilmektedir. *y*(0,6) ’yı , *h*=0,2 alarak,
a-) 4.derece Runge-Kutta yöntemi ile,
b-) Euler yöntemi ile çözünüz.

PROBLEM 16
Bir cismin soğuma hızı ile aşağıdaki diferansiyel denklem belirlenmiştir.

$$\frac{dT}{dt} = -k(T - T_a)$$

Burada T cismin (^{o}C) olarak sıcaklığı, T_a çevre ortamın (^{o}C) olarak sıcaklığı ve k orantı sabiti (dakika^{-1}) dir. Böylece bu eşitlik soğuma hızının cisim ve çevre ortam sıcaklık farkı ile orantılı olduğunu ortaya koymaktadır. Eğer bir metal bilya 90 ^{o}C sıcaklığına ısıtılıp T_a=20 ^{o}C sıcaklıkta tutulan bir su banyosuna atılıyorsa ve eğer k=0.1 dakika^{-1} ise bilyanın 30 ^{o}C sıcaklığa soğuması için geçen süreyi,

a-) Analitik olarak çözümleyiniz.

b-) Bir sayısal çözümleme tekniği kullanarak belirleyiniz.

PROBLEM 17

$\frac{dy}{dx} = yx^2 - y$ Diferansiyel denkleminin x=0 ve x=2 aralığında çözümü araştırılmaktadır. Başlangıç değeri olarak y(0)=1 verilmiştir.

a) Başlangıç değer problemini analitik olarak x=0 dan x=2 ye çözümleyiniz. Çözümü ölçekli olarak çiziniz.

b) h=0.5 adım aralığı ile Heun sayısal çözümleme yöntemi ile y(2) 'yi hesaplayınız.

c) Çözümü analitik çözümle karşılaştırmak üzere yaklaşım değerlerini (a) şıkkında çizdiğiniz grafiğe taşıyınız. Adım aralıklarında bulduğunuz sonuçlardaki yüzde bağıl yanlış değerleri nedir?

PROBLEM 18

Bir kimyasal reaksiyonda bir molekül A, bir molekül B ile birleşip bir molekül C kimyasal ürününü oluşturmaktadır. C 'nin zamana bağlı konsantrasyonu y(t) 'nin bulunması için aşağıdaki başlangıç değer probleminin çözümlenmesi gerekmektedir.

y' = k(a – y) (b– y) y(0)=0

Burada k pozitif bir sabit , a ve b ise sırasıyla A ve B maddelerinin başlangıç konsantrasyonlarıdır.

k=0.01 , a= 70 milimole/litre , b= 50 milimole/litre olduğunu varsayalım. 4. derece Runge-Kutta yöntemini kullanarak h=0.5 için t=1 saniyedeki konsantrasyon değerini tahminleyiniz.

PROBLEM 19

Bir cismin soğuma hızı ile aşağıdaki diferansiyel denklem belirlenmiştir.

$$\frac{dT}{dt} = -k(T - T_a)$$

Burada T cismin (^{o}C) olarak sıcaklığı, T_a çevre ortamın (^{o}C) olarak sıcaklığı ve k orantı sabiti (dakika^{-1}) dir. Böylece bu eşitlik soğuma hızının cisim ve çevre ortam sıcaklık farkı ile orantılı olduğunu ortaya koymaktadır. Eğer bir metal bilya 90 ^{o}C sıcaklığına ısıtılıp T_a=20 ^{o}C sıcaklıkta tutulan bir su banyosuna atılıyorsa ve eğer k=0.1 dakika^{-1} ise bilyanın 30 ^{o}C sıcaklığa soğuması için geçen süreyi,

a-) Analitik olarak çözümleyiniz.

b-) Bir sayısal çözümleme tekniği kullanarak belirleyiniz.

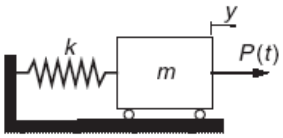
PROBLEM 20

$$\frac{dy}{dx} = yx^2 - y$$

Diferansiyel denkleminin x=0 ve x=2 aralığında çözümü araştırılmaktadır. Başlangıç değeri olarak y(0)=1 verilmiştir.

a) Başlangıç değer problemini analitik olarak x=0 dan x=2 ye çözümleyiniz. Çözümü ölçekli olarak çiziniz.
b) h=0.5 adım aralığı ile Heun sayısal çözümleme yöntemi ile y(2) 'yi hesaplayınız.
c) Çözümü analitik çözümle karşılaştırmak üzere yaklaşım değerlerini (a) şıkkında çizdiğiniz grafiğe taşıyınız. Adım aralıklarında bulduğunuz sonuçlardaki yüzde bağıl yanlış değerleri nedir?

PROBLEM 21



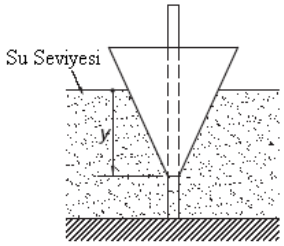
(bolum8_033.jpg ,Resimaltı: Problem 21)
Şekilde görülen yay-kütle sistemindeki P(t) kuvveti

$$P(t)=\begin{cases}10t \;\; N & t<2s\\ 20\,N & t\geq 2s\end{cases}$$

olarak verilmiştir. Hareket diferansiyel denklemi

$\frac{d^2y}{dt^2}=\frac{P(t)}{m}-\frac{k}{m}y$ şeklindedir. m=2.5 kg, k=75 N/m için kütlenin maksimum hareketini belirleyin.

PROBLEM 22



(bolum8_034.jpg ,Resimaltı: Problem 22)

koni şeklindeki su üzerinde yüzen bir duba bir dik çubuk üzerinde serbestçe kayabilmektedir. Duba herhangi bir etki ile denge durumundan saptırıldığında osilasyon hareketi yapmaktadır, bu osilasyonun diferansiyel denklemi :

$\frac{d^2y}{dt^2}=g(1-ay^3)$ şeklindedir. Eğer a=16 m^{-3}, g=9.80665 m/s^2 ise duba y=0.1 m kaldırılıp bırakıldığında oluşacak osilasonun genliğini(m) ve periodunu(s) bulunuz.

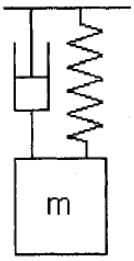
PROBLEM 23

Sürtünmeli top atışı problemini örnek problem olarak 6 derece Runge-Kutta metodu ile örnek problem olarak çözmüştük, şimdi aynı problemi hızı 250 m/s, yerçekimi ivmesi 9.806 m/s^2, sürtünme katsayısı γ = (k / m)= 0.01 değerleriyle 750 m ilerideki hedefi vurmak için gerekli açı θ yı bulmak için çözünüz.

PROBLEM 24

Otomobillerde ve bir çok dinamik makinada titreşim kontrolünü sağlamak için bir yay ve bir titreşim söndürücüden (amortisör) oluşan bir sistem kullanılır. Bu sisteme bir diferansiyel denklem olarak bakacak olursak

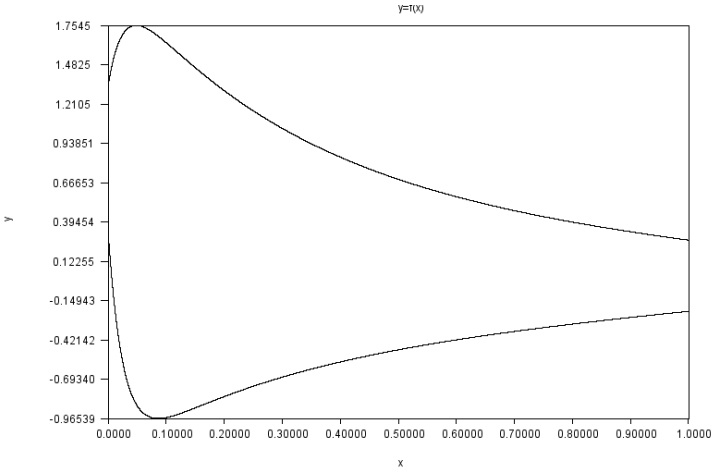
$m\frac{d^2x}{dt^2}+k_d\frac{dx}{dt}+k_sx=0$ Şeklinde yazılabilir. Burada m kütle k_d söndürücü katsayısı, k_s yay katsayısıdır.
Denklemi m=5 kg, k_s =500 N/m, k_d =33 N/m değerleri için çözünüz.



(bolum8_035.jpg ,Resimaltı: Problem 24)

PROBLEM 25 Java fonksiyonu olarak verilen diferansiyel denklem setini y(0)={4/3,1/3) x0=0,xn=1.0 sınırları ile çözünüz.

```
class fm1 extends f_xi
{
//multivariable function
 double func(double x[],int x_ref)
 {
   //x[0] is x, x[1] is y

   if(x_ref==0) return 9*x[1]+24*x[2]+5*Math.cos(x[0])-
1.0/3.0*Math.sin(x[0]);
   else if(x_ref==1) return -24.0*x[1]-51*x[2]-
9*Math.cos(x[0])+1.0/3.0*Math.sin(x[0]);
   else return 0;
 }
}
```



(bolum8_036.jpg ,Resimaltı: Problem 25)

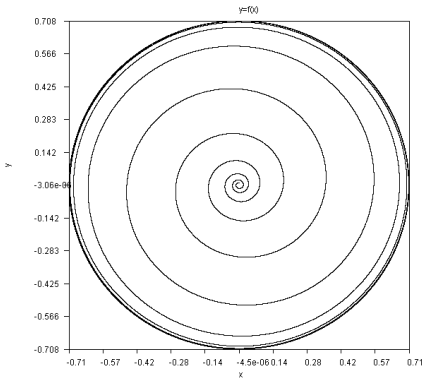
PROBLEM 26

Hopf denklemi

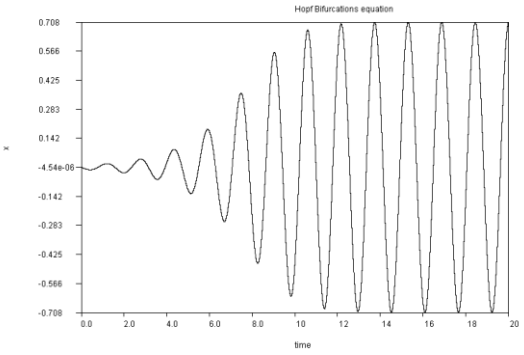
$$\frac{dx}{dt} = -(g + x^2 + y^2)x - wy$$

$$\frac{dy}{dt} = -(g + x^2 + y^2)y + wx$$

Olarak verilir. Bu denklemi g=-0.5,w=4, ve başlangıç değerlerini x=0,y=0.01 t=0 saniyede . $t_{bitiş}$=20 s



(bolum8_037.jpg ,Resimaltı: Problem 26)



(bolum8_038.jpg ,Resimaltı: Problem 26)

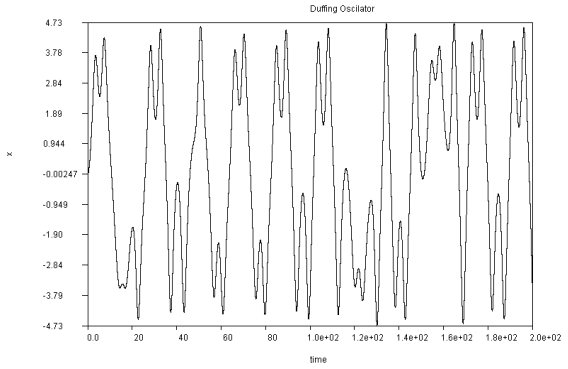
PROBLEM 27

The Duffing sarkacı denklemi

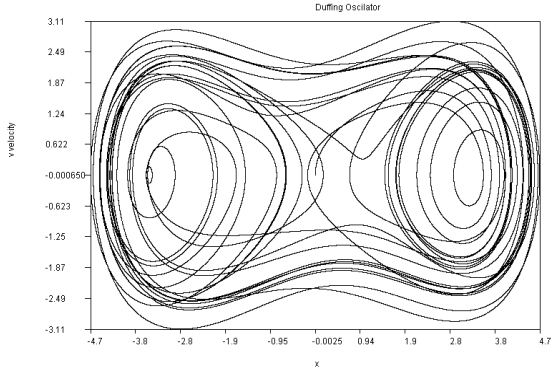
$$\frac{dx}{dt} = v$$

$$\frac{d^2x}{dt^2} = \frac{dv}{dx} = -0.2v + x - 0.1x^3 + \cos(t)$$

Bu diferansiyel denklemi x=v=0 , t=0 da 0 dan t=200 s periodu için çözünüz.



(bolum8_039.jpg ,Resimaltı: Problem 27)



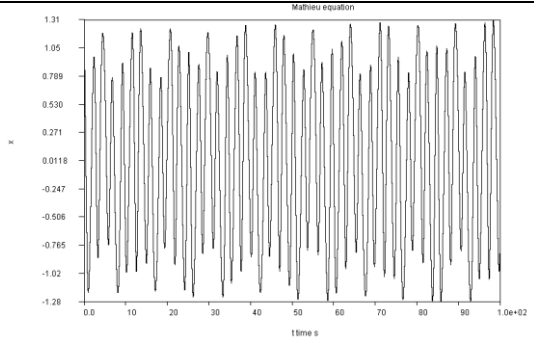
(bolum8_040.jpg ,Resimaltı: Problem 27)

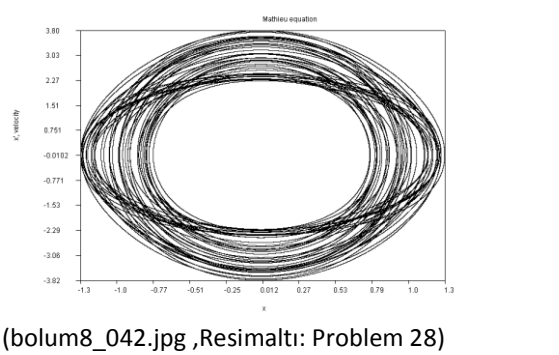
PROBLEM 28

The Mathieu denklemi

$$\frac{d^2x}{dt^2} + (\delta + 2\varepsilon\cos(2t))x = 0$$

Şeklinde verilmiştir. Bu denklemi δ=8, ε=2.5, için x=0,x'=0 başlangıç şartlarıyla ve 0<=t<=100 s. Periodunda çözünüz.





(bolum8_042.jpg ,Resimaltı: Problem 28)

(bolum8_041.jpg ,Resimaltı: Problem 28)	

PROBLEM 29

H'enon-Heiles denklemi

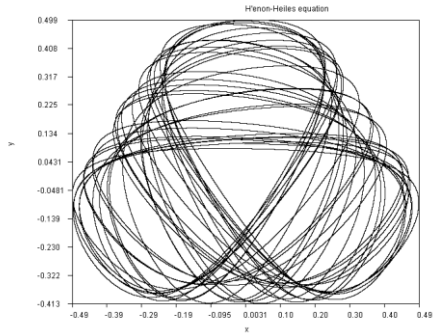
$$\frac{dx}{dt} = P_x$$

$$\frac{dy}{dt} = P_y$$

$$\frac{dP_x}{dt} = -x - 2xy$$

$$\frac{dP_y}{dt} = -y - x^2 + y^2$$

Şeklinde verilmiştir. Bu denklemi x=0,y=0.1 başlangıç koşulları ve P_x=0.49057, P_y=0 alarak of 0<=t<=200 s. Periodu için çözünüz.

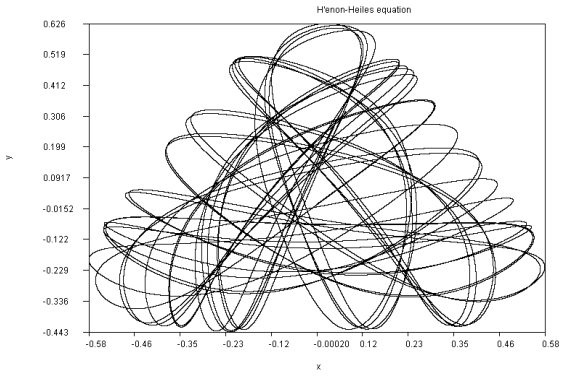


(bolum8_043.jpg ,Resimaltı: Problem 29)

PROBLEM 30

H'enon-Heiles denklemi Problem 29 da tanımlanmıştı. Aynı denklemi

x=0,y=0 ve P_x= 0.35355, P_y= 0.35355 başlangıç koşullarıyla ve 0<=t<=200 s. Periodu için çözünüz.



(bolum8_044.jpg ,Resimaltı: Problem 30)

PROBLEM 31

Josephson bağlantısı adını verdiğimiz fiziksel olay elektrik alanında bir çok ilginç sonuçlar veren kuantumsal bir kaotik davranıştır. Bu davranışın kuantum faz farkı , *φ*, diferansiyel denklemini boyutsuz formda yazarsak

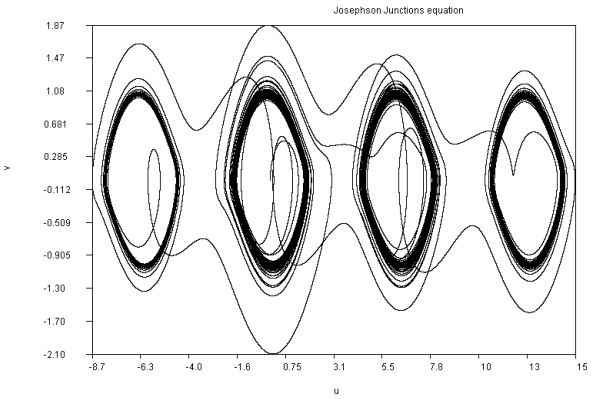
$$\frac{d^2\Phi}{dt^2} + \frac{1}{\sqrt{\beta_c}}\frac{d\Phi}{dt} + \sin\Phi = A\sin\Omega t$$

denklemi elde edilir. Burada β_c McCumber parametresidir ve *Ω* sürme akımının normalize edilmiş açısal frekansıdır. *u* = $v = \Phi$ ve $v = \frac{d\Phi}{dt}$, tanımlarını yapacak olursa diferansiyel denklem

$$\frac{du}{dt} = v$$

$$\frac{dv}{dt} = \mathrm{Asin}(\Omega t) - rv - \sin(u)$$

Formunu alır. Eğer çözüm parametrelerini A=0.9045, r=0.5, Ω=0.47 olarak alırsak ve başlangıç parametreleri u=v=0 ve 0<=t<=1200 s ise diferansiyel denklemi çözünüz.



(bolum8_045.jpg ,Resimaltı: Problem 31)

PROBLEM 32

Üç gezegen sisteminin diferansiyel denklemleri

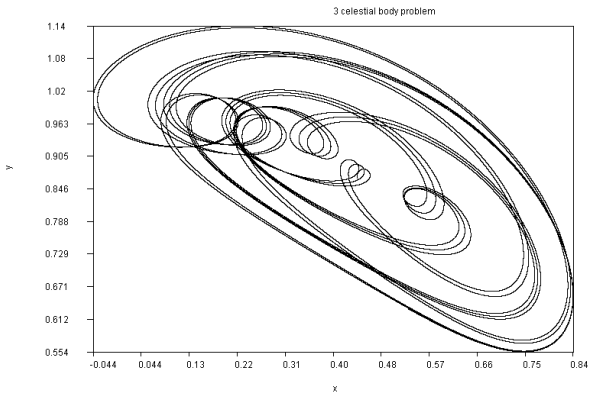
$$\frac{dx}{dt} = p + y$$

$$\frac{dy}{dt} = q - x$$

$$\frac{dp}{dt} = q - \frac{m(x+n)}{\left[(x+n)^2 + y^2\right]^{3/2}} - \frac{n(x-m)}{\left[(x+n)^2 + y^2\right]^{3/2}}$$

$$\frac{dq}{dt} = -p - \frac{my}{\left[(x+n)^2 + y^2\right]^{3/2}} - \frac{ny}{\left[(x+n)^2 + y^2\right]^{3/2}}$$

Şeklinde verilebilir. Burada p ve q, x ve y koordinatlarının nisbi momentleridir. Üçü sitemlerde Lagrange noktaları dediğimiz 5 tane denge noktası mevcuttur. Dünya-güneş-ay sisteminde bu noktalardan biri L4 noktasıdır bu nokta ($x_{ortalama}$,$y_{ortalama}$) = (0.48, 0.866) koordinatında yer alır. Bu diferansiyel denklemin çözümünde başlangıç noktası olarak : (x,y,p,q)={0.44,0.866,-0.866,0.48} alabiliriz. Bu denklemi 0<t<200 zaman aralığı için ve m=0.98 n=0.02 alarak çözünüz.



(bolum8_046.jpg ,Resimaltı: Problem 32)

PROBLEM 33 Laminar doğal taşınım problemi Falkner–Skan çözümü dönüştürülmüş parametreler cinsinden aşağıda verilmiştir. Bu denklemlerdeki ψ akım fonksiyonu, η boyutsuz koordinattır. Gr Grashoff sayısı adını alan bir boyutsuz hız parametresidir.

$$\eta = \left(\frac{1}{4}Gr\right)^{1/4}\frac{y}{x} \qquad \psi = 4\nu\left(\frac{1}{4}Gr\right)^{1/4} f(\eta)$$

$$Gr = \frac{g(\rho_\infty/\rho_w - 1)x^3}{\nu^2}$$

Doğal taşınım sınır tabakası için hız profili:

$$w_x = \frac{\partial\psi}{dy} = \frac{4\nu}{x}\left(\frac{1}{4}Gr\right)^{1/2} f'(\eta) \quad w_y = -\frac{\partial\psi}{dx} = \frac{\nu}{x}\left(\frac{1}{4}Gr\right)^{1/4}\left[\eta f'(\eta) - 3f(\eta)\right]$$

x dikey boyut (dik duran bir levhaüzerindeki sınır tabaka akışı) , y yatay boyut, Gr ,Grashoff sayısı, w_x ve w_y x ve boyutlarındaki hızlar, ψ akım fonksiyonudur.

$\theta(\eta) = \frac{t - t_\infty}{t_w - t_\infty}$ boyutsuz sıcaklık parametresidir Bu şartlar altında Falkner_Scan çözümü aşağıdaki diferansiyel denklemlere dönüşür.

$$f'''(\eta) + 3f(\eta)f''(\eta) - 2(f'(\eta))^2 + \theta(\eta) = 0$$

$$\theta''(\eta) + 3\,\mathrm{Pr}\, f(\eta)\theta'(\eta) = 0$$

Burada Pr Prandtl sayısıdır. *Denklemin sınır koşulları:*

$$\eta = 0: \quad f(\eta) = f'(\eta) = 0 \quad \theta(\eta) = 1$$

$$\eta \to \infty: \quad f'(\eta) = 0 \qquad \theta(\eta) = 0$$

Olarak verilmiştir. Diferansiyel denklem sistemini çözünüz.

PROBLEM 34

Mg kütleli yıldızın çekim kuvveti altında hareket eden bir gezegenin hareketini incelemek istersek, Newton çekim yasasını kullanabiliriz. Diferansiyel denklem setimiz

$$\frac{d^2x}{dt^2} = -\frac{GMg}{r^3}x$$

$$\frac{d^2y}{dt^2} = -\frac{GMg}{r^3}y$$

$$\frac{dx}{dt} = v_x$$

$$\frac{dy}{dt} = v_y$$

$$r = \sqrt{x^2 + y^2}$$

Sınır şartlarımız (1ab=1.5x10^{11}m, GMg=40 ab^3/yıl^2)

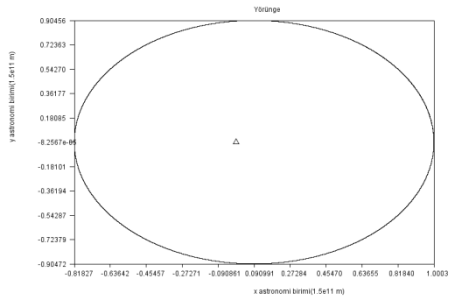
x(t=0 yıl)= 1 ab

y(t=0 yıl) = 0 ab

v_x(t=0 yıl) = 0 ab/yıl

v_y(t=0 yıl) = 6.0 ab/yıl olsun

Bu şartlar altında diferasiyel denklem setini çözünüz.



(bolum8_047.jpg ,Resimaltı: Problem 34)

9. Özdeğer - Sınır değer problemleri

9.0 Giriş

[A]{X} = 0 denkleminin sıfır olmayan çözümleri, yani bütün x'lerin 0 a eşit olmayan değerleri, mevcut olmakla birlikte bunlar tek değildir. Denklem

[A]{X} = [I]{λ} {X} =0

[A- λI]{X}=0 şeklinde de yazılabilir. Buradaki **λ** değerine özdeğer, sınırdeğer veya karekteristik değer (İngilizce : eigen-value) adı verilir. Açık formda yazarsak

$$\begin{bmatrix} a_{11}-\lambda & a_{12} & a_{13} & ... & a_{1n} \\ a_{11} & a_{22}-\lambda & a_{23} & ... & a_{2n} \\ a_{11} & a_{11} & a_{33}-\lambda & ... & a_{3n} \\ ... & ... & ... & .... & .... \\ a_{1n} & a_{2n} & a_{3n} & ... & a_{nn}-\lambda \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ ... \\ X_5 \end{bmatrix} = 0$$

bu çözüm setini veren X değerlerine de sınırdeğer vektörü (İngilizce : eigen-vector) adı verilir.

9.1 En büyük sınır değer (kuvvet metodu)

A n x n matris olsun. A matrisinin sınır değerlerini büyüklük sırasına göre sıralayalım
$|\lambda_1| > |\lambda_2| > |\lambda_3| > |\lambda_4| \ldots > |\lambda_{n-1}| > |\lambda_n|$
En büyük sınır değere dominant sınır değer de denilir. Eğer bir sınır değer tüm diğer sınır değerlerden büyük ise bu gerçek bir sınırdeğer olmalıdır ve gerçek sayılardan oluşan sınırdeğer vektörü bulunmalıdır. x^k □$_k$ sınır değerine karşı gelen sınır değer vektörü olsun.
$A\,x^k = \lambda_k x^k$, $1 <= k < n$.
$\{ x^1 , x^2 , x^3 , x^4 ,\ldots, x^n \}$ n sınır değer vektörü olsun. bu vektörün lineer olarak bağımsız bir set oluşturduğunu varsayalım, bu varsayım A simetrik matrisse ve aynı zamanda n birbirinden bağımsız sınır değeri varsa geçerlidir. Bu durumda n x 1 bir vektör y^0 n x 1 koordinat vektörü c sınır değer vektörü cinsinden

$$y^0 = \sum_{k=1}^{n} c_k x^k$$

denklemi ile ifade edilebilir.
şimdi y^k vektörlerinin bir serisini oluşturacak olursak

$y^{k+1}=A\,y^k,\ \ k \geq 0$

$$y^1 = A\left(\sum_{k=1}^{n} c_k x^k\right) = \sum_{k=1}^{n} c_k A x^k = \sum_{k=1}^{n} c_k \lambda_k\, x^k$$

denklemi genelleştirirsek

$$y^i = \sum_{k=1}^{n} c_k \lambda_k^i\, x^k,\ \ i \geq 0$$

λ_1 in en büyük sınır değer olduğunu hatırlayalım. Bu durumda denklemi λ_1 parantezine alırsak

$$y^i = \lambda_k^i \left[c_1 x^1 + \sum_{k=2}^{n} c_k \left(\frac{\lambda_k}{\lambda_1}\right)^i x^k \right],\ \ i \geq 0$$

Bu denklemde $\left|\lambda_k / \lambda_1\right| < 1$ dir. $i \to \infty$ iken $\left|\lambda_k / \lambda_1\right|^i \to 0$. Bundan dolay

$y^i \approx \lambda_1^i c_1 x^1\ ,\ \ i >> 1$

y vektörünü normalize edersek:

$$y^{k+1} = \frac{Ay^k}{\left\|Ay^k\right\|}$$

şeklini alır.

vektörü bilindiğinde sınır değere rahatlıkla geçebiliriz. Eğer x A matrisinin sınır değer vektörü ise,

A x = λ x veya her iki tarafı x^T ile çarparsak

x^T A x = λ x^T x bu durumda

$$\lambda = \frac{x^T A x}{x^T x}$$

k çok büyük olduğunda y^k nin dominant (en büyük) sınırdeğere yaklaştığını biliyoruz. Bu durumda yaklaşık olarak

$$\mu = \frac{(y^k)^T A y^k}{(y^k)^T y^k},\quad k \geq 0$$

yazabiliriz ve μ sınır değere yaklaşır.

μ nün sınır değerden sapmasının hata fonksiyonu

$$r^k \cong (\mu_k I - A) y^k$$

burada $r^k = 0$ olduğunda μ_k sınır değer ve y^k sınır değer vektörüne dönüşür. Bu yüzden $\left\|r_k\right\|$ yeterince küçülürse iterasyon prosesinde sınır değere ulaşmış oluruz.

Program 9.1.1 Kuvvet metodu ile en büyük sınırdeğerin bulunması

```
 import java.io.*;
import javax.swing.*;

class SCO13C
{
  //===========Sınır değer hesabı ==============
```

```
//*************norm methods definition*****************
public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}

  public static double norm(double v[],int p)
  {
   //p vector norm
  if(p==0) p=1;
  double total=0;
  double x,y;
  for(int i=0;i<v.length;i++)
    {
    x=Math.abs(v[i]);
    total+=Math.pow(x,p);
    }
  return Math.pow(total,(1.0/p));
  }
  public static double norminf(double v[])
    {
     //infinite vector norm
    double x;
    double max=0;
    for(int i=0;i<v.length;i++)
      {
      x=Math.abs(v[i]);
      if(x>max) max=x;
      }
    return max;
    }
   public static double norminf(double a[][])
        {
          //infinite matrix norm
          double x;
          double max = 0;
          double total;
        int i,j;

        int n=a.length;
        for(i=0;i<n;i++)
          {
          total=0;
          for(j=0;j<n;j++)
          {total+=Math.abs(a[i][j]);}
          x=total;
          if(x>max) max=x;
          }
```

```
      return max;
      }
  public static double norm(double a[][])
     {
     //matrix norm
            double x;
            double max = 0;
            double total;
            int i,j;
            int n=a.length;
            for(j=0;j<n;i++)
              {
              total=0;
              for(i=0;i<n;j++)
              {total+=Math.abs(a[i][j]);}
              x=total;
              if(x>max) max=x;
              }
            return max;
    }
  public static double normE(double a[][])
          {
          //Euclidian matrix norm
          double x;
          double total;
          int i,j;
          total=0;
          int n=a.length;
          for(j=0;j<n;i++)
          {for(i=0;i<n;j++) {total+=a[i][j]*a[i][j];}}
          return Math.sqrt(total);
         }
 public static double[] multiply(double[][] left,double[] right)
 {
 //multiplication of one matrix with one vector
 int ii,jj,i,j,k;
 int m1=left[0].length;
 int n1=left.length;
 int m2=right.length;
 double[] b;
 b=new double[m2];
 if(n1 != m2) {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
  {b[ii]=0;}
  return b;   }
for(i=0;i<m1;i++)
  { b[i]=0;
    for(k=0;k<n1;k++)
          b[i]+=left[i][k]*right[k];}
return b;
//end of multiply of a matrix and a vector
```

```
}

public static double  YT_X(double [] left,double [] right)
{
//multiplys a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
{
tot+=left[i]*right[i];
}
return tot;
}

public static double eigen_power(double A[][])
{
 double eps=1e-15;
 double err=1;
 int m=100;
 int n=A.length;
 double y[]=new double[n];
 double r[]=new double[n];
 double xnorm;
 double mu=1.0;
 for(int i=0;i<n;i++) y[i]=1.0;
 int k=0;
 double mueski;
 double x[]=new double[n];
 do{
            mueski=mu;
            x=multiply(A,y);
            xnorm=norminf(x);
            for(int i=0;i<n;i++) {y[i]=x[i]/xnorm;}
            mu=YT_X(y,x)/YT_X(y,y);
            for(int i=0;i<n;i++) {r[i]=mu*y[i]-x[i];}
            k++;
            err=Math.abs((mu-mueski)/mu);
   }while(err>eps);
 return mu;
}



public static void main(String args[]) throws IOException
{
//Eigen Value calculations using  method
//double a[][]={{3.556,-1.778,0},{-1.778,3.556,-1.778},{0,-1.778,3.556}};
//double a[][]={{-3.9,-0.6,-3.9,0.4},{1,0,0,0},{0,1,0,0},{0,0,1,0}};
double a[][]={{2,-3,6},{0,3,-4},{0,2,-3}};
String s="Orijinal Matris : \n";
s+=Matrix.toString(a);
s+="en büyük sınır değer = "+eigen_power(a);
```

```
  System.out.println("En büyük sınır değer metodu : \n"+s);
  }
}
```

Çıktı 8.3-1 Kuvvet metodu ile en büyük sınırdeğerin bulunması

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO13C
En büyük sınır değer metodu :
Orijinal Matris :
      2.000000000000000      -3.000000000000000       6.000000000000000
      0.000000000000000       3.000000000000000      -4.000000000000000
      0.000000000000000       2.000000000000000      -3.000000000000000
en büyük sınır değer = 1.9999999999999996

> Terminated with exit code 0.
```

Konumuzu bitirmeden önce detaylı çözülmüş örnek problemlere yer verelim

Örnek 1: Kuvvet metoduyla aşağıdaki metrisin en büyük sınır değerini bulunuz

$$A=\begin{bmatrix}1 & 2\\ 2 & 5\end{bmatrix}$$

Çözüm adımları aşağıda verilmiştir. Başlangıç vektörü olarak y={1,1} alınmıştır.

1	2	1	3	3
2	5	1	7	7
				7

1	2	0.428571	2.428571	2.428571
2	5	1	5.857143	5.857143
				5.857143

1	2	0.414634	2.414634	2.414634
2	5	1	5.829268	5.829268
				5.829268

1	2	0.414226	2.414226	2.414226
2	5	1	5.828452	5.828452
				5.828452

1	2	0.414214	2.414214	2.414214
2	5	1	5.828428	5.828428
				5.828428

1	2	0.414214	2.414214	2.414214
2	5	1	5.828427	5.828427
				5.828427

λ= 5.828427

Örnek 2 : Kuvvet metoduyla aşağıdaki metrisin en büyük sınır değerini bulunuz

$$A=\begin{bmatrix} 10 & 4 & -6 \\ 4 & -6 & 8 \\ -6 & 8 & 14 \end{bmatrix}$$ bu örnek problemde de başlangıç vektörü olarak y={1,1,1} vektörü alınmıştır.

3.556	-1.778	0	1
-1.778	3.556	-1.778	1
0	-1.778	3.556	1

1.778	1.778	
0	0	
1.778	1.778	
	1.778	

3.556	-1.778	0	1
-1.778	3.556	-1.778	0
0	-1.778	3.556	1

3.556	3.556	
-3.556	3.556	
3.556	3.556	
3.556	3.556	1

3.556	-1.778	0	1
-1.778	3.556	-1.778	-1
0	-1.778	3.556	1

5.334	5.334	
-7.112	7.112	
5.334	5.334	
5.334	7.112	1

3.556	-1.778	0	0.75
-1.778	3.556	-1.778	-1
0	-1.778	3.556	0.75

4.445	4.445	
-6.223	6.223	
4.445	4.445	
4.445	6.223	-0.125

3.556	-1.778	0	0.714286
-1.778	3.556	-1.778	-1
0	-1.778	3.556	0.714286

4.318	4.318	
-6.096	6.096	
4.318	4.318	
4.318	6.096	-0.020408163

3.556	-1.778	0	0.708333
-1.778	3.556	-1.778	-1
0	-1.778	3.556	0.708333

4.296833	4.296833	
-6.07483	6.074833	
4.296833	4.296833	
4.296833	6.074833	-0.003472222

3.556	-1.778	0	0.707317
-1.778	3.556	-1.778	-1
0	-1.778	3.556	0.707317

4.29322	4.29322	
-6.07122	6.07122	
4.29322	4.29322	
4.29322	6.07122	-0.000594884

3.556	-1.778	0	0.707143
-1.778	3.556	-1.778	-1
0	-1.778	3.556	0.707143

4.2926	4.2926	
-6.0706	6.0706	
4.2926	4.2926	
4.2926	6.0706	-0.000102041

3.556	-1.778	0	0.707113
-1.778	3.556	-1.778	-1
0	-1.778	3.556	0.707113

4.292494	4.292494	
-6.07049	6.070494	
4.292494	4.292494	
4.292494	6.070494	-1.75067E-05

3.556	-1.778	0	0.707108

4.292475	4.292475

-1.778	3.556	-1.778	-1	-6.07048	6.070475	
0	-1.778	3.556	0.707108	4.292475	4.292475	
				4.292475	6.070475	-3.00365E-06

3.556	-1.778	0	0.707107	4.292472	4.292472	
-1.778	3.556	-1.778	-1	-6.07047	6.070472	
0	-1.778	3.556	0.707107	4.292472	4.292472	
				4.292472	6.070472	-5.15345E-07

3.556	-1.778	0	0.707107	4.292472	4.292472	
-1.778	3.556	-1.778	-1	-6.07047	6.070472	
0	-1.778	3.556	0.707107	4.292472	4.292472	
				4.292472	6.070472	-8.84191E-08

λ= 6.070472

9.2 En küçük sınır değer(kuvvet metodu)

En büyük sınır değer (özdeğer) bulma işleminin tersini uygulayarak (ters matrisin en büyük sınır değerini bularak) en küçük sınır değeri bulabiliriz.

$$y^k = \frac{A^{-1}y^{k-1}}{\left\|A^{-1}y^{k-1}\right\|}$$

$$\mu_k = \frac{(y^k)^T A^{-1} y^k}{(y^k)^T y^k}$$

Program 9.2.1 En küçük sınır değer bulunması

```
 import java.io.*;
import javax.swing.*;

class SCO13D
{
  //===========Eigen value calculations ==============

//*************norm methods definition*****************
public static double norm(double v[])
{
// vector norm
double total=0;;
for(int i=0;i<v.length;i++)
  {total+=v[i]*v[i];}
return Math.sqrt(total);
}

  public static double norm(double v[],int p)
  {
   //p vector norm
  if(p==0) p=1;
  double total=0;
  double x,y;
  for(int i=0;i<v.length;i++)
    {
    x=Math.abs(v[i]);
    total+=Math.pow(x,p);
    }
  return Math.pow(total,(1.0/p));
  }
```

```
public static double norminf(double v[])
  {
   //infinite vector norm
  double x;
  double max=0;
  for(int i=0;i<v.length;i++)
    {
    x=Math.abs(v[i]);
    if(x>max) max=x;
    }
  return max;
  }
 public static double norminf(double a[][])
     {
       //infinite matrix norm
       double x;
       double max = 0;
       double total;
     int i,j;

     int n=a.length;
     for(i=0;i<n;i++)
       {
       total=0;
       for(j=0;j<n;j++)
       {total+=Math.abs(a[i][j]);}
       x=total;
       if(x>max) max=x;
       }
     return max;
     }
 public static double norm(double a[][])
    {
    //matrix norm
            double x;
            double max = 0;
            double total;
            int i,j;
            int n=a.length;
            for(j=0;j<n;i++)
              {
              total=0;
              for(i=0;i<n;j++)
              {total+=Math.abs(a[i][j]);}
              x=total;
              if(x>max) max=x;
              }
            return max;
   }
 public static double normE(double a[][])
          {
          //Euclidian matrix norm
          double x;
          double total;
          int i,j;
          total=0;
          int n=a.length;
          for(j=0;j<n;i++)
          {for(i=0;i<n;j++) {total+=a[i][j]*a[i][j];}}
          return Math.sqrt(total);
         }
public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int m1=left[0].length;
```

```
 int n1=left.length;
 int m2=right.length;
 double[] b;
 b=new double[m2];
 if(n1 != m2) {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
  {b[ii]=0;}
  return b;   }
for(i=0;i<m1;i++)
  { b[i]=0;
    for(k=0;k<n1;k++)
                b[i]+=left[i][k]*right[k];}
return b;
//end of multiply of a matrix and a vector
}

 public static double  YT_X(double [] left,double [] right)
 {
 //multiplys a vector transpose with a vector
 int n=left.length;
 double tot=0;
 for(int i=0;i<n;i++)
 {
 tot+=left[i]*right[i];
 }
 return tot;
 }

 public static double eigen_power(double A[][])
 {
  double eps=1e-15;
  double err=1;
  int m=100;
  int n=A.length;
  double y[]=new double[n];
  double r[]=new double[n];
  double xnorm;
  double mu=1.0;
  for(int i=0;i<n;i++) y[i]=1.0;
  int k=0;
  double mueski;
  double x[]=new double[n];
  do{
                    mueski=mu;
                    x=multiply(A,y);
                    xnorm=norminf(x);
                    for(int i=0;i<n;i++) {y[i]=x[i]/xnorm;}
                    mu=YT_X(y,x)/YT_X(y,y);
                    for(int i=0;i<n;i++) {r[i]=mu*y[i]-x[i];}
                    k++;
                    err=Math.abs((mu-mueski)/mu);
    }while(err>eps);
  return mu;
 }
   public static void pivot(double a[][],int o[],double s[],int k)
 {// GaussLU için pivot metodu
   int n=a.length;
   int dummy1=0;
   double buyuk;
   double dummy=0;
   //pivotlama
               int p=k;
               buyuk=Math.abs(a[o[k]][k]);
               for(int ii=k+1;ii<n;ii++)
               {  dummy=Math.abs(a[o[ii]][k]/s[o[ii]]);
```

```
                if(dummy > buyuk) {buyuk=dummy;p=ii;}
              }
              dummy1=o[p];
              o[p]=o[k];
              o[k]=dummy1;
}

public static int[] sirali_indeks(int n)
{   // sıralı rakamları taşıyan bir vektör oluşturur
               int o[]=new int[n];
               for(int i=0;i<n;i++) {o[i]=i;}
               return o;
}

public static double[][] gaussLU(double ai[][],int o[])
{ //kısmi pivotlu gauss eleme yöntemi
int n=ai.length;
double a[][]=new double[n][n];
double s[]=new double[n];
double carpan=0;
double toplam=0;
double absaij;

//gauss eleme
int i,j,k,p,ii,jj;
//System.out.println("Orijinal Matris :\n"+Matrix.toString(ai));
for(i=0;i<n;i++)
{for(j=0;j<n;j++){a[i][j]=ai[i][j];}}
for(i=0;i<n;i++)
{ o[i]=i;s[i]=Math.abs(a[i][0]);
  for(j=1;j<n;j++)
    { absaij=Math.abs(a[i][j]);
      if(absaij>s[i]) {s[i]=absaij;}
    }
}
for(k=0;k<(n-1);k++)
{ pivot(a,o,s,k);
              //pivot referansını değiştir
              //System.out.println("pivotlu Matris : k="+k+"\n"+Matrix.toString(a));
              //gauss elemeyi çözme
  for(i=k+1;i<n;i++)
  {  carpan=a[o[i]][k]/a[o[k]][k];
     a[o[i]][k]=carpan;
     for(j=k+1;j<n;j++)
     {   a[o[i]][j]-=carpan*a[o[k]][j]; }
              }
}
return a;
}

public static double[] yerine_koyma(double ai[][],double bi[])
{   int n=ai.length;
               return yerine_koyma(ai,bi,sirali_indeks(n));
}

public static double[][] yerine_koyma(double ai[][],double bi[][])
{   int n=ai.length;
               return yerine_koyma(ai,bi,sirali_indeks(n));
}

public static double[] yerine_koyma(double ai[][],double bi[],int o[])
{ //gauss LU çözümüyle LU indirgenmesi yapılmış Matris kullanarak lineer
  //denklem sistemi kökleri
  int n=ai.length;
  double toplam;
  double x[]=new double[n];
```

```
    double a[][]=new double[n][n];
   double b[]=new double[n];
   int i,j;
for(i=0;i<n;i++)
    {for(j=0;j<n;j++){a[i][j]=ai[i][j];};b[i]=bi[i];}
//System.out.println("a=\n "+Matrix.toString(a));
//System.out.println("b=\n "+Matrix.toStringT(b));
for(i=1;i<n;i++)
    {toplam=b[o[i]];for(j=0;j<i;j++){toplam-=a[o[i]][j]*b[o[j]];};b[o[i]]=toplam;}
 //System.out.println("d=\n "+Matrix.toStringT(b));
x[n-1]=b[o[n-1]]/a[o[n-1]][n-1];
for(i=(n-2);i>=0;i--)
    {toplam=0;
     for(j=i+1;j<n;j++)
      {toplam+=a[o[i]][j]*x[j];};
      x[i]=(b[o[i]]-toplam)/a[o[i]][i];
    }
//System.out.println("d=\n "+Matrix.toStringT(x));
return x;
}

public static double[][] yerine_koyma(double ai[][],double bi[][],int o[])
{
  int n=ai.length;
  int m=bi[0].length;
  double toplam;
  double x[][]=new double[n][m];
  double a[][]=new double[n][n];
  double b[][]=new double[n][m];
  int i,j,k;
for(i=0;i<n;i++)
    {for(j=0;j<n;j++){a[i][j]=ai[i][j];};for(k=0;k<m;k++) {b[i][k]=bi[i][k];}}
//System.out.println("a=\n "+Matrix.toString(a));
//System.out.println("b=\n "+Matrix.toStringT(b));
for(k=0;k<m;k++)
{
for(i=1;i<n;i++)
    {toplam=b[o[i]][k];for(j=0;j<i;j++){toplam-=a[o[i]][j]*b[o[j]][k];};b[o[i]][k]=toplam;}
 //System.out.println("d=\n "+Matrix.toStringT(b));
x[n-1][k]=b[o[n-1]][k]/a[o[n-1]][n-1];
for(i=(n-2);i>=0;i--)
    {toplam=0;
     for(j=i+1;j<n;j++)
      {toplam+=a[o[i]][j]*x[j][k];};
      x[i][k]=(b[o[i]][k]-toplam)/a[o[i]][i];
    }
}//kin sonu
//System.out.println("d=\n "+Matrix.toStringT(x));
return x;
}

public static double [] AXB(double a[][],double b[])
{
//denklem sistemi çözümü b vektörü için
int n=a.length;
int q[]=new int[n];
double c[][]=gaussLU(a,q);
double s[]=yerine_koyma(c,b,q);
return s;
}

public static double [][] AXB(double a[][],double b[][])
{//denklem sistemi çözümü b matrisi için
int n=a.length;
int q[]=new int[n];
double c[][]=gaussLU(a,q);
```

```
 double s[][]=yerine_koyma(c,b,q);
 return s;
 }

 public static double eigen_inversepower(double A[][])
 {
  double eps=1e-15;
  double err=1;
  int m=100;
  int n=A.length;
  double y[]=new double[n];
  double r[]=new double[n];
  int o[]=new int[n];
  double B[][]=new double[n][n];
  double xnorm;
  double mu=1.0;
  for(int i=0;i<n;i++) y[i]=1.0;
  int k=0;
  double mueski;
  double x[]=new double[n];
  do{
                mueski=mu;
                B=gaussLU(A,o);
                x=yerine_koyma(B,y,o);
                xnorm=norminf(x);
                for(int i=0;i<n;i++) {y[i]=x[i]/xnorm;}
                x=yerine_koyma(B,y,o);
                mu=YT_X(y,x)/YT_X(y,y);
                for(int i=0;i<n;i++) {r[i]=mu*y[i]-x[i];}
                System.out.println("mu="+mu+"k="+k+"\n"+Matrix.toString(y));
                k++;
                err=Math.abs((mu-mueski)/mu);
    }while(err>eps);
  return 1.0/mu;
 }


 public static void main(String args[]) throws IOException
 {
 //Eigen Value calculations using  method
 double a[][]={{3.556,-1.778,0},{-1.778,3.556,-1.778},{0,-1.778,3.556}};
 //double a[][]={{-3.9,-0.6,-3.9,0.4},{1,0,0,0},{0,1,0,0},{0,0,1,0}};
 //double a[][]={{-1,2},{2,2}};
 //double a[][]={{1,-2,3},{-2,4,1},{3,1,2}};
 String s="Orijinal Matris : \n";
 s+=Matrix.toString(a);
 s+="en küçük sınır değer = "+eigen_inversepower(a);
 System.out.println("En küçük sınır değer metodu : \n"+s);
  }
}
```

Çıktı 8.4-1 En küçük sınır değer bulunması

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO13D
mu=0.9594388936677033k=0
      0.750000000000000         1.000000000000000         0.750000000000000

mu=0.9601072593198579k=1
      0.714285714285714         1.000000000000000         0.714285714285714

mu=0.9601269512021569k=2
      0.708333333333333         1.000000000000000         0.708333333333333
```

```
mu=0.960127530891823k=3
      0.707317073170732         1.000000000000000         0.707317073170732

mu=0.9601275479563067k=4
      0.707142857142857         1.000000000000000         0.707142857142857

mu=0.960127548458638k=5
      0.707112970711297         1.000000000000000         0.707112970711297

mu=0.9601275484734253k=6
      0.707107843137255         1.000000000000000         0.707107843137255

mu=0.9601275484738608k=7
      0.707106963388370         1.000000000000000         0.707106963388370

mu=0.9601275484738734k=8
      0.707106812447435         1.000000000000000         0.707106812447435

mu=0.9601275484738739k=9
      0.707106786550068         1.000000000000000         0.707106786550068

En küçük sınır değer metodu :
Orijinal Matris :
      3.556000000000000        -1.778000000000000         0.000000000000000
     -1.778000000000000         3.556000000000000        -1.778000000000000
      0.000000000000000        -1.778000000000000         3.556000000000000
en küçük sınır değer = 1.0415282861006367

> Terminated with exit code 0.
```

Örnek problem1: $A=\begin{bmatrix}1 & 2\\ 2 & 5\end{bmatrix}$ Matrisinin en küçük sınır değerini ters kuvvet metoduyla bulunuz. A matrisinin tesr matrisi:

$$A^{-1}=\begin{bmatrix}5 & -2\\ -2 & 1\end{bmatrix}$$

5	-2	1	3	3
-2	1	1	-1	1
				3

5	-2	1	5.666667	5.666667
-2	1	-0.33333	-2.33333	2.333333
				5.666667

5	-2	1	5.823529	5.823529
-2	1	-0.41176	-2.41176	2.411765
				5.823529

5	-2	1	5.828283	5.828283
-2	1	-0.41414	-2.41414	2.414141
				5.828283

5	-2	1	5.828423	5.828423
-2	1	-0.41421	-2.41421	2.414211
				5.828423

5	-2	1	5.828427	5.828427
-2	1	-0.41421	-2.41421	2.414213
				5.828427

λ=1/5.828427= 0.171573

Örnek problem 2: $A=\begin{bmatrix} 3.556 & -1.778 & 0 \\ -1.778 & 3.556 & -1.778 \\ 0 & -1.778 & 3.556 \end{bmatrix}$ Matrisinin en küçük sınır değerini ters kuvvet metoduyla bulunuz. A matrisinin tesr matrisi:

$$A^{-1}=\begin{bmatrix} 0.42182227 & 0.281215 & 0.14060 \\ 0.28121484 & 0.56243 & 0.28121 \\ 0.14060742 & 0.281215 & 0.42182 \end{bmatrix}$$

0.421822272	0.281215	0.140607	1	0.843645	0.843645	
0.281214848	0.56243	0.281215	1	1.124859	1.124859	
0.140607424	0.281215	0.421822	1	0.843645	0.843645	
					1.124859	

0.421822272	0.281215	0.140607	0.75	0.703037	0.703037	
0.281214848	0.56243	0.281215	1	0.984252	0.984252	
0.140607424	0.281215	0.421822	0.75	0.703037	0.703037	
				0.984252	0.984252	-0.125

0.421822272	0.281215	0.140607	0.714286	0.68295	0.68295	
0.281214848	0.56243	0.281215	1	0.964165	0.964165	
0.140607424	0.281215	0.421822	0.714286	0.68295	0.68295	
				0.964165	0.964165	-0.02041

0.421822272	0.281215	0.140607	0.708333	0.679603	0.679603	
0.281214848	0.56243	0.281215	1	0.960817	0.960817	
0.140607424	0.281215	0.421822	0.708333	0.679603	0.679603	
				0.960817	0.960817	-0.00347

0.421822272	0.281215	0.140607	0.707317	0.679031	0.679031	
0.281214848	0.56243	0.281215	1	0.960246	0.960246	
0.140607424	0.281215	0.421822	0.707317	0.679031	0.679031	
				0.960246	0.960246	-0.00059

0.421822272	0.281215	0.140607	0.707143	0.678933	0.678933	
0.281214848	0.56243	0.281215	1	0.960148	0.960148	
0.140607424	0.281215	0.421822	0.707143	0.678933	0.678933	
				0.960148	0.960148	-0.0001

0.421822272	0.281215	0.140607	0.707113	0.678916	0.678916

0.281214848	0.56243	0.281215	1	0.960131	0.960131	
0.140607424	0.281215	0.421822	0.707113	0.678916	0.678916	
				0.960131	0.960131	-1.8E-05

0.421822272	0.281215	0.140607	0.707108	0.678913	0.678913	
0.281214848	0.56243	0.281215	1	0.960128	0.960128	
0.140607424	0.281215	0.421822	0.707108	0.678913	0.678913	
				0.960128	0.960128	-3E-06

0.421822272	0.281215	0.140607	0.707107	0.678913	0.678913	
0.281214848	0.56243	0.281215	1	0.960128	0.960128	
0.140607424	0.281215	0.421822	0.707107	0.678913	0.678913	
				0.960128	0.960128	-5.2E-07

0.421822272	0.281215	0.140607	0.707107	0.678913	0.678913	
0.281214848	0.56243	0.281215	1	0.960128	0.960128	
0.140607424	0.281215	0.421822	0.707107	0.678913	0.678913	
				0.960128	0.960128	-8.8E-08

0.421822272	0.281215	0.140607	0.707107	0.678913	0.678913	
0.281214848	0.56243	0.281215	1	0.960128	0.960128	
0.140607424	0.281215	0.421822	0.707107	0.678913	0.678913	
				0.960128	0.960128	-1.5E-08

0.421822272	0.281215	0.140607	0.707107	0.678913	0.678913	
0.281214848	0.56243	0.281215	1	0.960128	0.960128	
0.140607424	0.281215	0.421822	0.707107	0.678913	0.678913	
				0.960128	0.960128	-2.6E-09

λ=1/0.960128= 1.041528

9.3 Jacobi metodu ve Householder dönüşümü

Jacobi Metodu simetrik matrislerin sınır değer problemini çözmek için kulanılan bir metoddur. Simetrik matrisler için matrisin transpozu kendine eşittir.

$\{A\}^T=\{A\}$

Simetrik matrislerin bu özelliğini kullanarak benzerlik dönüşümü adı verdiğimiz bir dönüşüm prosesi ile matrisi diagonal form adını verdiğimiz forma dönüştürebiliriz. Bu form yardımı ile de sınır değerlerini hesaplarız.
P matrisinin tekil olmayan bir matris olduğunu varsayalım. Benzerlik dönüşümü

{B}={P}-1{A}{ P}

şeklinde yazılabilir. Benzerlik dönüşümü sonucu sınır değerler değişmez, fakat sınır değer vektörleri değişir. Bu yüzden B matrisinin sınır değerleri A matrisinin sınır değerleri ile aynıdır. Şimdi $[y]^k$ vektörünün {B} matrisinin λ_k sınır değerine karşı gelen sınır değer vektörü olduğunu varsayalım. Bu durumda

$\{B\}[y]^k = \lambda_k [y]^k$ olur.
$\{B\}=\{P\}^{-1}\{A\}\{ P\}$ denkleminin her iki tarafını da { P} ile çarpacak olursak :
$\{ P\} \{B\}=\{ P\} \{P\}^{-1}\{A\}\{ P\}=\{A\}\{ P\}$ olduğunu görürüz. Buradan
$\{A\}\{ P\}[y]^k = \{ P\} \{B\}[y]^k= \lambda_k\{ P\} [y]^k$ olur. Eğer $[y]^k$ {B} matrisinin sınırdeğer vektörü ise, {A} matrisinin sınırdeğer vektörü de $[x]^k$ gibi bir değer alacaktır. Bu durumda $[y]^k$ ve $[x]^k$ arasında

$[x]^k=\{P\}[y]^k$ bağıntısı bulunmalıdır. Yani {A} matrisinin sınırdeğer vektörleri {B} matrisinin sınır değer vektörlerinin {P} matrisiyle çarpımından elde edilebilir. Ortagonal benzerlik dönüşümü uygulanarak ($\{P\}^{-1}=\{P\}^T$) A simetrik matrisi diagonal forma dönüştürülebilir. Bu dönüşümü sağlamak için açısal döndürme matrisi R(ϕ) kullanılabilir. R(ϕ) matrisi p satırı ve q sütunu hariç birim matrise eşittir.

$R_{pp}=R_{qq} = \cos(\phi)=c$
$R_{pq}= - R_{qp}= - \sin(\phi)=-s$

Örneğin n=8 p=3 q=5 için

$$R(\phi)=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & R_{pp} & 0 & R_{pq} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & R_{qp} & 0 & R_{qq} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos(\phi) & 0 & -\sin(\phi) & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sin(\phi) & 0 & \cos(\phi) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Formunu alır.
Burada ϕ dönme açısıdır.Dönme matrisi R'nin ilginç özellikleri mevcuttur. Bunlardan birisi de transpose matris ile çarpıldığında birim matrise eşit olmasıdır.

$\{R(\phi)\}^T\{R(\phi)\}=I$

Aynı zamanda ters matris hesabı çok kolaydır.
$\{R(\phi)\}^{-1}=\{R(\phi)\}^T$

bu denklem de

$\{R(\phi)\}^T=\{R(-\phi)\}$

Eğer benzerlik dönüşümü R matrisi kullanılarak uygulanır ise

$\{B\}=\{R(\phi)\}^T\{A\}\{R(\phi)\}$

bu transformda ϕ açısını öyle seçebiliriz ki $B_{qp}=0$ olsun.

$B_{qp}=(A_{qq} - A_{pp}) \sin(\phi)\cos(\phi)+A_{pq}[\cos^2(\phi) - \sin^2(\phi)] = 0$

Eğer $\sin(2\phi)=2\sin(\phi)\cos(\phi)$

$\cos(2\phi)=\cos^2(\phi) - \sin^2(\phi)$ olduğunu hatırlarsak

$B_{qp}=(A_{qq} - A_{pp}) \sin(2\phi)/2+A_{pq}\cos(2\phi) = 0$

$\phi = \frac{1}{2}\arctan\left\{\frac{2A_{pq}}{A_{pp}-A_{qq}}\right\}$ olarak bulunur.

Bu açıyı kullanan R matrisini kullanarak benzerlik dönüşümü gerçekleştirilir.

Jacobi metodu şu şekilde uygulanır :

P=I seçilir. Diagonalin üst üçgenin düşen bölgede mutlak değeri en büyük olan A_{pq} elemanı bulunur. P ve q değerleri kullanarak ϕ açısı hesaplanır. R matrisi hesaplanır. {P} ={P}{R} dönüşümü ve {A} ={R}^{T}{A} {R} dönüşümü uygulanır sonra Diagonalin üst üçgenin düşen bölgede mutlak değeri en büyük olan A_{pq} elemanı bulunur, bu işlem iteratif olarak A_{pq} nun mutlak değeri 0 a yakın bir ε değerine ulaşıncaya kadar devam edilir. Sınır değerleri $\lambda_k=A_{kk}$ alınır, sınır değer vektörü de P=[x^1,x^2,…,x^n] olarak alınır.

Jacobi metodunun temel problemi A_{pq} elemanı sıfıra yaklaşıncaya kadar çok fazla iterasyon stepine ihtiyaç duymasıdır. Eğer A matrisini yukarıdaki işleme tabi tutmadan önce bir ön değişimden geçirir isek bu işlem hızlanabilir. Bu hızlanma işlemi için Matris üst Hesenberg matrisi denilen bir forma dönüştürülür.

$$B=\begin{bmatrix} B_{11} & B_{12} & B_{13} & \dots & B_{1,n-1} & B_{1,n} \\ B_{21} & B_{22} & B_{23} & \dots & B_{2,n-1} & B_{2,n} \\ 0 & B_{32} & B_{33} & \dots & B_{3,n-1} & B_{3,n} \\ 0 & 0 & B_{43} & \dots & B_{4,n-1} & B_{4,n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & B_{n,n-1} & B_{n,n} \end{bmatrix}$$

Şekilden de görüleceği gibi bu formda matrisin üst üçgeni ve diagonalin altındaki satırlar mevcuttur. Bu dönüşüm işlemi Householder dönüşüm işlemi adını alır. Dönüşüm işlemi şu steplerden oluşur :

1. Q=I alınır (I nxn birim matristir)
   k döngüsü 1den n-2 e kadar

   2. $\alpha = sign(A_{k+1,k})\sqrt{\sum_{j=k+1}^{n} A_{j,k}^2}$ buradaki $sign(A_{k+1,k})$, $A_{k+1,k}$ nin işaretidir (-1 veya +1). $sign(x)=\frac{x}{|x|}$
   2. u^k=[0,…,0,$A_{k+1,k}$ + α, $A_{k+2,k}$,….., $A_{n,k}$]^T olarak oluşturulur.
   3. $P = I - \frac{2uu^T}{u^T u}$
   4. Q = Q P çarpımı uygulanır.
   5. A=PAP^T çarpımı uygulanır.

6. B=A üst Hesenberg formuna dönüşmüş olur.

İlk örnek olarak küçük bir matris için Jacobi metodu kullanarak sınır değerleri elde hesaplayalım. $A=\begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix}$ olsun.

n=2 için Jacobi metodu iterasyona gerek bırakmadan bir adımda sonuca ulaşır. Önce faz değişim açısını hesaplayalım:

$\phi = \frac{1}{2}\arctan\left\{\frac{2(2)}{-1-2}\right\}$ =-1.1071487177940904 radyan . Bilgisayar sistemlerinde tanjant fonksiyonunun tersini tam doğru açı ile hesaplamak istersek atan2 fonksiyonlarını kullanmalıyız. Açının tam doğru olmaması buradaki hesabımızı temel olarak etkilemeyecektir (atan fonksiyonunda açı yerine aynı açının konjüge değeri gelebilir)

$$R=\begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}=\begin{bmatrix} 0.447213595499958 & -0.894427190999916 \\ -0.894427190999916 & 0.447213595499958 \end{bmatrix}$$

$A = R^T AR = \begin{bmatrix} 3 & 0 \\ 0 & -2 \end{bmatrix}$ olarak bulunur.

İkinci örneğimizde problemimizi program kullanarak çözelim

$$[A] = \begin{bmatrix} 12 & 6 & -6 \\ 6 & 16 & 2 \\ -6 & 2 & 16 \end{bmatrix}$$

Program 9.3.1 Jacobi ve House holder dönüşümü ile simetrik matrislerin gerçek sınır değerlerinin hesaplanması

```
 import java.io.*;
import javax.swing.*;

class SCO13E
{  // Jacobi eigenvalue calculation method
public static double[][] I(int n)
{
//unit matrix
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
   b[i][i]=1.0;
return b;
}

public static double[][] multiply(double[][] left,double right)
{
//multiplying a matrix with a constant
int i,j;
int n=left.length;
int m=left[0].length;
double b[][];
b=new double[n][m];
for(i=0;i<n;i++)
  {
  for(j=0;j<m;j++)
    b[i][j]=left[i][j]*right;
  }
return b;
//end of multiplying a matrix with a constant double
}

public static double[][] multiply(double[][] left,double[][] right)
{
//multiplication of two matrices
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right[0].length;
int n2=right.length;
double[][] b;
b=new double[m1][n2];
 if(n1 != m2)
 {
 System.out.println("inner matrix dimensions must agree");
 for(ii=0;ii<n1;ii++)
   {
   for(jj=0;jj<m2;jj++)
     b[ii][jj]=0;
   }
   return b;
 }
 for(i=0;i<m1;i++)
   {
   for(j=0;j<n2;j++)
```

```
    {
    for(k=0;k<n1;k++)
                b[i][j]+=left[i][k]*right[k][j];
    }
   }
return b;
//end of multiply of two matrices
}

public static double[][] substract(double[][] left,double[][] right)
{
//addition of two matrices
int n1=left.length;
int m1=left[0].length;
int n2=right.length;
int m2=right[0].length;
int nMax,mMax;
int i,j;
if(m1>=m2) mMax=m1;
else       mMax=m2;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[][];
b=new double[nMax][mMax];
for(i=0;i<n1;i++)
  {
  for(j=0;j<m1;j++)
    {
    b[i][j]=b[i][j]+left[i][j];
    }
  }
for(i=0;i<n2;i++)
  {
  for(j=0;j<m2;j++)
    {
    b[i][j]=b[i][j]-right[i][j];
    }
  }
return b;
//end of matrix substraction method
}

public static double[][] T(double [][] left)
{
//transpose matrix (if A=a(i,j) T(A)=a(j,i)
int i,j;
int n=left.length;
int m=left[0].length;
double b[][];
b=new double[m][n];
for(i=0;i<n;i++)
  {
  for(j=0;j<m;j++)
    {
    b[j][i]=left[i][j];
    }
  }
return b;
}

public static double VT_V(double [] left)
{
//multiply a vector transpose with a vector
int n=left.length;
double tot=0;
for(int i=0;i<n;i++)
```

```
  {
  tot+=left[i]*left[i];
  }
return tot;
}

public static double[][] V_VT(double [] left)
{
//multiplys a vector transpose with a vector
int n=left.length;
double aa[][]=new double[n][n];
for(int i=0;i<n;i++)
  {
  for(int j=0;j<n;j++)
    {
                aa[i][j]=left[i]*left[j];
                }
  }
return aa;
}

  public static double[] eigen_jacobi(double A[][])
  {
   A=Householder(A);
   double eps=1e-15;
   double err=1;
   int m=100;
   int n=A.length;
   double P[][]=new double[n][n];
   double R[][]=new double[n][n];
   int p=n,q=n;
   double max=-9.99e100;
   double fi;
   P=I(n);
   for(int i=0;i<n;i++)
   {for(int j=i+1;j<n;j++)
      {if(Math.abs(A[i][j]) > max) {max=Math.abs(A[i][j]);p=i;q=j;}}
   }
   int counter=0;
   while(Math.abs(A[p][q]) > eps )
   {
    fi=0.5*Math.atan2((2.0*A[p][q]),((A[p][p]-A[q][q])));
    System.out.println("fi="+fi);
    R=I(n);
    R[p][p]=Math.cos(fi);
    R[q][q]=Math.cos(fi);
    R[p][q]=-Math.sin(fi);
    R[q][p]=Math.sin(fi);
    System.out.println("counter="+counter+"R=\n"+Matrix.toString(R));
    counter++;
    P=multiply(P,R);
    //System.out.println("P=\n"+Matrix.toString(P));
    A=multiply(T(R),multiply(A,R));
    System.out.println("A=\n"+Matrix.toString(A));
    max=-9.99e100;
    for(int i=0;i<n;i++)
   {for(int j=i+1;j<n;j++){if(Math.abs(A[i][j])>max) {max=Math.abs(A[i][j]);p=i;q=j;}}}
    }
   double lambda[]=new double[n];
   for(int i=0;i<n;i++) {lambda[i]=A[i][i];}
   return lambda;
  }

  public static double sgn(double x)
  {return x/Math.abs(x);}
```

```
 public static double[][] Householder(double A[][])
 { //Householder transformation
  int n=A.length;
  double Q[][]=I(n);
  double alpha;
  double toplam;
  double u[]=new double[n];
  double C[][]=new double[n][n];
  double P[][]=new double[n][n];
  double B[][]=new double[n][n];
  double cc;
  for(int k=0;k<(n-2);k++)
  {
    toplam=0;
    for(int j=k+1;j<n;j++) {toplam+=A[j][k]*A[j][k];}
    toplam=Math.sqrt(toplam);
    alpha=sgn(A[k+1][k])*toplam;
    for(int j=0;j<n;j++)
    {   if(j<=k) u[j]=0.0;
        else if(j==k+1) u[j]=A[j][k]+alpha;
        else u[j]=A[j][k];
    }
    C=V_VT(u);
    cc=2.0/VT_V(u);
    C=multiply(C,cc);
    P=substract(I(n),C);
    Q=multiply(Q,P);
    A=multiply(P,multiply(A,P));
  }
  return A;
  }


 public static void main(String args[]) throws IOException
 {
 //Sınır değer hesapları
 double c[];
 c=new double[2];
 double a[][]={{12,6,-6},{6,16,2},{-6,2,16}};
 //double a[][]={{4,-2,6,4},{-2,2,-1,3},{6,-1,22,13},{4,3,13,46}};
 //double a[][]={{-1,2},{2,2}};
 String s="Orijinal matris : \n";
 s+=Matrix.toString(a);

 //c=eigen_jacobi(a);
 s+="Householder dönüşümü uygulanmış matris \n";
 s+=Matrix.toStringT(Householder(a),20,10);
 c=eigen_jacobi(a);
 s+="sınır değer vektörü \n";
 s+=Matrix.toStringT(c);
 System.out.println("Householder dönüşümü : \n"+s);
 }
}
```

Jacobi ve House holder dönüşümü ile simetrik matrislerin gerçek sınır değerlerinin hesaplanması

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO13E
fi=-0.844053176004745
counter=0
R=
       0.664439181868389       0.747342340295306       0.000000000000000
      -0.747342340295306       0.664439181868389       0.000000000000000
       0.000000000000000       0.000000000000000       1.000000000000000
```

```
A=
      21.544003745317536       0.000000000000000       0.000000000000002
       0.000000000000000       4.455996254682470      -0.000000000000001
       0.000000000000002      -0.000000000000001      18.000000000000000

fi=5.411075602359009E-16
counter=1R=
       1.000000000000000       0.000000000000000      -0.000000000000001
       0.000000000000000       1.000000000000000       0.000000000000000
       0.000000000000001       0.000000000000000       1.000000000000000

A=
      21.544003745317536       0.000000000000000       0.000000000000000
      -0.000000000000000       4.455996254682470      -0.000000000000001
       0.000000000000000      -0.000000000000001      18.000000000000000

Householder dönüşümü :
Orijinal matris :
      12.000000000000000       6.000000000000000      -6.000000000000000
       6.000000000000000      16.000000000000000       2.000000000000000
      -6.000000000000000       2.000000000000000      16.000000000000000
Householder dönüşümü uygulanmış matris
      12.0000000000      -8.4852813742       0.0000000000
      -8.4852813742      14.0000000000      -0.0000000000
       0.0000000000      -0.0000000000      18.0000000000
sınır değer vektörü
      21.544003745317536
       4.455996254682470
      18.000000000000000

> Terminated with exit code 0.
```

Son bir örnek problem, Bu örnekte 4x4 matriste iterasyon sayısı arttığı için iterasyon detayları verilmedi

$[A]=\begin{bmatrix} 4 & -2 & 6 & 4 \\ -2 & 2 & -1 & 3 \\ 6 & -1 & 22 & 13 \\ 4 & 3 & 13 & 46 \end{bmatrix}$ matrisinin önce Householder dönüşümünü yapalım sonra da sınır değerlerini bulalım.

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO13E
Householder dönüşümü :
Orijinal matris :
       4.000000000000000      -2.000000000000000       6.000000000000000       4.000000000000000
      -2.000000000000000       2.000000000000000      -1.000000000000000       3.000000000000000
       6.000000000000000      -1.000000000000000      22.000000000000000      13.000000000000000
       4.000000000000000       3.000000000000000      13.000000000000000      46.000000000000000
Householder dönüşümü uygulanmış matris
       4.0000000000       7.4833147735       0.0000000000       0.0000000000
       7.4833147735      38.1428571429     -18.2555554865       0.0000000000
       0.0000000000     -18.2555554865      25.9763100341      -8.1005851348
```

```
      0.0000000000      0.0000000000     -8.1005851348      5.8808328230
sınır değer vektörü
     52.542418859813130
     17.810856825590957
      3.469311789717206
      0.177412524878721

> Terminated with exit code 0.
```

9.4 Karekteristik polinom ve Leverrier metodu

Bir Matrisin karekteristik polinomu deyince Matrisin sınır değer matrisinden (birim matrisin sınır değerle çarpılmış hali) matrisis çıkardıktan sonra determinantını aldığımızda elde ettiğimiz polinomdur. Bu polinomun kökleri sınır değerlere eşittir.

$$\Delta(\lambda)=\det\left\{I\lambda-\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{11} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{11} & a_{11} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots. & \dots. \\ a_{1n} & a_{2n} & a_{3n} & \dots & a_{nn} \end{bmatrix}\right\} \qquad \Delta(\lambda)=\det\left\{\begin{bmatrix} \lambda-a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{11} & \lambda-a_{22} & a_{23} & \dots & a_{2n} \\ a_{11} & a_{11} & \lambda-a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots. & \dots. \\ a_{1n} & a_{2n} & a_{3n} & \dots & \lambda-a_{nn} \end{bmatrix}\right\}$$

Bir örnekle bunu açıklayalım $A=\begin{bmatrix} 1 & 1 \\ 6 & 2 \end{bmatrix}$ Matrisi verilmişse, bu matrisin karekteristik polinomu

$$\Delta(\lambda)=\det(\lambda I-A)=\det\left\{\begin{bmatrix} \lambda-1 & 1 \\ 6 & \lambda-2 \end{bmatrix}\right\}=(\lambda-1)((\lambda-2)-6=\lambda^2-3\lambda-4$$

Bu denklemin köklerini ikinci derece denklem kök formülünden rahatça hesaplayabiliriz.

$$\lambda_{1,2}=\frac{3\pm\sqrt{9+16}}{2}=\{4,-1\}$$

Bu durumda karekteristik polinomu oluşturup köklerini bulursak sınır değer vektörünü bulmuş oluruz. Burada karekteristik denklemi oluşturmak için Leverrier metodunu göstereceğiz. Bu metod:

B_n=A ve a_n=-trace(B_n) (trave işlemi köşegen değerlerin toplamını alma işlemidir) $trace(B_n)=\sum_{i=0}^{n-1} b_{ii}$

For k=n-2 den 0 a kadar
{ $C_k=(B_{k+1}+a_{k+1}I)$
  $B_k=AC_k$
  $a_k=-\frac{trace(B_n)}{n-k}$
}

Görüldüğü gibi uygulama açısından bakıldığında metod oldukça basittir. Örnek problemimizde en polinom katsayıları Leverrier metoduyla bulunduktan sonra en küçük ve en büyük sınır değerler kuvvet metoduyla saptanmakta diğer sınır değerler de sınır değer bölgeleri tayin edilerek bulunduktan sonra brent algoritmasıyla bulunmaktadır. Metodumuzun diğer bir ilginç özelliğide bize ters matrisin değerini hesaplamasıdır. Ters Matris değeri $A^{-1}=\frac{C_0}{a_0}$ değerinde bulunur.

Burada Leverier metodu kullanarak simetrik olmayan matrislerin sınır değerlerini de bulabileceğimizi anımsatalım, bunun için sadece bulduğumuz polinomun kompleks köklerini bulma işlemi yapmamız gerekir.

İlk örnek Matrisimiz :

$$[A]=\begin{bmatrix} 12 & 6 & -6 \\ 6 & 16 & 2 \\ -6 & 2 & 16 \end{bmatrix}$$

Program 9.4.1 Leverrier metodu ile karekteristik polinom ve simetrik matrislerin gerçek sınır değerlerinin hesaplanması

```
 public class Leverrier
{
double D[][]; //inverse of matri
public static double trace(double B[][])
{
double toplam=0;
int n=B.length;
for(int i=0;i<n;i++)
{toplam+=B[i][i];
}
return toplam;
}

public static double[][] topla(double A[][],double a)
{
int n=A.length;
int m=A[0].length;
double B[][]=new double[n][m];
for(int i=0;i<n;i++)
{for(int j=0;j<m;j++)
  if(i!=j) B[i][j]=A[i][j];
  else     B[i][j]=A[i][j]+a;
}
return B;
}
public static double[][] carp(double left,double[][] right)
{
//multiplying a matrix with a constant
int i,j;
int n=right.length;
int m=right[0].length;
double b[][];
b=new double[n][m];
for(i=0;i<n;i++)
  {
  for(j=0;j<m;j++)
    b[i][j]=right[i][j]*left;
  }
return b;
//end of multiplying a matrix with a constant double
}
public static double[][] carp(double[][] left,double[][] right)
{
//iki matrisin çarpımı
```

```
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right[0].length;
int n2=right.length;
double[][] b;
b=new double[n1][m2];
 if(m1 != n2)
 {
 System.out.println("uyarı: matris boyutları aynı olmalıdır");
 for(ii=0;ii<n1;ii++)
   {
   for(jj=0;jj<m2;jj++)
     b[ii][jj]=0;
   }
   return b;
 }
 for(i=0;i<n1;i++)
   {
   for(j=0;j<m2;j++)
     {
     for(k=0;k<m1;k++){
       //System.out.println(i+"  "+j+"   "+k);
                b[i][j]+=left[i][k]*right[k][j];
                //System.out.println(b[i][j]);
    }
     }
   }
return b;
}
public  double[] polynom_katsayi(double A[][])
{
int n=A.length;
int m=A[0].length;
double B[][]=new double[n][m];
for(int i=0;i<n;i++)
{for(int j=0;j<m;j++)
  B[i][j]=A[i][j];
}
System.out.println("i="+(n-1)+"\n"+Matrix.toString(B));
double a[]=new double[n+1];
double C[][]=new double[n][m];
a[n]=1.0;
a[n-1]=-trace(B);
for(int k=n-2;k>=0;k--)
{
C=topla(B,a[k+1]);
B=carp(A,C);
System.out.println("i="+k+"\n"+Matrix.toString(C));
a[k]=-trace(B)/(double)(n-k);
}
D=carp((1.0/a[0]),C);
```

```
return a;
}
//polynom fonksiyonu
public static double func(double e[],double x)
{
int n=e.length;
double ff;
if(n!=0.0)
  { ff=e[n-1];
  for(int i=n-2;i>=0;i--)
   { ff=ff*x+e[i]; }
  }
else
  ff=0;
return ff;
}
public static double[][] koksinirisapta(double e[],double x1,double x2,int n,int nbb)
{
        // koklerin yer aldığı alt bölgeleri saptar
        // n : verilen bölgeyi böldüğümüz alt bölge sayısı
        // x1,x2 : sınır değerleri
        // nbb = aranan bölgedeki köksayısı
        //
        int nb;
        int i;
        double x,fp,fc,dx;
        double xb[][]=new double[2][nbb];
        nb=0;
        dx=(x2-x1)/n;
        x=x1;
        fp=func(e,x1);
        for (i=1;i<=n;i++)
        {
                x+=dx;
                fc=func(e,x);
                // eğer kök olan bölge bulunduysa.....
                if (fc*fp < 0.0 || fp==0) {
                        xb[0][nb]=x-dx;
                        xb[1][nb]=x;
                        nb++;
                }
                fp=fc;
                if (nbb == nb) return xb;
        }
if( nb == 0)
        System.out.println("arama tamamlandı kök olan bölge bulunamadı ");
else if(nb<nbb)
  {       System.out.println("arama tamamlandı sadece "+nb+" adet kök bulundu \n"+
    "siz "+nbb+" adet kök için arama yaptırdınız");
   double xc[][]=new double[2][nb];
   for (i=0;i<nb;i++) {xc[0][i]=xb[0][i];xc[1][i]=xb[1][i];}
   return xc;
```

```
  }
return xb;
}
public static double brent(double e1[], double x1,double x2)
{
   //lineer olmayan denklemin kökü : brent metodu
        //referans : Numerical Recipes in C, second edition, William H. Press,
   //Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery
   //cambridge university press
        double ITMAX=200;
   double EPS=3.0e-10;
        int iter;
        double a=x1,b=x2,c,d,e,min1,min2;
        double tol=1.0e-5;
        c=0;
        d=0;
        e=0;
        double fa=func(e1,a),fb=func(e1,b),fc,p,q,r,s,tol1,xm;
        if (fb*fa > 0.0) System.out.println("Kök sınırları doğru olarak seçilmemiş \n sonuç hatalı olabilir");
        fc=fb;
        for (iter=1;iter<=ITMAX;iter++) {
                if (fb*fc > 0.0) {
                        c=a;
                        fc=fa;
                        e=d=b-a;
                }
                if (Math.abs(fc) < Math.abs(fb)) {
                        a=b;
                        b=c;
                        c=a;
                        fa=fb;
                        fb=fc;
                        fc=fa;
                }
                tol1=2.0*EPS*Math.abs(b)+0.5*tol;
                xm=0.5*(c-b);
                if (Math.abs(xm) <= tol1 || fb == 0.0) return b;
                if (Math.abs(e) >= tol1 && Math.abs(fa) > Math.abs(fb)) {
                        s=fb/fa;
                        if (a == c) {
                                p=2.0*xm*s;
                                q=1.0-s;
                        } else {
                                q=fa/fc;
                                r=fb/fc;
                                p=s*(2.0*xm*q*(q-r)-(b-a)*(r-1.0));
                                q=(q-1.0)*(r-1.0)*(s-1.0);
                        }
                        if (p > 0.0)  q = -q;
                        p=Math.abs(p);
                        min1=3.0*xm*q-Math.abs(tol1*q);
                        min2=Math.abs(e*q);
```

```
                        if (2.0*p < (min1 < min2 ? min1 : min2)) {
                                e=d;
                                d=p/q;
                        } else {
                                d=xm;
                                e=d;
                        }
                } else {
                        d=xm;
                        e=d;
                }
                a=b;
                fa=fb;
                if (Math.abs(d) > tol1)
                        b += d;
                else
                        b += (xm > 0.0 ? Math.abs(tol1) : -Math.abs(tol1));
                fb=func(e1,b);
        }
        System.out.println("Uyarı maximum iterasyon sayısı aşıldı \n"+
  " çözüm geçerli olmıyabilir");
return b;
}
public  double[] Leverrier(double A[][])
{
int n=A.length;
double a[]=polynom_katsayi(A);
System.out.println("Polynom katsayıları a[0]..a[n] \n"+Matrix.toStringT(a));
double lambda[]=new double[n];
lambda[n-1]=SCO13C.eigen_power(A);
lambda[0]=SCO13D.eigen_inversepower(A);
double xx[][]=koksinirisapta(a,lambda[0]-0.1,lambda[n-1]+0.1,40,n);
System.out.println("kök sınırları : \n"+Matrix.toString(xx));
for(int k=1;k<(n-1);k++)
{lambda[k]=brent(a, xx[0][k],xx[1][k]);}
return lambda;
}

public static void main(String arg[])
{
double A[][]={{12,6,-6},{6,16,2},{-6,2,16}};
Leverrier lv=new Leverrier();
double a[]=lv.Leverrier(A);
System.out.println("A-1 = \n"+Matrix.toString(lv.D));
System.out.println("lambda = \n"+Matrix.toStringT(a));
}
}
```

```
 ---------- Capture Output ----------
> "c:\java\bin\javaw.exe" Leverrier
i=2
      12.000000000000000        6.000000000000000       -6.000000000000000
```

```
      6.000000000000000      16.000000000000000       2.000000000000000
     -6.000000000000000       2.000000000000000      16.000000000000000

i=1
    -32.000000000000000       6.000000000000000      -6.000000000000000
      6.000000000000000     -28.000000000000000       2.000000000000000
     -6.000000000000000       2.000000000000000     -28.000000000000000

i=0
    252.000000000000000    -108.000000000000000     108.000000000000000
   -108.000000000000000     156.000000000000000     -60.000000000000000
    108.000000000000000     -60.000000000000000     156.000000000000000

Polynom katsayıları a[0]..a[n]
   -1728.000000000000000
     564.000000000000000
     -44.000000000000000
       1.000000000000000

kök sınırları :
      4.355996254682469      17.754202059924630      21.211803558051650
      4.788196441948346      18.186402247190507      21.644003745317526

A-1 =
     -0.145833333333333       0.062500000000000      -0.062500000000000
      0.062500000000000      -0.090277777777778       0.034722222222222
     -0.062500000000000       0.034722222222222      -0.090277777777778

lambda =
      4.455996254682469
     18.000000089225104
     21.544003745317520


> Terminated with exit code 0.
```

İkinci örneğimiz :

$$[A]=\begin{bmatrix} 4 & -2 & 6 & 4 \\ -2 & 2 & -1 & 3 \\ 6 & -1 & 22 & 13 \\ 4 & 3 & 13 & 46 \end{bmatrix}$$

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" Leverrier
Polynom katsayıları a[0]..a[n]
     576.000000000000000
   -3456.000000000000000
    1193.000000000000000
     -74.000000000000000
       1.000000000000000

kök sınırları :
```

```
      0.077412524878722   2.705662841625443   17.161039583732414   51.328293701439755
      1.391537683252082   4.019787999998804   18.475164742105775   52.642418859813110

lambda =
      0.177412524878722
      3.469311676467504
     17.810855411801622
     52.542418859813150
> Terminated with exit code 0.
```

9.5 QL algoritması

Simetrik bir A matrisi alalım. Bu matrisi Q ortagonal ve L alt matris olmak üzere
A=Q L biçiminde yazılabilir. Q ortogonal bir matris olduğundan $Q^T = Q^{-1}$ dir. bu yüzden denklemin her iki tarafını da Q^T ile çarparak
$Q^T A = L$ yazılabilir. buradan yeni bir B matrisi oluşturulabilir.
$B=L\ Q = Q^T A\ Q$ olur.B matrisi A matrisinin benzerlik dönüşümüne uğramış durumu olup özdeğerleri aynıdır.
QL algoritması bir dizi ortogonal dönüşümü esas almaktadır. Bu bir tekrarlama işlemidir ve tekrarlama bağıntısı
$\mathbf{A_m=Q_m\ L_m}$ olmak üzere $\mathbf{A_{m+1}= L_m\ Q_m = Q_m^T\ A_m\ Q_m}$ olarak verilir. Burada ilgilendiğimiz A matrisi simetrik olduğundan özdeğerleri gerçek sayılar olacaktır.

Program 9.5.1 QL algoritması

```
 import java.io.*;
import javax.swing.*;
//import Numeric;

class SCO13A
{
public static double[][] tridiagonal(double b[][], double d[], double e[])
{
//reduces matrix to tridiaonal form by using HOUSEHOLDER TRANSFORMATION
//this method is used by QL method to calculate eigen values
//and eigen  vectors of a symmetric matrix
            int l,k,j,i;
   int n=b.length;
            double scale,hh,h,g,f;
   double a[][]=new double[n+1][n+1];
   double c[][]=new double[n][n];
   for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      a[i][j]=b[i][j];
            for (i=n;i>=2;i--) {
                        l=i-1;
                        h=scale=0.0;
                        if (l > 1) {
                                    for (k=1;k<=l;k++)
                                                scale += Math.abs(a[i-1][k-1]);
                                    if (scale == 0.0)
                                                e[i-1]=a[i-1][l-1];
                                    else {
                                                for (k=1;k<=l;k++) {
                                                            a[i-1][k-1] /= scale;
                                                            h += a[i-1][k-1]*a[i-1][k-1];
                                                }
                                                f=a[i-1][l-1];
                                                g=(f >= 0.0 ? -Math.sqrt(h) : Math.sqrt(h));
                                                e[i-1]=scale*g;
                                                h -= f*g;
```

```
                                        a[i-1][l-1]=f-g;
                                        f=0.0;
                                        for (j=1;j<=l;j++) {
                                                a[j-1][i-1]=a[i-1][j-1]/h;
                                                g=0.0;
                                                for (k=1;k<=j;k++)
                                                        g += a[j-1][k-1]*a[i-1][k-1];
                                                for (k=j+1;k<=l;k++)
                                                        g += a[k-1][j-1]*a[i-1][k-1];
                                                e[j-1]=g/h;
                                                f += e[j-1]*a[i-1][j-1];
                                        }
                                        hh=f/(h+h);
                                        for (j=1;j<=l;j++) {
                                                f=a[i-1][j-1];
                                                e[j-1]=g=e[j-1]-hh*f;
                                                for (k=1;k<=j;k++)
                                                        a[j-1][k-1] -= (f*e[k-1]+g*a[i-1][k-1]);
                                        }
                                }
                        } else
                                e[i-1]=a[i-1][l-1];
                        d[i-1]=h;
                }
                d[1-1]=0.0;
                e[1-1]=0.0;
                /* Contents of this loop can be omitted if eigenvectors not
                                wanted except for statement d[i-1]=a[i-1][i-1]; */
                for (i=1;i<=n;i++) {
                        l=i-1;
                        if (d[i-1] != 0) {
                                for (j=1;j<=l;j++) {
                                        g=0.0;
                                        for (k=1;k<=l;k++)
                                                g += a[i-1][k-1]*a[k-1][j-1];
                                        for (k=1;k<=l;k++)
                                                a[k-1][j-1] -= g*a[k-1][i-1];
                                }
                        }
                        d[i-1]=a[i-1][i-1];
                        a[i-1][i-1]=1.0;
                        for (j=1;j<=l;j++) a[j-1][i-1]=a[i-1][j-1]=0.0;
                }
return  a;
}

public static double pythag(double a, double b)
{
//this method is used by QL method
                double absa,absb;
                absa=Math.abs(a);
                absb=Math.abs(b);
                if (absa > absb) return absa*Math.sqrt(1.0+(absb/absa)*(absb/absa));
                else return (absb==0.0 ? 0.0 : absb*Math.sqrt(1.0+(absa/absb)*(absa/absb)));
}

public static double[][] QL(double d[], double e[], double a[][])
{
// QL algorithm : eigenvalues of a symmetric matrix reduced to tridiagonal
// form by using method tridiagonal
    int n=d.length;
                int m,l,iter,i,j,k;
                double s,r,p,g,f,dd,c,b;
                for (i=2;i<=n;i++) e[i-2]=e[i-1];
                e[n-1]=0.0;
    double z[][]=new double[n][n];
```

```
    for(i=0;i<n;i++)
      for(j=0;j<n;j++)
        z[i][j]=a[i][j];
              for (l=1;l<=n;l++) {
                        iter=0;
                        do {
                                  for (m=l;m<=n-1;m++) {
                                            dd=Math.abs(d[m-1])+Math.abs(d[m]);
                                            if ((Math.abs(e[m-1])+dd) == dd) break;
                                  }
                                  if (m != l) {
          if (iter++ == 30) System.out.println("Too many iterations in QL");
          g=(d[l]-d[l-1])/(2.0*e[l-1]);
          r=Matrix.pythag(g,1.0);
          g=d[m-1]-d[l-1]+e[l-1]/(g+Matrix.SIGN(r,g));
          s=c=1.0;
          p=0.0;
          for (i=m-1;i>=l;i--) {
                                                      f=s*e[i-1];
                                                      b=c*e[i-1];
                                                      e[i]=(r=Matrix.pythag(f,g));
                                                      if (r == 0.0) {
                                                                d[i] -= p;
                                                                e[m-1]=0.0;
                                                                break;
                                                      }
                                                      s=f/r;
                                                      c=g/r;
                                                      g=d[i ]-p;
                                                      r=(d[i-1]-g)*s+2.0*c*b;
                                                      d[i ]=g+(p=s*r);
                                                      g=c*r-b;
                                                      for (k=1;k<=n;k++) {
                                                                f=z[k-1][i ];
                                                                z[k-1][i ]=s*z[k-1][i-1]+c*f;
                                                                z[k-1][i-1]=c*z[k-1][i-1]-s*f;
                                                      }
                                            }
                                            if (r == 0.0 && i >= l) continue;
                                            d[l-1] -= p;
                                            e[l-1]=g;
                                            e[m-1]=0.0;
                                  }
                        } while (m != l);
              }
    return z;
}

public static double[][] eigenQL(double a[][])
{
 // QL algoritm to solve eigen value problems
 // symmetric matrices only (real eigen values)
 // first column of the matrix returns eigen values
 // second..n+1 column returns eigen vectors.
 // Note : If matrix is not symmetric DO NOT use
 // this method use eigenValue method (a QR algorithm)
 int i,j;
 int n=a.length;
 double sum[]=new double[n];;
 double d[]=new double[n];
 double b[][]=new double[n][n];
 double e[]=new double[n];
 double z[][]=new double[n+1][n];
 b=tridiagonal(a,d,e);
 b=QL(d,e,b);
 for(j=0;j<n;j++)
```

```
    {
    z[0][j]=d[j];
    for(i=0;i<n;i++)
      {
      z[i+1][j]=b[i][j]/b[0][j];
      if(z[i+1][j]<1e-13) z[i+1][i]=0;
      }
    }
 // result: first row is eigenvalues, the rest is eigenvector
 return z;
}

  public static void main(String args[]) throws IOException
  {
  //Eigen Value calculations
  double c[][];
  c=new double[3][3];
  double a[][]={{3.556,-1.778,0},{-1.778,3.556,-1.778},{0,-1.778,3.556}};
  //double a[][]={{-19,-4.7,6.45},{-4.7,4.6,11.8},  {6.45,11.8,-8.3}};
  //double a[][]={{3,-1,0},{-1,2,-1},{0,-1,3}};
  String s="Orijinal matris : \n";
  s+=Matrix.toString(a);
  c=new double[3][3];
  c=eigenQL(a);
  s+="sınır değer matrisi (birinci kolon sınır değerler) \n";
  s+=Matrix.toString(c);
  System.out.println("QL eigenvalue metodu : \n"+s);
  }
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO13A
QL eigenvalue metodu :
Orijinal matris :
       3.556000000000000      -1.778000000000000       0.000000000000000
      -1.778000000000000       3.556000000000000      -1.778000000000000
       0.000000000000000      -1.778000000000000       3.556000000000000
sınır değer matrisi (birinci kolon sınır değerler)
       1.041528286100637       3.556000000000000       6.070471713899361
       1.000000000000000       1.000000000000000       1.000000000000000
       1.414213562373095       0.000000000000000      -1.414213562373095
       1.000000000000000      -1.000000000000000       1.000000000000000


> Terminated with exit code 0.
```

9.6 QR algoritması

Benzerlik transformasyonu ile A matrisi ortogonal Q matrisi ve üst üçgen R matrisine ayrılabilir

$A=Q\,R$

Q ortogonal bir matris olduğundan $Q^T = Q^{-1}$ bu yüzden denklemin her iki tarafını da Q^T ile çarparak

$Q^T A = R$ buradan yeni bir B matrisi oluşturulabilir.

$B=R\,Q = Q^T A\,Q$

Bu yüzden B ve A benzerlik transformasyonu ile birbiriyle ilişkilidir, yani aynı sınır değere sahiptir. B yine aynı şekilde tekrar benzerlik transformasyonundan geçirilir, B üst üçgen forma ulaştığı zaman, üst üçgen formdan sınır değerler elde edilebilir. Üst üçgene çevirme prosesi oldukça uzun sürebilir. Bu prosesi kısaltabilmek için matrisi üst hesenberg formuna çevirmek gerekir. Bunun için Householder transferiyle gerçekleştirilebilir. QR algoritmasında matris simetrik

olmak zorunda değildir. Elbette simetrik olmayan matrisler için özdeğerlerin kompleks (karmaşık) sayılar çıkacağını göz önünde bulundurmamız gereklidir.

Program 9.6.1 QR algoritması

```
 import java.io.*;
import javax.swing.*;


class SCO13B
{
  //===========Eigen value calculations ==============

public static double[][] balance(double b[][])
{
// balance of a matrix for more accurate eigenvalue
// calculations
double radix=2.0;
double sqrdx=radix*radix;
double c,r,f,s,g;
int m,j,i,last;
int n=b.length;
last=0;
double a[][];
a=new double[n][n];
f=1;
s=1;
for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
    a[i-1][j-1]=b[i-1][j-1];
while(last==0)
  {
  last=1;
  for(i=1;i<=n;i++)
    {
    c=0;r=0;
    for(j=1;j<=n;j++)
      {
      if(j != i)
                {
                c+=Math.abs(a[j-1][i-1]);
                r+=Math.abs(a[i-1][j-1]);
                } //end of if(j!=..
      } //end of for(j=1...
    if(c != 0 &&  r != 0 )
      {
      g=r/radix;
      f=1.0;
      s=c+r;
      while(c<g)
                {
                f*=radix;
                c*=sqrdx;
                }
      g=r*radix;
      while(c>g)
                {
                f/=radix;
                c/=sqrdx;
                }
      } //end of if(c != 0 && ....
    if( (c+r)/f < 0.95*s )
      {
      last=0;
      g=1.0/f;
      for(j=1;j<=n;j++)  { a[i-1][j-1]*=g; }
```

```
      for(j=1;j<=n;j++)  { a[j-1][i-1]*=f; }
     } //end of if( ((c+r..
    }//end of for(i=1;i<=n....
  } //end of while last==0
return a;
}

public static double[][] Hessenberg(double b[][])
{
// Calculates the hessenberg matrix
// it is used in QR method to calculate eigenvalues
// of a matrix(symmetric or non-symmetric)
int m,j,i;
int n=b.length;
double a[][];
a=new double[n][n];
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    a[i][j]=b[i][j];
double x,y;
if(n>2)
  {
  for(m=2;m<=(n-1);m++)
    {
    x=0.0;
    i=m;
    for(j=m;j<=n;j++)
      {
      if(Math.abs(a[j-1][m-2]) > Math.abs(x))
                {
                x=a[j-1][m-2];
                i=j;
                } //end of if(Math.abs(..
      }//end of for(j=m,j<=n...
    if(i!=m)
      {
      for(j=(m-1);j<=n;j++)
                {
                y=a[i-1][j-1];
                a[i-1][j-1]=a[m-1][j-1];
                a[m-1][j-1]=y;
                }//end of for(j=(m-1)..
      for(j=1;j<=n;j++)
                {
                y=a[j-1][i-1];
                a[j-1][i-1]=a[j-1][m-1];
                a[j-1][m-1]=y;
                }//end of for(j=1;j<=n....
      } //end of if(i!=m)
    if(x != 0.0)
      {
      for(i=(m+1);i<=n;i++)
                {
                y=a[i-1][m-2];
                if(y!=0.0)
                  {
                  y=y/x;
                  a[i-1][m-2]=y;
                  for(j=m;j<=n;j++)
                    {
                    a[i-1][j-1]-=y*a[m-1][j-1];
                    }
                  for(j=1;j<=n;j++)
                    {
                    a[j-1][m-1]+=y*a[j-1][i-1];
                    }
```

```
                }//end of if(y!=0..
              }//end of for(i=(m+1)...
      } //end of if(x != 0.0...
    }//end of for(m=2;m<=(n-1)..
  }//end of Hessenberg
for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
  {
  if(i>(j+1)) a[i-1][j-1]=0;
  }
return a;
}

public static double[][] QR(double b[][])
{
//calculates eigenvalues of a Hessenberg matrix
int n=b.length;
double rm[][]=new double[2][n];
double a[][]=new double[n+1][n+1];
double wr[]=new double[n+1];
double wi[]=new double[n+1];
int nn,m,l,k,j,its,i,mmin;
double z,y,x,w,v,u,t,s,r=0,q=0,p=0,anorm;
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
      a[i+1][j+1]=b[i][j];
anorm=Math.abs(a[1][1]);
for (i=2;i<=n;i++)
   for (j=(i-1);j<=n;j++)
       anorm += Math.abs(a[i][j]);
nn=n;
t=0.0;
while (nn >= 1) {
   its=0;
   do {
      for (l=nn;l>=2;l--) {
      s=Math.abs(a[l-1][l-1])+Math.abs(a[l][l]);
      if (s == 0.0) s=anorm;
      if ((Math.abs(a[l][l-1]) + s) == s) break;
                    }
      x=a[nn][nn];
      if (l == nn) {
         wr[nn]=x+t;
                wi[nn--]=0.0;
                            }
      else {
          y=a[nn-1][nn-1];
                w=a[nn][nn-1]*a[nn-1][nn];
                if (l == (nn-1)) {
                   p=0.5*(y-x);
                   q=p*p+w;
                   z=Math.sqrt(Math.abs(q));
                   x += t;
                   if (q >= 0.0) {
                      z=p+Matrix.SIGN(z,p);
                            wr[nn-1]=wr[nn]=x+z;
                            if (z!=0) wr[nn]=x-w/z;
                            wi[nn-1]=wi[nn]=0.0;
                                        }
                   else {
                      wr[nn-1]=wr[nn]=x+p;
                            wi[nn-1]= -(wi[nn]=z);
                             }
                   nn -= 2;
                                         }
                else {
```

```
            if (its == 30) System.out.println("Matrix.java, module QR, Too many iterations in hqr");
            if (its == 10 || its == 20) {
              t += x;
                    for (i=1;i<=nn;i++) a[i][i] -= x;
                    s=Math.abs(a[nn][nn-1])+Math.abs(a[nn-1][nn-2]);
                    y=x=0.75*s;
                    w = -0.4375*s*s;
                                                        }
            ++its;
            for (m=(nn-2);m>=l;m--) {
            z=a[m][m];
            r=x-z;
            s=y-z;
            p=(r*s-w)/a[m+1][m]+a[m][m+1];
            q=a[m+1][m+1]-z-r-s;
            r=a[m+2][m+1];
            s=Math.abs(p)+Math.abs(q)+Math.abs(r);
            p /= s;
            q /= s;
            r /= s;
            if (m == l) break;
            u=Math.abs(a[m][m-1])*(Math.abs(q)+Math.abs(r));
            v=Math.abs(p)*(Math.abs(a[m-1][m-1])+
        Math.abs(z)+Math.abs(a[m+1][m+1]));
            if ((u+v) == v) break;
                                                        }
            for (i=m+2;i<=nn;i++) {
              a[i][i-2]=0.0;
              if (i != (m+2)) a[i][i-3]=0.0;
                                }
            for (k=m;k<=nn-1;k++) {
            if (k != m) {
              p=a[k][k-1];
                    q=a[k+1][k-1];
                    r=0.0;
                    if (k != (nn-1)) r=a[k+2][k-1];
                    if ((x=Math.abs(p)+Math.abs(q)+Math.abs(r)) != 0.0) {
                      p /= x;
                      q /= x;
                      r /= x;
                                                        }                                       }
                    if ((s=Matrix.SIGN(Math.sqrt(p*p+q*q+r*r),p)) != 0.0) {
                      if (k == m) {
                        if (l != m)
                        a[k][k-1] = -a[k][k-1];
                                          }
                      else
                        a[k][k-1] = -s*x;
                      p += s;
                      x=p/s;
                      y=q/s;
                      z=r/s;
                      q /= p;
                      r /= p;
                      for (j=k;j<=nn;j++) {
                      p=a[k][j]+q*a[k+1][j];
                      if (k != (nn-1)) {
                        p += r*a[k+2][j];
                        a[k+2][j] -= p*z;
                                }
                      a[k+1][j] -= p*y;
                      a[k][j] -= p*x;
                                                }
                      mmin = nn<k+3 ? nn : k+3;
                      for (i=l;i<=mmin;i++) {
                        p=x*a[i][k]+y*a[i][k+1];
```

```
                                if (k != (nn-1)) {
                                  p += z*a[i][k+2];
                                        a[i][k+2] -= p*r;
                                                }
                                a[i][k+1] -= p*q;
                                a[i][k] -= p;
                            }
                    }
                  }
        }
      }
   } while (l < nn-1);
}
for(i=0;i<n;i++)
{
rm[0][i]=wr[i+1];
rm[1][i]=wi[i+1];
}
return rm;
} //end of QR

public static double[][] eigenValue(double b[][])
{
// this routine input a matrix (non symetric or symmetric)
// and calculate eigen values
// method balance can be used prior to this method to balance
// the input matrix
int n=b.length;
double d[][]=new double[2][n];
d=QR(Matrix.Hessenberg(b));
return d;
}

public static complex[] eigenValueC(double b[][])
{
// this routine input a matrix (non symetric or symmetric)
// and calculate eigen values
// method balance can be used prior to this method to balance
// the input matrix
//output eigenvalues will be in a vector of complex form
int n=b.length;
double d[][]=new double[2][n];
d=QR(Matrix.Hessenberg(b));
complex c[]=new complex[n];
for(int i=0;i<n;i++)
  {
  c[i]=new complex(d[0][i],d[1][i]);
  }
return c;
}
  public static void main(String args[]) throws IOException
  {
  //Eigen Value calculations using  method
  //double a[][]={{8,-1,-5},{-4,4,-2},{18,-5,-7}};
    //double a[][]={{3,-1,0},{-1,2,-1},{0,-1,3}};
    double a[][]={{3.556,-1.778,0},{-1.778,3.556,-1.778},{0,-1.778,3.556}};
  String s="Orijinal Matris : \n";
  s+=Matrix.toString(a);
  //Eigenvlue calculation
  complex c1[];
  int n=a.length;
  c1=new complex[n];
  c1=eigenValueC(a);
  s+="QR metodu Sınır değer vektörü \n";
  s+=Matrix.toStringT(c1);
  System.out.println("QR  eigenvalue metodu : \n"+s);
```

```
  }
}
```

Bu programda kullanılan kompleks değişken hesabını yapan complex sınıfı da program 4.4-3 de verilmiştir.

Çıktı 8.2-1 QR algoritması

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SCO13B
QR  eigenvalue metodu :
Orijinal Matris :
      3.556000000000000      -1.778000000000000       0.000000000000000
     -1.778000000000000       3.556000000000000      -1.778000000000000
      0.000000000000000      -1.778000000000000       3.556000000000000
QR metodu Sınır değer vektörü
(   1.0415282861 +    0.0000000000i )
(   3.5560000000 +    0.0000000000i )
(   6.0704717139 +    0.0000000000i )


> Terminated with exit code 0.
```

9.7 Polinomların kökleri ve refakatçi matrisi

$f(x)= a_n x^n + \ldots + a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0$

ninci derece polinomunu göz önüne alalım. Bu polinomun katsayılarını a_n değerine bölecek olursak
$f(x)= x^n + b_{n-1} x^4 + \ldots + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0$

durumunu alır. Bu polinomun kökleri

$$A=\begin{bmatrix} -b_{n-1} & -b_{n-2} & \ldots & -b_1 & -b_0 \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ & & \ldots & 1 & 0 \end{bmatrix}$$ matrisinin sınır değerleri ile aynıdır. Bu yüzden n'inci dereceden

polinomun köklerini bulmak için refakatçi matrisinin özdeğerlerini hesaplayabiliriz.

Program 9.7.1 QR metodu ile refakatçi matris kullanarak ninci dereceden denklemin köklerinin çözülmesi

```
 import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;


class SCO13I extends JFrame implements ActionListener
{ private static final long serialVersionUID = 86798467L;
```

```
public static double[][] balancedEigenValue(double b[][])
{
// this routine input a matrix (non symetric or symmetric)
// and calculates eigen values
// method balance is used to balance the matrix previous to
// actual calculations
int n=b.length;
double d[][]=new double[2][n];
d=SCO13B.QR(SCO13B.Hessenberg(Matrix.balance(b)));
return d;
}
public static complex[] poly_rootsC(double c[])
{
// roots of a degree n polynomial
// P(x)=c[n]*x^n+c[n-1]*x^(n-1)+....+c[1]*x+c[0]=0;
// roots are returned as complex variables
int n=c.length-1;
double a[][]=new double[n][n];
a=characteristic_matrix(c);
double d[][]=new double[2][n];
d=balancedEigenValue(a);
complex e[]=new complex[n];
for(int i=0;i<n;i++)
  e[i]=new complex(d[0][i],d[1][i]);
return e;
}

public static double SIGN(double a,double b)
{
//returns the value of double a with sign of double b;
//if a=-2, b= 3 SIGN(a,b) returns  2
//if a=-2, b=-3 SIGN(a,b) returns -2
//if a= 2, b=-3 SIGN(a,b) returns -2
//if a= 2, b= 3 SIGN(a,b) returns  2
if(b!=0)
  return Math.abs(a)*b/Math.abs(b);
else
  return Math.abs(a);
}
public static double[][] characteristic_matrix(double c[])
{
//this routine converts polynomial coefficients to a matrix
//with the same eigenvalues (roots)
int n=c.length-1;
int i;
double a[][]=new double[n][n];
for(i=0;i<n;i++)
{
  a[0][i]=-c[i+1]/c[0];
}
for(i=0;i<n-1;i++)
{
  a[i+1][i]=1;
}
return a;
}

public static double[][] tridiagonal(double b[][], double d[], double e[])
{
//reduces matrix to tridiaonal form by using HOUSEHOLDER TRANSFORMATION
//this method is used by QL method to calculate eigen values
//and eigen  vectors of a symmetric matrix
```

```
        int l,k,j,i;
int n=b.length;
        double scale,hh,h,g,f;
double a[][]=new double[n+1][n+1];
double c[][]=new double[n][n];
for(i=0;i<n;i++)
 for(j=0;j<n;j++)
   a[i][j]=b[i][j];
        for (i=n;i>=2;i--) {
                l=i-1;
                h=scale=0.0;
                if (l > 1) {
                        for (k=1;k<=l;k++)
                                scale += Math.abs(a[i-1][k-1]);
                        if (scale == 0.0)
                                e[i-1]=a[i-1][l-1];
                        else {
                                for (k=1;k<=l;k++) {
                                        a[i-1][k-1] /= scale;
                                        h += a[i-1][k-1]*a[i-1][k-1];
                                }
                                f=a[i-1][l-1];
                                g=(f >= 0.0 ? -Math.sqrt(h) : Math.sqrt(h));
                                e[i-1]=scale*g;
                                h -= f*g;
                                a[i-1][l-1]=f-g;
                                f=0.0;
                                for (j=1;j<=l;j++) {
                                        a[j-1][i-1]=a[i-1][j-1]/h;
                                        g=0.0;
                                        for (k=1;k<=j;k++)
                                                g += a[j-1][k-1]*a[i-1][k-1];
                                        for (k=j+1;k<=l;k++)
                                                g += a[k-1][j-1]*a[i-1][k-1];
                                        e[j-1]=g/h;
                                        f += e[j-1]*a[i-1][j-1];
                                }
                                hh=f/(h+h);
                                for (j=1;j<=l;j++) {
                                        f=a[i-1][j-1];
                                        e[j-1]=g=e[j-1]-hh*f;
                                        for (k=1;k<=j;k++)
                                                a[j-1][k-1] -= (f*e[k-1]+g*a[i-1][k-1]);
                                }
                        }
                } else
                        e[i-1]=a[i-1][l-1];
                d[i-1]=h;
        }
        d[1-1]=0.0;
        e[1-1]=0.0;
        /* Contents of this loop can be omitted if eigenvectors not
                        wanted except for statement d[i-1]=a[i-1][i-1]; */
        for (i=1;i<=n;i++) {
                l=i-1;
                if (d[i-1] != 0) {
                        for (j=1;j<=l;j++) {
                                g=0.0;
                                for (k=1;k<=l;k++)
                                        g += a[i-1][k-1]*a[k-1][j-1];
                                for (k=1;k<=l;k++)
                                        a[k-1][j-1] -= g*a[k-1][i-1];
```

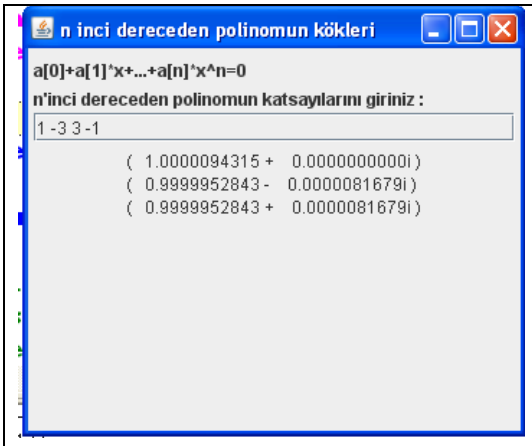
```
                                        }
                                }
                                d[i-1]=a[i-1][i-1];
                                a[i-1][i-1]=1.0;
                                for (j=1;j<=l;j++) a[j-1][i-1]=a[i-1][j-1]=0.0;
                        }
return  a;
}

public static double pythag(double a, double b)
{
//this method is used by QL method
                double absa,absb;
                absa=Math.abs(a);
                absb=Math.abs(b);
                if (absa > absb) return absa*Math.sqrt(1.0+(absb/absa)*(absb/absa));
                else return (absb==0.0 ? 0.0 : absb*Math.sqrt(1.0+(absa/absb)*(absa/absb)));
}

public static double[][] QL(double d[], double e[], double a[][])
{
// QL algorithm : eigenvalues of a symmetric matrix reduced to tridiagonal
// form by using method tridiagonal
   int n=d.length;
                int m,l,iter,i,j,k;
                double s,r,p,g,f,dd,c,b;
                for (i=2;i<=n;i++) e[i-2]=e[i-1];
                e[n-1]=0.0;
   double z[][]=new double[n][n];
   for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      z[i][j]=a[i][j];
                for (l=1;l<=n;l++) {
                                iter=0;
                                do {
                                                for (m=l;m<=n-1;m++) {
                                                                dd=Math.abs(d[m-1])+Math.abs(d[m]);
                                                                if ((Math.abs(e[m-1])+dd) == dd) break;
                                                }
                                                if (m != l) {
        if (iter++ == 30) System.out.println("Too many iterations in QL");
        g=(d[l]-d[l-1])/(2.0*e[l-1]);
        r=pythag(g,1.0);
        g=d[m-1]-d[l-1]+e[l-1]/(g+SIGN(r,g));
        s=c=1.0;
        p=0.0;
        for (i=m-1;i>=l;i--) {
                                                                                f=s*e[i-1];
                                                                                b=c*e[i-1];
                                                                                e[i]=(r=pythag(f,g));
                                                                                if (r == 0.0) {
                                                                                                d[i] -= p;
                                                                                                e[m-1]=0.0;
                                                                                                break;
                                                                                }
                                                                                s=f/r;
                                                                                c=g/r;
                                                                                g=d[i ]-p;
                                                                                r=(d[i-1]-g)*s+2.0*c*b;
                                                                                d[i ]=g+(p=s*r);
                                                                                g=c*r-b;
                                                                                for (k=1;k<=n;k++) {
```

```
                                                    f=z[k-1][i ];
                                                    z[k-1][i ]=s*z[k-1][i-1]+c*f;
                                                    z[k-1][i-1]=c*z[k-1][i-1]-s*f;
                                                }
                                            }
                                            if (r == 0.0 && i >= l) continue;
                                            d[l-1] -= p;
                                            e[l-1]=g;
                                            e[m-1]=0.0;
                                        }
                                    } while (m != l);
                                }
   return z;
}

public static double[][] eigenQL(double a[][])
{
 // QL algoritm to solve eigen value problems
 // symmetric matrices only (real eigen values)
 // first column of the matrix returns eigen values
 // second..n+1 column returns eigen vectors.
 // Note : If matrix is not symmetric DO NOT use
 // this method use eigenValue method (a QR algorithm)
 int i,j;
 int n=a.length;
 double sum[]=new double[n];;
 double d[]=new double[n];
 double b[][]=new double[n][n];
 double e[]=new double[n];
 double z[][]=new double[n+1][n];
 b=tridiagonal(a,d,e);
 b=QL(d,e,b);
 for(j=0;j<n;j++)
   {
   z[0][j]=d[j];
   for(i=0;i<n;i++)
     {
     z[i+1][j]=b[i][j]/b[0][j];
     if(z[i+1][j]<1e-13) z[i+1][i]=0;
     }
   }
 // result: first row is eigenvalues, the rest is eigenvector
 return z;
}

   private JLabel prompt1,prompt2;
   private JTextField input;
   JTextArea t;
   JPanel YaziPaneli;
   int n;
   String s;
   Container c;
   public SCO13I()
   {
      super("n inci dereceden polinomun kökleri");
      c=getContentPane();
      c.setLayout(new FlowLayout());
      YaziPaneli=new JPanel();
      YaziPaneli.setFont(new Font("Serif",Font.BOLD,12));
      YaziPaneli.setLayout( new GridLayout(3,1) );
      t=new JTextArea();
      t.setBackground(c.getBackground());
```

```
    prompt1= new JLabel("a[0]+a[1]*x+...+a[n]*x^n=0");
    prompt2= new JLabel("n'inci dereceden polinomun katsayılarını giriniz : ");
    input = new JTextField(30);
    input.setBackground(c.getBackground());
    YaziPaneli.add(prompt1);
    YaziPaneli.add(prompt2);
    YaziPaneli.add(input);
    c.add(YaziPaneli);
    c.add(t);
    input.addActionListener(this);
  }

  public void  actionPerformed(ActionEvent e)
  {
       s=input.getText();
       StringTokenizer token=new StringTokenizer(s);
       t.setText("");
       n=token.countTokens()-1;
       int m=n+1;
       double a[]=new double[m];
       complex z[]=new complex[n];
       for(int i=0;i<n;i++)
       {
       z[i]=new complex();
       }
       int j=0;
          while(token.hasMoreTokens())
          {
          Double ax=new Double(token.nextToken());
          a[j++]=ax.doubleValue();
          }
          z=poly_rootsC(a);
          int i=0;
          t.setText(Matrix.toStringT(z));
          input.setText("");
  }

   public static void main(String[] args)
   {
   SCO13I pencere= new SCO13I();
   pencere.addWindowListener(new BasicWindowMonitor());
   pencere.setSize(350,300);
   pencere.setVisible(true);
   }
}
```

Bu programda kullanılan kompleks değişken hesabını yapan complex sınıfı da program 4.4-3 de verilmiştir.

Çıktı 8.6-1 ninci dereceden denklemin köklerini QR metodu sınır değeri ile hesaplayan programın çıktısı



(bolum9_000.jpg,Resimaltı: **ninci dereceden polinomun kökleri**)

9.8 Tekil değer ayrıştırması (Singular value decompositıon –SVD) – Lineer denklem sistemlerinin çözümü

A m x n bir matris olsun. Bu matristeki satır sayısının , m, sütun sayısından büyük veya eşit olduğunu kabul edelim. Daha önce lineer denklem sistemlerinin çözümlerini incelemiştik, ancak satır ve sütun sayılarının aynı olması m=n koşulu mevcuttu. Telik değer ayrıştırması yapabilmek için mxn ortagonal bir U metrisinin mevcut bulunduğunu varsayalım. Ortogonal bir matrisin transpoze matrisi ters matrisine eşit olacaktır. Ek olarak nxn diagonal matris S matrisinin olduğunu varsayalım. Ve sonuncu olarak bir V matrisi olsun. Bu matrislerin

$$A=USV^T$$

Bağıntısını sağlamasını istiyoruz. Eğer bu özelliklerde U,S ve V matrisini bulabilirsek bu işleme A matrisinin **tekil değer ayrıştırması** adını vereceğiz. Uluslararası literatürde genellikle İngilizce isminin baş harfleriyle **SVD** olarak anılır. Eğer bir matris bu şekilde ayrıştırılabiliyorsa Bu matrisin ters matrisi:

$$A^{-1}=V\ [diag\ (s_1^{-1},s_2^{-1},\ldots,s_n^{-1})]\ (U^T)$$

Şeklinde hesaplanabilir. Ortagonal özelliklerinden dolayı U ve V matrislerinin terslerinin sadece Tranpose değerleri olduğunu not edelim. S matrisinin tersi ise sadece köşegen matris olması sebebiyle değerlerinin bire bölünmesiyle elde edilebilir. Bu özellik bize bulunan ters matrisi kullanarak lineer denklem sistemini çözme olasılığını sağlar. $\mathbf{X = A^{-1} \times B}$

Bir örnek problemle detay işlemleri inceleyelim. A matrisi verilmiş

$$A=\begin{bmatrix}4 & 0\\ 3 & -5\end{bmatrix}$$

A matrisinin Transpoze matrisi

$$A^T=\begin{bmatrix}4 & 3\\ 0 & -5\end{bmatrix}$$

A matrisini Transpoze matrisi ile çarparsak

$$A^TA=\begin{bmatrix}4 & 3\\ 0 & -5\end{bmatrix}\begin{bmatrix}4 & 0\\ 3 & -5\end{bmatrix}=\begin{bmatrix}25 & -15\\ -15 & 25\end{bmatrix}$$

Bu yeni matrisin sınır değerlerini bulalım. Burada matrisin transpozu ile çarpıldığını bu yüzden her zaman simetrik bir matris olacağını düşünürsek Jacobi metodu veya QL metodu bize çözümleri sağlayacaktır. Matrisin sınır değerleri:

$\lambda = \begin{bmatrix} 40 \\ 10 \end{bmatrix}$. S matrisini oluşturmak için sınır değerlerin kare köklerini alıp köşeğen matris haline getireceğiz.

$$S = \begin{bmatrix} \sqrt{40} & 0 \\ 0 & \sqrt{10} \end{bmatrix} = \begin{bmatrix} 6.3245 & 0 \\ 0 & 3.1622 \end{bmatrix}$$

Bu değerlerin 1 e bölünmesiyle tear S matrisi kolaylıkla hesaplanabilir.

$$S^{-1} = \begin{bmatrix} 1/6.3245 & 0 \\ 0 & 1/3.1622 \end{bmatrix} = \begin{bmatrix} 0.1581 & 0 \\ 0 & 0.3162 \end{bmatrix}$$

Bu sınır değerlere karşı gelen sını değer vektörleri temel sınır değer vektör tanımından hesaplanabilir.

$$\left[A^T A - \lambda I\right] X^1] = \begin{bmatrix} 25-40 & -15 \\ -15 & 25-40 \end{bmatrix} \begin{bmatrix} x_1^1 \\ x_2^1 \end{bmatrix} = 0$$

$$\left[A^T A - \lambda I\right] X^2] = \begin{bmatrix} 25-10 & -15 \\ -15 & 25-10 \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} = 0$$

$$[X] = \begin{bmatrix} 0.707106781186548 & 0.707106781186548 \\ -0.707106781186548 & 0.707106781186548 \end{bmatrix}$$

Burada bulduğumuz sınır değer vektörünü boyut vektörü L ile normalize edeceğiz.

$L = \sqrt{(x_1)^2 + (x_2)^2}$ ve böylece normalize edilmiş sınırdeğer vektörü bie V vektörünü verir.

$$V = \begin{bmatrix} 0.707106781186548 & 0.707106781186548 \\ -0.707106781186548 & 0.707106781186548 \end{bmatrix}$$

U vektörü temel olarak A, V ve S vektörlerini kullanarak hesaplanır.

$$U = AVS^{-1} = \begin{bmatrix} 0.4472 & 0.8944 \\ 0.8944 & -0.4472 \end{bmatrix}$$

Jacobi metodunu kullanarak tekil değer ayrıştırması yapan ve lineer denklem sistemini çözen bir program oluşturulmuştur. Örnek problem olarak

$$\begin{bmatrix} 3 & 2 & -1 \\ 2 & -2 & 4 \\ -1 & 0.5 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix}$$

Denklem sistemini çözelim.

Program 9.8.1 Tekil değer ayrıştırması ile lineer denklem sistemi çözümü Jacobi sınır değer metodu kullanıyor.

```
 import java.io.*;
import javax.swing.*;

class SVD
{
double U[][];
double V[][];
double S[][];
double invS[][];

public static double[][] Transpose(double [][] left)
{
//transpose matrix (if A=a(i,j) Transpose(A)=a(j,i)
```

```
int i,j;
int n=left.length;
int m=left[0].length;
double b[][];
b=new double[m][n];
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
{
b[j][i]=left[i][j];
}
}
return b;
}

public static double[][] T(double [][] left)
{return Transpose(left);}

public static double[][] I(int n)
{
//unit matrix
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
b[i][i]=1.0;
return b;
}

public static double[][] S(double[] left)
{
int n=left.length;
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
b[i][i]=left[i];
return b;
}

public static double[][] S(double left,int n)
{
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
b[i][i]=left;
return b;
}
//ters S matrisi
public static double[][] invS(double[][] S)
{
int n=S.length;
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
{  if(S[i][i]==0) b[i][i]=0;
else b[i][i]=1.0/S[i][i];
}
return b;
}
//köşe matrisin kare kökü
public static double[][] sqrtS(double[] left)
{
int n=left.length;
```

```
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
b[i][i]=Math.sqrt(left[i]);
return b;
}
//iki matris çarpımı
public static double[][] carp(double[][] left,double[][] right)
{
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right[0].length;
int n2=right.length;
double[][] b;
b=new double[m1][n2];
if(n1 != m2)
{
System.out.println("m1="+m1+"n1="+n1+"m2="+m2+"n2="+n2+"inner matrix dimensions must agree");
for(ii=0;ii<n1;ii++)
{
for(jj=0;jj<m2;jj++)
b[ii][jj]=0;
}
return b;
}
for(i=0;i<m1;i++)
{
for(j=0;j<n2;j++)
{
for(k=0;k<n1;k++)
b[i][j]+=left[i][k]*right[k][j];
}
}
return b;
//end of carp of two matrices
}
//carp a matrix with a double value
public static double[][] carp(double[][] left,double right)
{
//carping a matrix with a constant
int i,j;
int n=left.length;
int m=left[0].length;
double b[][];
b=new double[n][m];
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
b[i][j]=left[i][j]*right;
}
return b;

}
//matris vektör çarpımı
public static double[] carp(double[][] left,double[] right)
{
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right.length;
double[] b;
```

```
b=new double[m2];
if(n1 != m2)
{
System.out.println("matris boyutları aynı olmalıdır");
for(ii=0;ii<n1;ii++)
{
b[ii]=0;
}
return b;
}
for(i=0;i<m1;i++)
{
b[i]=0;
for(k=0;k<n1;k++)
b[i]+=left[i][k]*right[k];
}
return b;
}

//iki matrisin farkı
public static double[][] substract(double[][] left,double[][] right)
{
int n1=left.length;
int m1=left[0].length;
int n2=right.length;
int m2=right[0].length;
int nMax,mMax;
int i,j;
if(m1>=m2) mMax=m1;
else      mMax=m2;
if(n1>=n2) nMax=n1;
else      nMax=n2;
double b[][];
b=new double[nMax][mMax];
for(i=0;i<n1;i++)
{
for(j=0;j<m1;j++)
{
b[i][j]=b[i][j]+left[i][j];
}
}
for(i=0;i<n2;i++)
{
for(j=0;j<m2;j++)
{
b[i][j]=b[i][j]-right[i][j];
}
}
return b;
}

public SVD(double A[][])
{ilkdeger(A);}

public static double[] coz(double a[][],double b[])
{ SVD q=new SVD(a);
double x[]=carp(carp(q.V,q.invS),carp(SVD.T(q.U),b));
return x;
}

public  double[] coz(double b[])
{
```

```
double x[]=carp(carp(V,invS),carp(T(U),b));
return x;
}

public void ilkdeger(double A[][])
{
double ATA[][]=carp(Transpose(A),A);
//double aa[][]=Householder(ATA);
double c[][];
c=eigen_jacobi(ATA);//eigenQL(aa);
int n=c[0].length;
int n1=c.length;
double dummy[]=new double[n1];
for(int i=0;i<n;i++)
{for(int k=i+1;k<n;k++)
{if(c[0][i]<c[0][k])
{ for(int j=0;j<n1;j++)
{ dummy[j]=c[j][k];
c[j][k]=c[j][i];
c[j][i]=dummy[j];
}
}
}
}
double x[]=new double[n];
for(int i=0;i<n;i++)
{x[i]=c[0][i];}
S=sqrtS(x);
invS=invS(S);
double xx[][]=new double[n][n];
V=new double[n][n];
U=new double[n][n];
double L[]=new double[n];
for(int i=0;i<n;i++)
{ for(int j=0;j<n;j++)
{xx[i][j]=c[i+1][j];}
}

double bb[]=new double[n];
for(int j=0;j<n;j++)
{ for(int i=0;i<n;i++)
{L[j]+=xx[i][j]*xx[i][j];}
L[j]=Math.sqrt(L[j]);
}
for(int i=0;i<n;i++)
{ for(int j=0;j<n;j++)
{V[i][j]=xx[i][j]/L[i];}
}
U=carp(A,carp(V,invS));
}

public static void print(String s)
{JOptionPane.showMessageDialog(null,s,
"Tekil değer ayrıştırması SVD : ",JOptionPane.PLAIN_MESSAGE);}

public static double[][] eigen_jacobi(double A[][])
{
double eps=1e-15;
double err=1;
int m=100;
int n=A.length;
double P[][]=new double[n][n];
```

```
double R[][]=new double[n][n];
int p=n,q=n;
double max=-9.99e100;
double fi;
P=I(n);
for(int i=0;i<n;i++)
{for(int j=i+1;j<n;j++)
{if(Math.abs(A[i][j]) > max) {max=Math.abs(A[i][j]);p=i;q=j;}}
}
while(Math.abs(A[p][q]) > eps )
{
fi=0.5*Math.atan(2.0*A[p][q]/(A[p][p]-A[q][q]));
R=I(n);
R[p][p]=Math.cos(fi);
R[q][q]=Math.cos(fi);
R[p][q]=-Math.sin(fi);
R[q][p]=Math.sin(fi);
P=carp(P,R);
A=carp(T(R),carp(A,R));
max=-9.99e100;
for(int i=0;i<n;i++)
{for(int j=i+1;j<n;j++){if(Math.abs(A[i][j])>max) {max=Math.abs(A[i][j]);p=i;q=j;}}}
}
double lambda[]=new double[n];
for(int i=0;i<n;i++) {lambda[i]=A[i][i];}
double PP[][]=new double[n+1][n];
PP[0]=lambda;
for(int i=0;i<n;i++)
{for(int j=0;j<n;j++)
{PP[i+1][j]=P[i][j];}
}
return PP;
}

public static void main(String args[]) throws IOException
{
double a[][]={{3,2,-1},{2,-2,4},{-1,0.5,-1}};
double b[]={1,-2,0};
//Dynamic call to SVD
SVD q=new SVD(a);
double x[]=q.coz(b);
System.out.println("S:\n"+Matrix.toString(q.S));
System.out.println("V:\n"+Matrix.toString(q.V));
System.out.println("U:\n"+Matrix.toString(q.U));
//Static çağırma SVD
//double x2[]=coz(a,b);
System.out.println("TEKİL DEĞER AYRIŞTIRMASI SVD  x=  \n"+Matrix.toStringT(x));
}
}
```

Singular value decomposition linear system of equation solving method by using Jacobi eigenvalue method results

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SVD
S:
     5.126196943738645      0.000000000000000      0.000000000000000
     0.000000000000000      3.734641762863597      0.000000000000000
```

```
        0.000000000000000       0.000000000000000       0.156702894292504

V:
        0.347022679391807       0.881301927095452      -0.320752510957550
       -0.450285808796899       0.456574244972559       0.767321737750118
        0.822689461694906      -0.121847741569467       0.555279368868110

U:
       -0.133079756513558       0.985074940622975       0.109165652853519
        0.953021289812956       0.096947557746556       0.286969671237626
       -0.272103288421038      -0.142227045260808       0.951699147854431

TEKİL DEĞER AYRIŞTIRMASI SVD  x =
        1.000000000000017000000000
       -2.000000000000042000000000
       -2.000000000000030000000000


> Terminated with exit code 0.
```

9.9 Tekil değer ayrıştırması kullanarak polinom en küçük kareler metodu eğri uydurma

Tekil değer ayrıştırması bir çok alanda kullanılabilen bir yöntemdir. Bir örneği en küçük kareler yöntemi çözme metodu olarak kullanılmasıdır. *A* matrisini *m* × *n* boyutlu bir matris olarak alalım, elbette *m => n*, olması şartı mevcuttur. Ve **b** bir vektördür. Polinomun derecesi (n-1) olacaktır. Bu matrisi oluşturmak $[x_i,y_i]$ i=0..(m-1veri seti gereklidir. A ve b

$$A=\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ \dots & \dots & .. & \dots & \dots \\ 1 & x_{m-2} & x_{m-2}^2 & \dots & x_{m-2}^{n-1} \\ 1 & x_{m-1} & x_{m-1}^2 & \dots & x_{m-1}^{n-1} \end{bmatrix} \quad b=\begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_{m-2} \\ y_{m-1} \end{bmatrix}$$

Eklinde yazılabilir. Görüldüğü gibi matrisler Vandemonde matrisine benzemektedir, ancak burada Vandemonde matrisindeki gibi satır ve sütun sayısı eşitliği gerekmemektedir. En küçük kareler objektifi $||A\mathbf{x}-\mathbf{b}||_2$ normunu minimize edecek bir x vektörü bulmaktır. A vektörünün tekil deer ayrıştırmasınınyapıldığını varsayalım, A = U SV^T, Bu durumda

$$\|Ax-b\|_2=\|USV^Tx-UU^Tb\|_2=\|SV^Tx-U^Tb\|_2$$

yazılabilir. If $z=V^Tx$ and $c=U^Tb$ tanımlanırsa denklem:

$$\|Ax-b\|_2=\|Sz-c\|_2=\left\{\sum_{i=1}^{k}(s_iz_i-c_i)^2+\sum_{i=k+1}^{m}(c_i)^2\right\}$$

Halini alır. Norm türevi 0 a eşitlenirse minimize edilmiş olur.

$$2(s_i z_i - c_i)\frac{dz_i}{dx_i} = 2(s_i z_i - c_i)V^T = 0 \quad i = 1..k$$

Bu durumda z_i

$$z_i = \left\{ \begin{array}{ll} \left(\frac{c_i}{s_i}\right) & i <= k \\ herhangibirsayı & i > k \end{array} \right\}$$

Şeklinde seçilebilir. Buradan en küçük kareler x=Vz olarak hesaplanır. Bu metodu uygulayan bilgisayar kodu altta verilmiştir.

Program 9.9.1 Tekil değer ayrıştırmasıyla en küçük kareler metodu

```
  import java.io.*;
import java.util.*;
import javax.swing.*;

class SVD_EKK
{

public static double[][] Transpose(double [][] left)
{
//transpose matrix (if A=a(i,j) Transpose(A)=a(j,i)
int i,j;
int n=left.length;
int m=left[0].length;
double b[][];
b=new double[m][n];
for(i=0;i<n;i++)
 {
 for(j=0;j<m;j++)
   {
   b[j][i]=left[i][j];
   }
 }
return b;
}

public static double[][] T(double [][] left)
{return Transpose(left);}

public static double[][] I(int n)
{
//unit matrix
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
   b[i][i]=1.0;
return b;
}
//Converts vector to diagonal values of matrix
public static double[][] S(double[] left,int m)
{
int n=left.length;
```

```
double b[][];
b=new double[m][n];
for(int i=0;i<n;i++)
   b[i][i]=left[i];
return b;
}

//Converts double to diagonal values of matrix
public static double[][] S(double left,int n)
{
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
   b[i][i]=left;
return b;
}

//takes invert value of a diagonal matrix
public static double[][] invS(double[][] S)
{
int m=S.length;
int n=S[0].length;
double b[][];
b=new double[m][n];
for(int i=0;i<n;i++)
   {  if(S[i][i]==0) b[i][i]=0;
                else b[i][i]=1.0/S[i][i];
              }
return b;
}

//square root of a diagonal matrix
public static double[][] sqrtS(double[] left,int m)
{
int n=left.length;
double b[][];
b=new double[m][n];
for(int i=0;i<n;i++)
   b[i][i]=Math.sqrt(left[i]);
return b;
}
//multiply two matrices
public static double[][] multiply(double[][] left,double[][] right)
{
//multiplication of two matrices
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right[0].length;
int n2=right.length;
double[][] b;
b=new double[n1][m2];
 if(m1 !=n2 )
 {
 System.out.println("inner matrix dimensions must agree"+"m1="+m1+"n2="+n2);
```

```
 return b;
 }
 for(i=0;i<n1;i++)
  {
  for(j=0;j<m2;j++)
    {
    for(k=0;k<m1;k++)
                {b[i][j]+=left[i][k]*right[k][j];
                }
    }
  }
return b;
//end of multiply of two matrices
}

//multiply a matrix with a double value
public static double[][] multiply(double[][] left,double right)
{
//multiplying a matrix with a constant
int i,j;
int n=left.length;
int m=left[0].length;
double b[][];
b=new double[n][m];
for(i=0;i<n;i++)
 {
 for(j=0;j<m;j++)
   b[i][j]=left[i][j]*right;
 }
return b;
//end of multiplying a matrix with a constant double
}

public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int n1=left.length;
int m1=left[0].length;

int m2=right.length;
double[] b;
b=new double[m2];
 for(i=0;i<n1;i++)
  {
    b[i]=0;
    for(k=0;k<m1;k++)
              {b[i]+=left[i][k]*right[k];
              }
  }
return b;
//end of multiply of a matrix and a vector
}

//substract two matrices
```

```
public static double[][] substract(double[][] left,double[][] right)
{
//addition of two matrices
int n1=left.length;
int m1=left[0].length;
int n2=right.length;
int m2=right[0].length;
int nMax,mMax;
int i,j;
if(m1>=m2) mMax=m1;
else      mMax=m2;
if(n1>=n2) nMax=n1;
else      nMax=n2;
double b[][];
b=new double[nMax][mMax];
for(i=0;i<n1;i++)
 {
 for(j=0;j<m1;j++)
   {
   b[i][j]=b[i][j]+left[i][j];
   }
 }
for(i=0;i<n2;i++)
 {
 for(j=0;j<m2;j++)
   {
   b[i][j]=b[i][j]-right[i][j];
   }
 }
return b;
//end of matrix substraction method
}

public static double[] SVD_EKK(double xi[],double yi[],int n_EKK)
{ // Singular value decomposition least square approximation
 double U[][];
 double V[][];
 double S[][];
 double invS[][];
 double A[][]=A(xi,n_EKK);
 int m=A.length;
 int m1=A[0].length;
 System.out.println("A="+Matrix.toString(A));
 double ATA[][]=multiply(Transpose(A),A);
 System.out.println("ATA="+Matrix.toString(ATA));
 //double aa[][]=Householder(ATA);
 double cc[][];
 cc=eigen_jacobi(ATA);//eigenQL(aa);
  System.out.println("c="+Matrix.toString(cc));
 int n=cc[0].length;
 int n1=cc.length;
 double dummy[]=new double[n1];
 for(int i=0;i<n;i++)
 {for(int k=i+1;k<n;k++)
  {if(cc[0][i]<cc[0][k])
```

```
    { for(int j=0;j<n1;j++)
      { dummy[j]=cc[j][k];
        cc[j][k]=cc[j][i];
        cc[j][i]=dummy[j];
      }
    }
  }
 }
 double x[]=new double[n];
 for(int i=0;i<n;i++)
 {x[i]=cc[0][i];}
 S=sqrtS(x,m1);
 System.out.println("S="+Matrix.toString(S));
 invS=invS(S);
 System.out.println("invS="+Matrix.toString(invS));
 double xx[][]=new double[n][n];
 V=new double[n][n];
 U=new double[n][n];
 double L[]=new double[n];

 for(int i=0;i<n;i++)
 { for(int j=0;j<n;j++)
  {xx[i][j]=cc[i+1][j];}
 }

 double bb[]=new double[n];
 for(int j=0;j<n;j++)
 { for(int i=0;i<n;i++)
  {L[j]+=xx[i][j]*xx[i][j];}
  L[j]=Math.sqrt(L[j]);
 }
 for(int i=0;i<n;i++)
 { for(int j=0;j<n;j++)
  {V[i][j]=xx[i][j]/L[i];}
 }
 System.out.println("VT="+Matrix.toString(T(V)));
 U=multiply(A,multiply(V,invS));
  System.out.println("U="+Matrix.toString(U));
 double c[]=multiply(T(U),yi);
  System.out.println("c="+Matrix.toString(c));
  double z[]=new double[n];
  for(int i=0;i<n;i++) z[i]=c[i]/S[i][i];
  System.out.println("z="+Matrix.toString(z));
  double a[]=multiply(V,z);
  System.out.println("a="+Matrix.toString(a));
  return a;
}
public static double func(double e[],double x)
{
// this function calculates the value of
// least square curve fitting function
int n=e.length;
double ff;
if(n!=0.0)
 { ff=e[n-1];
```

```
  for(int i=n-2;i>=0;i--)
   { ff=ff*x+e[i]; }
  }
else
  ff=0;
return ff;
}

public static double error(double x[],double y[],double e[])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
 double total=0;
 for(k=0;k<n;k++)
 {
 total+=(y[k]-func(e,x[k]))*(y[k]-func(e,x[k]));
 }
total=Math.sqrt(total);
return total;
}

public static double[][]  funcSVD_EKK(double xi[],double yi[],int polinomkatsayisi,int aradegersayisi)
 {
              //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
              // ara değer sayısı 2 dir
               int n=xi.length;
              int nn=(n-1)*(aradegersayisi+1)+1;
              double z[][]=new double[2][nn];
   double E[]=SVD_EKK(xi,yi,polinomkatsayisi);
   System.out.println("coefficients :\n"+Matrix.toStringT(E));
   double dx=0;
   int k=0;
   int i;
   for(i=0;i<(n-1);i++)
     {z[0][k]=xi[i];z[1][k]=func(E,z[0][k]);k++;
     for(int j=0;j<aradegersayisi;j++)
     {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
     z[0][k]=z[0][k-1]+dx;z[1][k]=func(E,z[0][k]);k++;}
     }
     z[0][k]=xi[i];z[1][k]=func(E,z[0][k]);
   return z;
 }
  public static void print(String s)
  {JOptionPane.showMessageDialog(null,s,
    "Singular value decomposition : ",JOptionPane.PLAIN_MESSAGE);}

  public static double[][] eigen_jacobi(double A[][])
  {             //sınır değer hesabı
   double eps=1e-15;
   double err=1;
   int m=100;
   int n=A.length;
   double P[][]=new double[n][n];
   double R[][]=new double[n][n];
```

```
 int p=n,q=n;
 double max=-9.99e100;
 double fi;
 P=I(n);
 for(int i=0;i<n;i++)
 {for(int j=i+1;j<n;j++)
    {if(Math.abs(A[i][j]) > max) {max=Math.abs(A[i][j]);p=i;q=j;}}
 }
 while(Math.abs(A[p][q]) > eps )
 {
  fi=0.5*Math.atan2(2.0*A[p][q],(A[p][p]-A[q][q]));
  R=I(n);
  R[p][p]=Math.cos(fi);
  R[q][q]=Math.cos(fi);
  R[p][q]=-Math.sin(fi);
  R[q][p]=Math.sin(fi);
  P=multiply(P,R);
  A=multiply(T(R),multiply(A,R));
  max=-9.99e100;
  for(int i=0;i<n;i++)
 {for(int j=i+1;j<n;j++){if(Math.abs(A[i][j])>max) {max=Math.abs(A[i][j]);p=i;q=j;}}}
  }
 double lambda[]=new double[n];
 for(int i=0;i<n;i++) {lambda[i]=A[i][i];}
 double PP[][]=new double[n+1][n];
 PP[0]=lambda;
 for(int i=0;i<n;i++)
 {for(int j=0;j<n;j++)
   {PP[i+1][j]=P[i][j];}
 }
 return PP;
}

public static double[] gauss(double a[][],double b[])
{ // Gauss eleme
int n=b.length;
double x[]=new double[n];
double carpan=0;
double toplam=0;
double buyuk;
double dummy=0;
//gauss elimination
int i,j,k,p,ii,jj;
for(k=0;k<(n-1);k++)
{ //partial pivoting
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
               if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
```

```
                a[k][jj]=dummy;
               }
               dummy=b[p];
               b[p]=b[k];
               b[k]=dummy;
              }
              //
   for(i=k+1;i<n;i++)
   {  carpan=a[i][k]/a[k][k];
      a[i][k]=0;
      for(j=k+1;j<n;j++)
      {   a[i][j]-=carpan*a[k][j]; }
         b[i]   =b[i]   -carpan*b[k];
             }
  }
  // backward substitution
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double[][] A(double xi[],int n_EKK)
{
int n=xi.length;
int m=n_EKK+1;
int l=xi.length;
int i,j,k;
double A[][];
A=new double[n][m];
double B[];
B=new double[n];
double X[];
X=new double[n];
double cc=1.0;
for(i=0;i<n;i++)
  {
     cc=1.0;
              for(j=0;j<m;j++)
    {
              A[i][j]=cc;
              cc*=xi[i];
    }
  }
return A;
}

  public static void main(String args[]) throws IOException
  {
  double x[];
```

```
 double y[];
 String s1=JOptionPane.showInputDialog("veri dosyası ismi (a.txt)");
 // JFileChooser fc=new JFileChooser();
 // if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName(); }
 double a[][]=Text.readDoubleT(s1);
 x=a[0];
 y=a[1];
 int n=Integer.parseInt(JOptionPane.showInputDialog("polinom derecesi (n)"));
 double z[][]=funcSVD_EKK(x,y,n,4);
 System.out.println("polinom en küçük kareler eğri uydurma\n"+Matrix.toString(Matrix.Transpose(z)));
 Plot pp=new Plot(a[0],a[1]);
 pp.setPlabel("Polinom tekil değer ayrıştırmal (SVD) en küçük kareler metodu");
 pp.setXlabel("x");
 pp.setYlabel("y=f(x)");
 pp.setPlotType(0,22);
 pp.addData(z[0],z[1]);
 pp.setPlotType(1,0);
 pp.setGrid(1,1);
 pp.plot();
 }
}
```

Girdi verisi:

```
 0.0     112.84
0.939    112.084811061789
1.56     111.59894162906366
1.881    109.43951145096956
2.814    103.56036629468154
3.178    101.61691074103722
3.988    91.66619441429282
4.039    91.17489545584903
4.131    89.61102039506228
4.44     84.88757503069911
4.726    79.70615657762187
4.965    75.5393519519086
5.689    61.3144994155904
6.9      34.0      5.34
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" SVD_EKK
A=      1.000000000000000      0.000000000000000      0.000000000000000      0.000000000000000      0.000000000000000      0.000000000000000
      1.000000000000000      0.939000000000000      0.881721000000000      0.827936019000000      0.777431921841000      0.730008574608699
      1.000000000000000      1.560000000000000      2.433600000000000      3.796416000000000      5.922408960000000      9.238957977600000
      1.000000000000000      1.881000000000000      3.538161000000000      6.655280841000001     12.518583261921002     23.547455115673404
      1.000000000000000      2.814000000000000      7.918596000000000     22.282929144000000     62.704162611216006    176.449513587961830
      1.000000000000000      3.178000000000000     10.099684000000000     32.096795752000000    102.003616899856000    324.167494507742330
      1.000000000000000      3.988000000000000     15.904144000000000     63.425726272000000    252.941796372736000   1008.731883934471200
      1.000000000000000      4.039000000000000     16.313520999999998     65.890311318999990    266.130967417440960   1074.902977399044000
      1.000000000000000      4.131000000000000     17.065161000000003     70.496180091000010    291.219719955921050   1203.028663137910000
      1.000000000000000      4.440000000000000     19.713600000000003     87.528384000000020    388.626024960000100   1725.499550822400600
      1.000000000000000      4.726000000000000     22.335076000000000    105.555569176000000    498.855619925776000   2357.591659769217600
      1.000000000000000      4.965000000000000     24.651225000000000    122.393332125000000    607.682894000625000   3017.145568713103000
      1.000000000000000      5.689000000000000     32.364721000000000    184.122897769000020   1047.475165407841100   5959.086216005208000
      1.000000000000000      6.900000000000000     47.610000000000010    328.509000000000070   2266.712100000000600  15640.313490000006000

ATA=      14.000000000000000      49.250000000000000      220.829210000000020      1093.580758508000100      5803.570491695174500
32520.433439544948000
     49.250000000000000      220.829210000000020      1093.580758508000100      5803.570491695174500      32520.433439544948000
```

190522.920032854450000
220.829210000000020 1093.580758508000100 5803.570491695174500 32520.433439544948000 190522.920032854480000
1157434.630377732500000
1093.580758508000100 5803.570491695174500 32520.433439544948000 190522.920032854480000 1157434.630377732200000
7238496.696487922000000
5803.570491695174500 32520.433439544948000 190522.920032854480000 1157434.630377732200000 7238496.696487922000000
46310263.822179720000000
32520.433439544948000 190522.920032854450000 1157434.630377732500000 7238496.696487922000000
46310263.822179720000000301525962.870225250000000

c=308820176.992987450000000 140620.223670170120000 217.777347878512780 5.121074059352600 0.738566316358630
0.032801888216496
0.000106993923435 -0.006120602521989 0.108086752734584 -0.663034748645800 0.736952628659654 -0.074602452105737
0.000626053674052 -0.024534188406045 0.248592931058318 -0.544356069662945 -0.460068603512538 0.655448385280658
0.003799619599305 -0.095600100405855 0.522324412932851 -0.270364923544877 -0.391623358035143 -0.701111506755876
0.023744702643023 -0.340328373912707 0.738086426486559 0.423374583566057 0.296886119534798 0.267308451157957
0.151826419941596 -0.922922654899264 -0.328617379093438 -0.107963891968719 -0.060925527193120 -0.042485381090190
0.988114409774552 0.150371551864179 0.030578773022157 0.007871445046489 0.003944715292266 0.002393282359140

S= 17573.280200150097000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000
0.000000000000000 374.993631506149900 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000
0.000000000000000 0.000000000000000 14.757281181793372 0.000000000000000 0.000000000000000 0.000000000000000
0.000000000000000 0.000000000000000 0.000000000000000 2.262979023179976 0.000000000000000 0.000000000000000
0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.859398811006060 0.000000000000000
0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.181112915653456

invS= 0.000056904572659 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000
0.000000000000000 0.002666711954503 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000
0.000000000000000 0.000000000000000 0.067763159601088 0.000000000000000 0.000000000000000 0.000000000000000
0.000000000000000 0.000000000000000 0.000000000000000 0.441895390879401 0.000000000000000 0.000000000000000
0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 1.163604123246743 0.000000000000000
0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 5.521417378721987

VT= 0.000106993923435 0.000626053674052 0.003799619599305 0.023744702643023 0.151826419941596 0.988114409774552
-0.006120602521989 -0.024534188406045 -0.095600100405855 -0.340328373912707 -0.922922654899264 0.150371551864179
0.108086752734584 0.248592931058318 0.522324412932851 0.738086426486559 -0.328617379093438 0.030578773022157
-0.663034748645800 -0.544356069662945 -0.270364923544877 0.423374583566057 -0.107963891968719 0.007871445046489
0.736952628659654 -0.460068603512538 -0.391623358035143 0.296886119534798 -0.060925527193120 0.003944715292266
-0.074602452105737 0.655448385280658 -0.701111506755876 0.267308451157957 -0.042485381090190 0.002393282359140

U= 0.000000006088443 -0.000016321883914 0.007324299876317 -0.292991999419461 0.857521117345899 -0.411911275551890
0.000049112678153 -0.002674599959625 0.079960079889320 -0.503863625563200 0.187296755856183 0.622323165010934
0.000576375082192 -0.015055518020789 0.196879966452320 -0.499150284553907 -0.152532922082322 0.148977671813185
0.001442018054423 -0.028449324733704 0.267133125107088 -0.438399495162974 -0.242047905266337 -0.104010733921390
0.010495110263203 -0.106011924281933 0.418800914980734 -0.124884176738417 -0.194945908953828 -0.371541301838679
0.019154307736835 -0.152986467633865 0.423934381744930 0.001912544580568 -0.101466370952060 -0.279856251304806
0.058993813568520 -0.279927715079148 0.267324229609947 0.154923146080787 0.084396561535102 0.059805832061299
0.062831866741354 -0.288198497381644 0.249354973661742 0.155765935653006 0.090662478747960 0.077653850093984
0.070259293428937 -0.302945792462177 0.214682316760166 0.154247958438513 0.099513190656125 0.106592575561583
0.100502056217252 -0.349322691511117 0.079055173262204 0.120216840591619 0.103866895256680 0.165179837920796
0.137020752125792 -0.384196648709247 -0.066552968506399 0.050606491306805 0.070631360577851 0.153909201607451
0.175069786058912 -0.403446757284115 -0.195117649035849 -0.031320641625310 0.016264288537343 0.090856309024069
0.344374634686152 -0.364175601150877 -0.519822799765418 -0.327039046143298 -0.235867082713874 -0.331472419986170
0.899465063504796 0.382220893365691 0.172110558094118 0.079542624730465 0.050043754293313 0.059982071893927

c= 105.067757400044330 -210.789422940325440 174.568931430187120 -169.207937805285800
71.579023010015960 -1.417405780206394 0.000000000000000 0.000000000000000
0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000
0.000000000000000 0.000000000000000

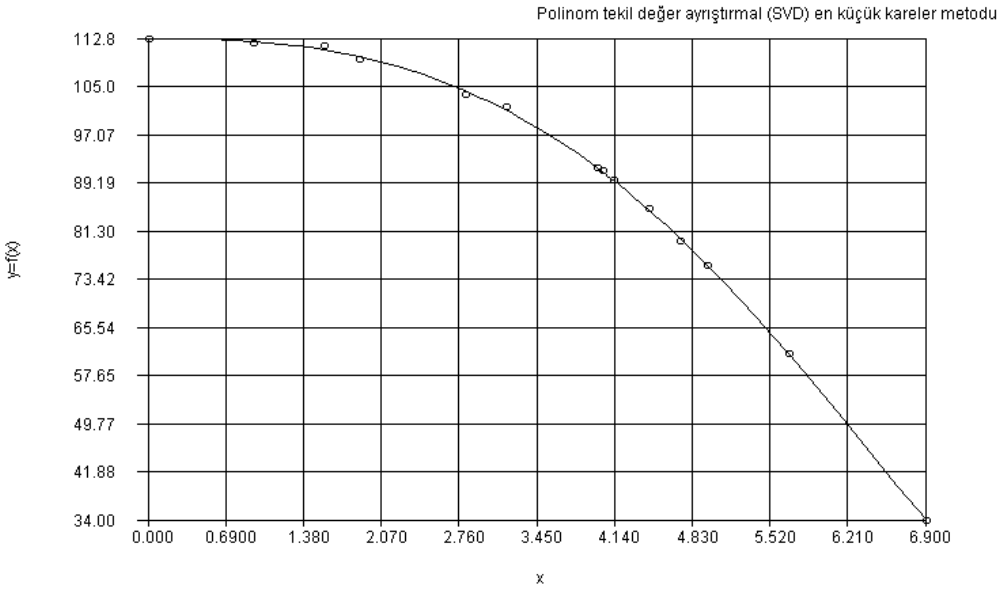
z= 0.005978835835051 -0.562114674037787 11.829342361895195 -74.772207816364100
83.289646312428060 -7.826088907532581

a= 112.822977551264230 0.208642224046439 -0.682912501775864 -0.098569589718137 -
0.036904045486362 0.004367069208825

coefficients :
112.822977551264230
0.208642224046439
-0.682912501775864
-0.098569589718137
-0.036904045486362
0.004367069208825

polinom en küçük kareler eğri uydurma

0.000000000000000	112.822977551264230
0.187800000000000	112.837377270217910
0.375600000000000	112.799076618796620
0.563400000000000	112.702658807783790
0.751200000000000	112.541850175499420
0.939000000000000	112.309642606922000
1.063200000000000	112.113158416054770
1.187400000000000	111.879796672979750
1.311600000000000	111.607152429799170
1.435800000000000	111.292758052462120
1.560000000000000	110.934098708192500
1.624200000000000	110.730513836589280
1.688400000000000	110.514067663269120
1.752600000000000	110.284405794630620
1.816800000000000	110.041173821886350
1.881000000000000	109.784017892602550
2.067600000000000	108.954266232347400
2.254200000000000	107.995360959302130
2.440800000000000	106.898988605733340
2.627400000000000	105.657194105140770
2.814000000000000	104.262499349889780
2.886800000000000	103.675402255039150
2.959600000000000	103.063596417132100
3.032400000000000	102.426718726169180
3.105200000000000	101.764424758633110
3.178000000000000	101.076389849077710
3.340000000000000	99.451448539595830
3.502000000000000	97.694625992461480
3.664000000000000	95.803570533022140
3.826000000000000	93.776584466235650
3.988000000000000	91.612682548460840
3.998200000000000	91.471845887522890
4.008400000000000	91.330465376671920
4.018600000000000	91.188541081899020
4.028800000000000	91.046073082345690
4.039000000000000	90.903061470361700
4.057400000000000	90.643705275532210
4.075799999999999	90.382581175204280
4.094199999999999	90.119690041408690
4.112599999999999	89.855032889479870
4.131000000000000	89.588610879161210
4.192800000000000	88.680880279617210
4.254600000000000	87.753318411812060
4.316400000000000	86.806006075761160
4.378200000000000	85.839043674678450
4.440000000000000	84.852551687380540
4.497200000000000	83.922206922696280
4.554400000000000	82.975380141301060
4.611600000000000	82.012214945507890
4.668800000000000	81.032871006045330
4.726000000000000	80.037524382944030
4.773800000000000	79.193613906672620
4.821599999999999	78.338784121103560
4.869399999999999	77.473165066287560
4.917199999999998	76.596895349388760
4.965000000000000	75.710122275455430
5.109800000000000	72.961373339872500
5.254600000000000	70.122577684156510
5.399400000000000	67.199372624170550
5.544200000000000	64.198216665447970
5.689000000000000	61.126422862277394
5.931200000000000	55.853354604306970
6.173400000000001	50.450198110619740
6.415600000000001	44.968810658032180
6.657800000000002	39.469134245763030
6.900000000000000	34.019632354585994



(bolum9_001.jpg,Resimaltı: **tekil değer ayrıştırmalı EKK**)

Aynı kavram daha genel en küçük kareler metodu uygulaması için de kullanılabilir. Eğer genel en küçük kareler metodu eğrisi

$$f(x) = \sum_{i=0}^{n} a_i \phi_i(x)$$

Denklemiyle tanımlanmışsa A matrisi

$$A = \begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \phi_2(x_0) & \dots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_n(x_1) \\ \dots & \dots & .. & \dots & \dots \\ \phi_0(x_{m-1}) & \phi_1(x_{m-1}) & \phi_2(x_{m-1}) & \dots & \phi_n(x_{m-1}) \\ \phi_0(x_m) & \phi_1(x_m) & \phi_2(x_m) & \dots & \phi_n(x_m) \end{bmatrix}$$

Formunu alır. Problemin geri kalan kısmı üstte tanımladığımızın aynıdır. Şüphesiz bu fonksiyon tanımı daha genel olduğundan üstte tanımladığımız polinom tanımını da içerir. ($\phi_0(x) = 1,\ \phi_1(x) = x,\ \phi_2(x) = x^2, \dots$). Programın değiştirilmiş yeni versiyonu aşağıda verilmiştir.

Program 9.9.2 General linear least square solution by using singular value decomposition

```
import java.io.*;
import java.util.*;
import javax.swing.*;

abstract class f_xr
{  abstract double func(double x,int equation_ref);}

class fa extends f_xr
{
            double func(double x,int i)
            {
    double xx=1.0;
     if(i==0) xx=1.0;
     else if(i==1) xx=x;
     else if(i==2) xx=x*x;
     else if(i==3) xx=x*x*x;
     else if(i==4) xx=x*x*x*x;
```

```
      else if(i==5) xx=x*x*x*x*x;
      else if(i==6) xx=x*x*x*x*x*x;
      else if(i==7) xx=x*x*x*x*x*x*x;
      else if(i==8) xx=x*x*x*x*x*x*x*x;
      else if(i==9) xx=x*x*x*x*x*x*x*x*x;

   return xx;
              }
}

class SVDGeneralleastsquare
{

public static double[][] Transpose(double [][] left)
{
//transpose matrix (if A=a(i,j) Transpose(A)=a(j,i)
int i,j;
int n=left.length;
int m=left[0].length;
double b[][];
b=new double[m][n];
for(i=0;i<n;i++)
  {
  for(j=0;j<m;j++)
    {
    b[j][i]=left[i][j];
    }
  }
return b;
}

public static double[][] T(double [][] left)
{return Transpose(left);}

public static double[][] I(int n)
{
//unit matrix
double b[][];
b=new double[n][n];
for(int i=0;i<n;i++)
    b[i][i]=1.0;
return b;
}
//Converts vector to diagonal values of matrix
public static double[][] S(double[] left,int m)
{
int n=left.length;
double b[][];
b=new double[m][n];
for(int i=0;i<n;i++)
    b[i][i]=left[i];
return b;
}

//Converts double to diagonal values of matrix
public static double[][] S(double left,int n)
{
double b[][];
```

```
b=new double[n][n];
for(int i=0;i<n;i++)
   b[i][i]=left;
return b;
}

//takes invert value of a diagonal matrix
public static double[][] invS(double[][] S)
{
int m=S.length;
int n=S[0].length;
double b[][];
b=new double[m][n];
for(int i=0;i<n;i++)
   {  if(S[i][i]==0) b[i][i]=0;
                else b[i][i]=1.0/S[i][i];
              }
return b;
}

//square root of a diagonal matrix
public static double[][] sqrtS(double[] left,int m)
{
int n=left.length;
double b[][];
b=new double[m][n];
for(int i=0;i<n;i++)
   b[i][i]=Math.sqrt(left[i]);
return b;
}
//multiply two matrices
public static double[][] multiply(double[][] left,double[][] right)
{
//multiplication of two matrices
int ii,jj,i,j,k;
int m1=left[0].length;
int n1=left.length;
int m2=right[0].length;
int n2=right.length;
double[][] b;
b=new double[n1][m2];
 if(m1 !=n2 )
 {
 System.out.println("inner matrix dimensions must agree"+"m1="+m1+"n2="+n2);
 return b;
 }
 for(i=0;i<n1;i++)
   {
   for(j=0;j<m2;j++)
     {
     for(k=0;k<m1;k++)
                {b[i][j]+=left[i][k]*right[k][j];
                }
     }
   }
return b;
//end of multiply of two matrices
}
```

```
//multiply a matrix with a double value
public static double[][] multiply(double[][] left,double right)
{
//multiplying a matrix with a constant
int i,j;
int n=left.length;
int m=left[0].length;
double b[][];
b=new double[n][m];
for(i=0;i<n;i++)
  {
  for(j=0;j<m;j++)
    b[i][j]=left[i][j]*right;
  }
return b;
//end of multiplying a matrix with a constant double
}

public static double[] multiply(double[][] left,double[] right)
{
//multiplication of one matrix with one vector
int ii,jj,i,j,k;
int n1=left.length;
int m1=left[0].length;

int m2=right.length;
double[] b;
b=new double[m2];
 for(i=0;i<n1;i++)
   {
     b[i]=0;
     for(k=0;k<m1;k++)
               {b[i]+=left[i][k]*right[k];
               }
   }
return b;
//end of multiply of a matrix and a vector
}

//substract two matrices
public static double[][] substract(double[][] left,double[][] right)
{
//addition of two matrices
int n1=left.length;
int m1=left[0].length;
int n2=right.length;
int m2=right[0].length;
int nMax,mMax;
int i,j;
if(m1>=m2) mMax=m1;
else       mMax=m2;
if(n1>=n2) nMax=n1;
else       nMax=n2;
double b[][];
b=new double[nMax][mMax];
for(i=0;i<n1;i++)
  {
```

```
  for(j=0;j<m1;j++)
    {
    b[i][j]=b[i][j]+left[i][j];
    }
  }
for(i=0;i<n2;i++)
  {
  for(j=0;j<m2;j++)
    {
    b[i][j]=b[i][j]-right[i][j];
    }
  }
return b;
//end of matrix substraction method
}

public static double[] SVD_LSQ(f_xr f,double xi[],double yi[],int n_leastsquare)
{ // Singular value decomposition least square approximation
  double U[][];
  double V[][];
  double S[][];
  double invS[][];
  double A[][]=A(f,xi,n_leastsquare);
  int m=A.length;
  int m1=A[0].length;
  System.out.println("A="+Matrix.toString(A));
  double ATA[][]=multiply(Transpose(A),A);
  System.out.println("ATA="+Matrix.toString(ATA));
  //double aa[][]=Householder(ATA);
  double cc[][];
  cc=eigen_jacobi(ATA);//eigenQL(aa);
   System.out.println("c="+Matrix.toString(cc));
  int n=cc[0].length;
  int n1=cc.length;
  double dummy[]=new double[n1];
  for(int i=0;i<n;i++)
  {for(int k=i+1;k<n;k++)
    {if(cc[0][i]<cc[0][k])
      { for(int j=0;j<n1;j++)
        { dummy[j]=cc[j][k];
          cc[j][k]=cc[j][i];
          cc[j][i]=dummy[j];
        }
      }
    }
  }
  double x[]=new double[n];
  for(int i=0;i<n;i++)
  {x[i]=cc[0][i];}
  S=sqrtS(x,m1);
  System.out.println("S="+Matrix.toString(S));
  invS=invS(S);
  System.out.println("invS="+Matrix.toString(invS));
  double xx[][]=new double[n][n];
  V=new double[n][n];
  U=new double[n][n];
  double L[]=new double[n];
```

```
  for(int i=0;i<n;i++)
  { for(int j=0;j<n;j++)
    {xx[i][j]=cc[i+1][j];}
  }

  double bb[]=new double[n];
  for(int j=0;j<n;j++)
  { for(int i=0;i<n;i++)
    {L[j]+=xx[i][j]*xx[i][j];}
    L[j]=Math.sqrt(L[j]);
  }
  for(int i=0;i<n;i++)
  { for(int j=0;j<n;j++)
    {V[i][j]=xx[i][j]/L[i];}
  }
  System.out.println("VT="+Matrix.toString(T(V)));
  U=multiply(A,multiply(V,invS));
   System.out.println("U="+Matrix.toString(U));
  double c[]=multiply(T(U),yi);
   System.out.println("c="+Matrix.toString(c));
   double z[]=new double[n];
   for(int i=0;i<n;i++) z[i]=c[i]/S[i][i];
   System.out.println("z="+Matrix.toString(z));
   double a[]=multiply(V,z);
   System.out.println("a="+Matrix.toString(a));
   return a;
}
public static double func(f_xr f,double e[],double x)
{
// this function calculates the value of
// least square curve fitting function
int n=e.length;
double ff;
if(n!=0.0)
  { ff=0;
  for(int i=0;i<e.length;i++)
   { ff+=e[i]*f.func(x,i); }
  }
else
  ff=0;
return ff;
}

public static double error(f_xr f,double x[],double y[],double e[])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
 double total=0;
 for(k=0;k<n;k++)
 {
 total+=(y[k]-func(f,e,x[k]))*(y[k]-func(f,e,x[k]));
 }
total=Math.sqrt(total);
return total;
}

public static double[][]  funcSVD_LSQ(f_xr f,double xi[],double yi[],int polinomkatsayisi,int aradegersayisi)
```

```
{
            //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
            // ara değer sayısı 2 dir
             int n=xi.length;
            int nn=(n-1)*(aradegersayisi+1)+1;
            double z[][]=new double[2][nn];
  double E[]=SVD_LSQ(f,xi,yi,polinomkatsayisi);
  System.out.println("coefficients :\n"+Matrix.toStringT(E));
  double dx=0;
  int k=0;
  int i;
  for(i=0;i<(n-1);i++)
     {z[0][k]=xi[i];z[1][k]=func(f,E,z[0][k]);k++;
     for(int j=0;j<aradegersayisi;j++)
     {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
     z[0][k]=z[0][k-1]+dx;z[1][k]=func(f,E,z[0][k]);k++;}
     }
     z[0][k]=xi[i];z[1][k]=func(f,E,z[0][k]);
  return z;
}

 public static void print(String s)
 {JOptionPane.showMessageDialog(null,s,
   "Singular value decomposition : ",JOptionPane.PLAIN_MESSAGE);}

 public static double[][] eigen_jacobi(double A[][])
 {            //Eigen value calculation
  double eps=1e-15;
  double err=1;
  int m=100;
  int n=A.length;
  double P[][]=new double[n][n];
  double R[][]=new double[n][n];
  int p=n,q=n;
  double max=-9.99e100;
  double fi;
  P=I(n);
  for(int i=0;i<n;i++)
  {for(int j=i+1;j<n;j++)
     {if(Math.abs(A[i][j]) > max) {max=Math.abs(A[i][j]);p=i;q=j;}}
  }
  while(Math.abs(A[p][q]) > eps )
  {
   fi=0.5*Math.atan2(2.0*A[p][q],(A[p][p]-A[q][q]));
   R=I(n);
   R[p][p]=Math.cos(fi);
   R[q][q]=Math.cos(fi);
   R[p][q]=-Math.sin(fi);
   R[q][p]=Math.sin(fi);
   P=multiply(P,R);
   A=multiply(T(R),multiply(A,R));
   max=-9.99e100;
   for(int i=0;i<n;i++)
  {for(int j=i+1;j<n;j++){if(Math.abs(A[i][j])>max) {max=Math.abs(A[i][j]);p=i;q=j;}}}
   }
  double lambda[]=new double[n];
  for(int i=0;i<n;i++) {lambda[i]=A[i][i];}
  double PP[][]=new double[n+1][n];
```

```
 PP[0]=lambda;
 for(int i=0;i<n;i++)
 {for(int j=0;j<n;j++)
   {PP[i+1][j]=P[i][j];}
 }
 return PP;
}

public static double[] gausswithpartialpivot(double a[][],double b[])
{ // Gauss elimination with partial pivoting
int n=b.length;
double x[]=new double[n];
double carpan=0;
double toplam=0;
double buyuk;
double dummy=0;
//gauss elimination
int i,j,k,p,ii,jj;
for(k=0;k<(n-1);k++)
{ //partial pivoting
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
              if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
                a[k][jj]=dummy;
              }
              dummy=b[p];
              b[p]=b[k];
              b[k]=dummy;
            }
            //
  for(i=k+1;i<n;i++)
  {  carpan=a[i][k]/a[k][k];
     a[i][k]=0;
     for(j=k+1;j<n;j++)
     {   a[i][j]-=carpan*a[k][j]; }
        b[i]   =b[i]   -carpan*b[k];
            }
}
// backward substitution
x[n-1]=b[n-1]/a[n-1][n-1];
for(i=n-2;i>=0;i--)
{
  toplam=0;
  for(j=i+1;j<n;j++)
  {  toplam+=a[i][j]*x[j];}
  x[i]=(b[i]-toplam)/a[i][i];
}
return x;
}

public static double[][] A(f_xr f,double xi[],int n_leastsquare)
```

```
{
int n=xi.length;
int m=n_leastsquare+1;
int l=xi.length;
int i,j,k;
double A[][];
A=new double[n][m];
double B[];
B=new double[n];
double X[];
X=new double[n];
double cc=1.0;
for(i=0;i<n;i++)
  {  for(j=0;j<m;j++) { A[i][j]=f.func(xi[i],j);}}
return A;
}

  public static void main(String args[]) throws IOException
  {
  double x[];
  double y[];
  String s1=JOptionPane.showInputDialog("Data file name (a.txt)");
  // JFileChooser fc=new JFileChooser();
  // if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file = fc.getSelectedFile();s1=file.getName();
}
  double a[][]=Text.readDoubleT(s1);
  x=a[0];
  y=a[1];
  int n=Integer.parseInt(JOptionPane.showInputDialog("polinom derecesi (n)"));
  fa f1=new fa();
  double z[][]=funcSVD_LSQ(f1,x,y,n,4);
  System.out.println("Polynomial Least Square Curve fitting\n"+Matrix.toStringT(z));
  Plot pp=new Plot(a[0],a[1]);
  pp.setPlabel("Polynomial Singular Value Decomposition Least Square Curve fitting");
  pp.setXlabel("x");
  pp.setYlabel("y=f(x)");
  pp.setPlotType(0,22);
  pp.addData(z[0],z[1]);
  pp.setPlotType(1,0);
  pp.setGrid(1,1);
  pp.plot();
  }
}
```

```
---------- Capture Output ----------
> "D:\java\bin\javaw.exe" SVDGeneralleastsquare
A=      1.000000000000000       0.000000000000000       0.000000000000000
       1.000000000000000       0.250000000000000       0.062500000000000
       1.000000000000000       0.500000000000000       0.250000000000000
       1.000000000000000       0.750000000000000       0.562500000000000
       1.000000000000000       1.000000000000000       1.000000000000000

ATA=       5.000000000000000       2.500000000000000       1.875000000000000
       2.500000000000000       1.875000000000000       1.562500000000000
       1.875000000000000       1.562500000000000       1.382812500000000

c=       7.353236163159007       0.878108943148838       0.026467393692153
       0.798724502630908      -0.592863568473215       0.102722724234778
       0.471216204284530       0.510173089573159      -0.719498928070819
       0.374158332416491       0.623086035674515       0.686854667619525
```

```
S=      2.711685115045441       0.000000000000000       0.000000000000000
        0.000000000000000       0.937074673197840       0.000000000000000
        0.000000000000000       0.000000000000000       0.162688025656939

invS=       0.368774381085631       0.000000000000000       0.000000000000000
        0.000000000000000       1.067150813699214       0.000000000000000
        0.000000000000000       0.000000000000000       6.146733885066024

VT=       0.798724502630908       0.471216204284530       0.374158332416491
       -0.592863568473215       0.510173089573159       0.623086035674515
        0.102722724234778      -0.719498928070819       0.686854667619525

U=       0.294549134115641      -0.632674839508811       0.631409249820203
        0.346616000605336      -0.455009009469034      -0.210363306816900
        0.415930368028146      -0.194227333182423      -0.524396756007415
        0.502492236384071       0.149670189351023      -0.310691097751344
        0.606301605673112       0.576683558131304       0.430753667951314

c=       4.13723419578719600000000000            0.34732169503952750000000000            0.00991447389033983700000000
0.00000000000000000000000000        0.00000000000000000000000000

z=       1.52570597995773040000000000            0.37064462947682210000000000            0.06094163261435424400000000

a=       1.00513714285714000000000000            0.86418285714286660000000000            0.84365714285713820000000000

coefficients :
        1.00513714285714000000000000
        0.86418285714286660000000000
        0.84365714285713820000000000

Polynomial Least Square Curve fitting
  0.0000000000   1.0051371429
  0.0500000000   1.0504554286
  0.1000000000   1.0999920000
  0.1500000000   1.1537468571
  0.2000000000   1.2117200000
  0.2500000000   1.2739114286
  0.3000000000   1.3403211429
  0.3500000000   1.4109491429
  0.4000000000   1.4857954286
  0.4500000000   1.5648600000
  0.5000000000   1.6481428571
  0.5500000000   1.7356440000
  0.6000000000   1.8273634286
  0.6500000000   1.9233011429
  0.7000000000   2.0234571429
  0.7500000000   2.1278314286
  0.8000000000   2.2364240000
  0.8500000000   2.3492348571
  0.9000000000   2.4662640000
  0.9500000000   2.5875114286
  1.0000000000   2.7129771429
```

9.10 Problemler

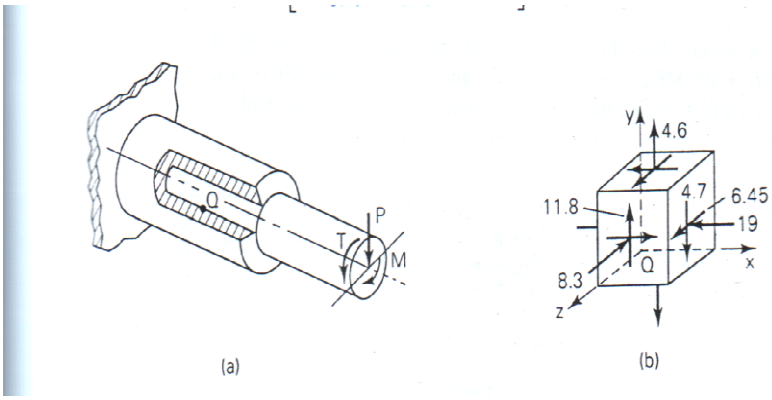
PROBLEM 1

Çelik bir mil dökme demirden yapılmış bir yatağın içine şekildeki gibi oturtulmuştur. Milin üzerine T burkulma torku ve M eğilme momenti uygulanmaktadır. Bu kuvvetler altında Q noktasındaki gerilme matrisi (Mpa)

-19 ***-4.7*** ***6.45***

-19	*-4.7*	*6.45*
-4.7	**4.6**	**11.8**
6.45	**11.8**	**-8.3**

olarak verilmiştir. Q noktasındaki asal gerilmeleri hesaplayan bir program yazınız.
Not : Asal gerilmeler gerilme matrisinin sınır değerleridir.



(bolum9_002.jpg,Resimaltı: **Problem 1**)

PROBLEM 2

$A=\begin{bmatrix}1 & 2\\ 3 & 5\end{bmatrix}$ matrisinin sınır değerlerini hesaplayınız.

PROBLEM 3

$A=\begin{bmatrix}1 & 0 & -1\\ -1 & 1 & 0\\ 0 & -1 & 1\end{bmatrix}$ matrisinin sınır değerlerini hesaplayınız.

PROBLEM 4

$A=\begin{bmatrix}3 & -2 & 0\\ 2 & 1 & 4\\ 1 & -1 & 2\end{bmatrix}$ matrisinin sınır değerlerini hesaplayınız.

PROBLEM 5

$A=\begin{bmatrix}4 & -1 & 3 & 9 & -5\\ -1 & 2 & 7 & 12 & 7\\ 3 & 7 & 0 & -11 & 8\\ 2 & -4 & 14 & 1 & -5\\ 11 & -8 & 6 & -2 & 13\end{bmatrix}$ matrisinin en büyük sınır değerini hesaplayınız

PROBLEM 6

$A=\begin{bmatrix}8 & -1 & -5\\ -4 & 4 & -2\\ 18 & -5 & -7\end{bmatrix}$ matrisinin sınır değerlerini ve sınır matrislerini bulunuz.

PROBLEM 7

$A=\begin{bmatrix}\sigma_x & \tau_{xy} & \tau_{xz}\\ \tau_{yx} & \sigma_y & \tau_{yz}\\ \tau_{zx} & \tau_{zy} & \sigma_z\end{bmatrix}=\begin{bmatrix}10 & 4 & -6\\ 4 & -6 & 8\\ -6 & 8 & 14\end{bmatrix}MPa$ normal ve kesme gerilmeleri verilmiştir. Asal gerilmeleri hesaplayınız. Not asal gerilmeler verilen gerilme matrisinin sınır değerleridir.

10.0Sonlu farklar metodu ile kısmi diferansiyel denklemlerin çözülmesi

10.1 Sonlu farklar metoduna giriş

Sonlu farklar metoduyla kısmi diferansiyel denklemlerin çözülmesi kısmifark kavramına dayanır. Türev konusunu incelerken türevlere sonlu fark denklemleriyle yaklaşmayı incelemiştik. Fark denklemleri temel olarak taylor formülüne dayanır.

$$f(x_{i+1})=f(x_i)+f'(x_i)h+\frac{f''(x_i)}{2}h^2+\frac{f^{(3)}(x_i)}{3!}h^3+\frac{f^{(4)}(x_i)}{4!}h^4+\dots$$

Eğer bu denklemden birinci türevi çekersek:

$$f'(x_i)=\frac{f(x_{i+1})-f(x_i)}{h}-\frac{f''(x_i)}{2}h-\frac{f^{(3)}(x_i)}{3!}h^2-\frac{f^{(4)}(x_i)}{4!}h^3+O(h^4)\dots$$

veya

$$f'(x_i)=\frac{f(x_{i+1})-f(x_i)}{h}+O(h)$$

O(h) terimi hata miktarını bildirir. Hata h fark teriminin fonksiyonu olarak değişir. İkinci türevi de aynı şekilde yazabiliriz.

$$f''(x_i)=\frac{f(x_{i+2})-2f(x_{i+1})+f(x_i)}{h^2}+O(h)$$

Hata miktarını düşürmek için örneğin ikinci türev formülünü tekrar temel Taylor denklemine geri koyacak olursak, ikinci dereceden bir birinci türev elde edebiliriz.

$$f'(x_i)=\frac{f(x_{i+1})-f(x_i)}{h}-\frac{f(x_{i+2})-2f(x_{i+1})+f(x_i)}{2h^2}h+O(h^2)$$

$$f'(x_i)=\frac{2f(x_{i+1})-2f(x_i)-f(x_{i+2})+2f(x_{i+1})-f(x_i)}{h}+O(h^2)$$

$$f'(x_i)=\frac{-f(x_{i+2})+4f(x_{i+1})-3f(x_i)}{2h}+O(h^2)$$

Fark denklemleri ileri doğru ve geriye doğru fark şeklinde hesaplanabilir. Geriye doğru fark denklemi: $f'(x_i)=\frac{f(x_i)-f(x_{i-1})}{h}+O(h)$. Geriye doğru fark ve ileriye doğru fark denklemlerinin ortalaması alınarak merkezi fark elde edilebilir.

$$f'(x_i)=\frac{f'(x_i)_{forward}+f'(x_i)_{backward}}{2}=\frac{\frac{f(x_i)-f(x_{i-1})}{h}}{2}+\frac{\frac{f(x_i)-f(x_{i-1})}{h}}{2}=\frac{f(x_{i+1})-f(x_{i-1})}{2h}+O(h^2)$$

Alttaki tabloda çeşitli fark denklemlerinin katsayıları bir tablo olarak verilmiştir.

Merkezi farklar

Türev	hassaslık	−4	−3	−2	−1	0	1	2	3	4
1	2				−1/2	0	1/2			

	4			1/12	−2/3	0	2/3	−1/12		
	6		−1/60	3/20	−3/4	0	3/4	−3/20	21916	
	8	1/280	−4/105	1/5	−4/5	0	4/5	−1/5	4/105	−1/280
2	2				1	−2	1			
	4			−1/12	4/3	−5/2	4/3	−1/12		
	6		32874	−3/20	3/2	−49/18	3/2	−3/20	32874	
	8	−1/560	8/315	−1/5	8/5	−205/72	8/5	−1/5	8/315	−1/560
3	2			−1/2	1	0	−1	1/2		
	4		41122	−1	13/8	0	−13/8	1	−1/8	
	6	−7/240	41185	−169/120	61/30	0	−61/30	169/120	−3/10	7/240
4	2			1	−4	6	−4	1		
	4		−1/6	2	−13/2	28/3	−13/2	2	−1/6	
	6	7/240	−2/5	169/60	−122/15	91/8	−122/15	169/60	−2/5	7/240

İleri doğru farklar

Türev	hassaslık	0	1	2	3	4	5	6	7	8
1	1	−1	1							
	2	−3/2	2	−1/2						
	3	−11/6	3	−3/2	1/3					
	4	−25/12	4	−3	4/3	−1/4				
	5	−137/60	5	−5	10/3	−5/4	41030			
	6	−49/20	6	−15/2	20/3	−15/4	41035	−1/6		
2	1	1	−2	1						
	2	2	−5	4	−1					
	3	35/12	−26/3	40958	−14/3	41254				
	4	41014	−77/6	107/6	−13	61/12	−5/6			
	5	203/45	−87/5	117/4	−254/9	33/2	−27/5	137/180		
	6	469/90	−223/10	879/20	−949/18	41	−201/10	1019/180	−7/10	
3	1	−1	3	−3	1					
	2	−5/2	9	−12	7	−3/2				
	3	−17/4	71/4	−59/2	49/2	−41/4	7/4			
	4	−49/8	29	−461/8	62	−307/8	13	−15/8		
	5	−967/120	638/15	−3929/40	389/3	−2545/24	268/5	−1849/120	29/15	
	6	−801/80	349/6	−18353/120	2391/10	−1457/6	4891/30	−561/8	527/30	−469/240
4	1	1	−4	6	−4	1				
	2	3	−14	26	−24	11	−2			
	3	35/6	−31	137/2	−242/3	107/2	−19	41077		
	4	28/3	−111/2	142	−1219/6	176	−185/2	82/3	−7/2	
	5	1069/80	−1316/15	15289/60	−2144/5	10993/24	−4772/15	2803/20	−536/15	967/240

Şidi bu fark denklemlerinin diferansiyel denklem çözüm yöntemi olarak nasıl kullanacağımıza bakalım: Bunun için Isı tansferinden bir örnek kısmi diferansiyel denklemi inceleyelim. Kısmi diferansiyel

denklemleri çözmek için sınır şartlarının da verilmiş olmaları gerekir. Bu diferansiyel denklemimiz iki boyutlu zamandan bağımsız bir kısmi diferansiyel denklemdir. Sınır şartları olarak T1,T2,T3,T4 gibi dört sıcaklık verilmiştir.

$$\frac{\partial^2 T(x,y)}{\partial x^2}+\frac{\partial^2 T(x,y)}{\partial y^2}=0$$

$T(x,y)=T_1$ x=0, $0 \le y \le W$

$T(x,y)=T_2$ x=L, $0 \le y \le W$

$T(x,y)=T_3$ y=0, $0 \le x \le L$

$T(x,y)=T_4$ y=W, $0 \le x \le L$

Şimdi diferansiyel denklemdeki terimleri fark denklemi terimleri ile değiştirelim:

$$\frac{\partial^2 T(x,y)}{\partial x^2} \cong \frac{T_{m+1,n}-2T_{m,n}+T_{m-1,n}}{\Delta x^2}$$

$$\frac{\partial^2 T(x,y)}{\partial y^2} \cong \frac{T_{m,n+1}-2T_{m,n}+T_{m,n-1}}{\Delta y^2}$$

$$\frac{T_{m+1,n}-2T_{m,n}+T_{m-1,n}}{\Delta x^2}+\frac{T_{m,n+1}-2T_{m,n}+T_{m,n-1}}{\Delta y^2}=0$$

Eğer $\Delta x = \Delta y$ olarak alınırsa denklem biraz daha sadeleştirilmiş olur.

$$T_{m+1,n}+T_{m-1,n}+T_{m,n+1}+T_{m,n-1}-4T_{m,n}=0$$

Ve şimdi de sınır şartlarını uygulamamız gerekir.

x=0 (m=0) $T_{0,n}=T_1$

x=L (m=M) $T_{M,n}=T_2$

y=0 (n=0) $T_{m,0}=T_3$

y=W (n=M) $T_{m,N}=T_4$

Denklem ve sınır şartları bir lineer denklem sistemi olarak oluşturulur ve denklem çözülerek her noktadaki çözüm seti (bu problem için sıcaklık profili)

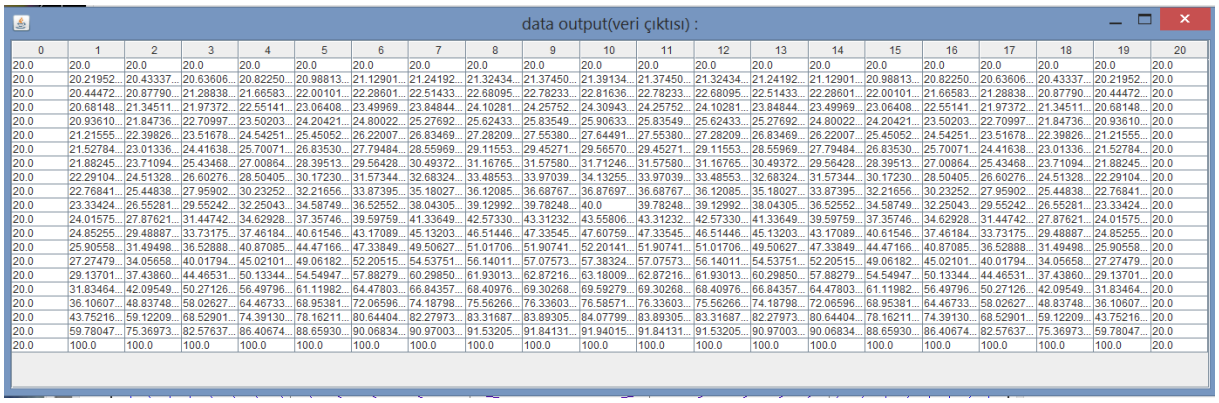
Program 10.1 Sonlu farklar yöntemi ile sıcaklık profilinin çözülmesi

```
import java.io.*;
import visad.*;
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;

class HT_plane_FD_T
{
  public static double [] gauss_seidel(double a[][],double b[],double x[],double lambda)
  {
  //denklem sistemi çözümü b iteratif gauss_seidel metodu metodu
  //yakınsama faktörü lambda ile birlikte verilmiş formül
  int n=a.length;
  int imax=1000;//maksimum iterasyon sayısı
  double es=1.0e-10;
  double dummy=0;
  double ea=1e-1;
```

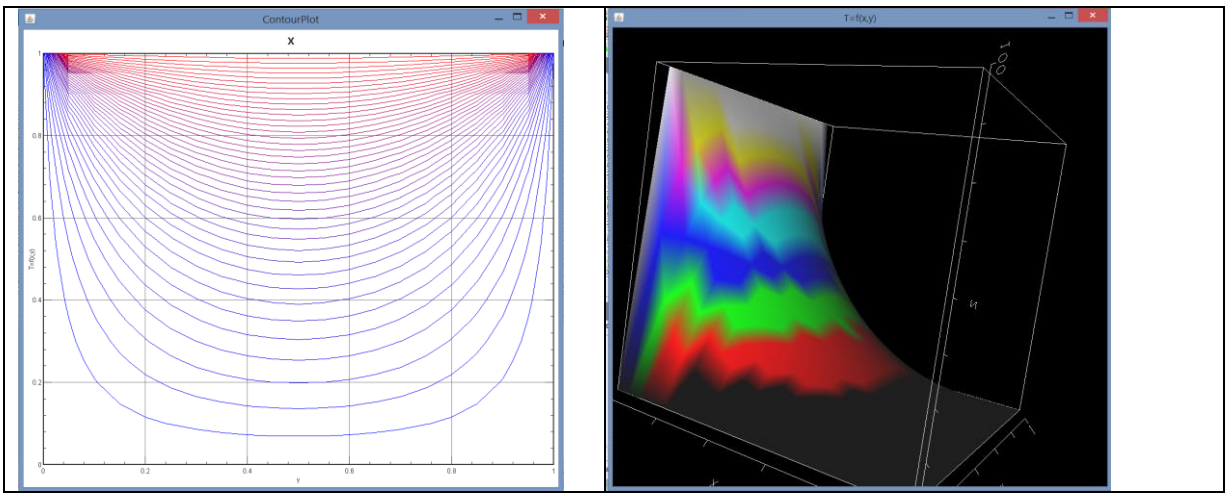
```
double sum=0;
double old=0;
int iter=0;
int sentinel=0;
//System.out.print("iter = "+(iter+1)+" ");
for(int i=0;i<n;i++)
{  dummy=a[i][i];
   for(int j=0;j<n;j++)
     {a[i][j]/=dummy;}
   b[i]/=dummy;
}
for(int i=0;i<n;i++)
{  sum=b[i];
   for(int j=0;j<n;j++)
     {if(i!=j)sum-=a[i][j]*x[j];}
   x[i]=sum;
   //System.out.print(x[i]+ "  ");
}
 //System.out.println();
iter++;
do
{
sentinel=1;
//System.out.print("iter = "+(iter+1)+" ");
for(int i=0;i<n;i++)
   {
              old=x[i];
              sum=b[i];
              for(int j=0;j<n;j++)
                 {if(i!=j) sum-=a[i][j]*x[j];}
                 x[i]=lambda*sum+(1.0-lambda)*old;
                 if(sentinel==1 && x[i]!=0.0 ) {ea=Math.abs((x[i]-old)/x[i])*100.0;}
                 if(ea<es) sentinel=0;
               //System.out.print(x[i]+ "  ");
   }
//System.out.println();
iter++;
}while(sentinel==1 && iter<imax);
if(iter>=imax) System.out.println("Maksimum iterasyon sayısı aşıldı sonuç geçerli olmayabilir iter = "+iter);
return x;
}

public static double[] pivotlugauss(double a[][],double b[])
{ //kısmi pivotlu gauss eleme yöntemi
int n=b.length;
double x[]=new double[n];
double carpan=0;
double toplam=0;
double buyuk;
double dummy=0;
//gauss eleme
int i,j,k,p,ii,jj;
//System.out.println("Orijinal Matris :\n"+Matrix.toString(a));
for(k=0;k<(n-1);k++)
{ //pivotlama
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
               if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
                a[k][jj]=dummy;
              }
              dummy=b[p];
              b[p]=b[k];
              b[k]=dummy;
            }
              //System.out.println("pivotlu Matris : k="+k+"\n"+Matrix.toString(a));
            //gauss elemeyi çözme
  for(i=k+1;i<n;i++)
  {  carpan=a[i][k]/a[k][k];
     a[i][k]=0;
```

```
      for(j=k+1;j<n;j++)
     {   a[i][j]-=carpan*a[k][j]; }
        b[i]   =b[i]   -carpan*b[k];
              }
}
//back substitution
x[n-1]=b[n-1]/a[n-1][n-1];
for(i=n-2;i>=0;i--)
{
  toplam=0;
  for(j=i+1;j<n;j++)
  {  toplam+=a[i][j]*x[j];}
  x[i]=(b[i]-toplam)/a[i][i];
}
return x;
}


public static double [][] HT_planecoz( double L,double W,double dx, double Tx0,double Txn,double Ty0,double Tyn)
{
int nx=(int)(L/dx+0.01);
int ny=(int)(W/dx+0.01);
int mx=(nx-1);
int my=(ny-1);
int mt=mx*my;
double A[][]=new double[mt][mt];
double b[]=new double[mt];
double x[]=new double[mt];
double x1,x2,y1,y2;
// boundary conditions
int i=0,j=0;
for(i=0;i<my;i++)
{
   for(j=0;j<mx;j++)
   {
              x1=dx*j;
              x2=(L-x1);
              y1=dx*i;
              y2=(W-y1);
              int k=i*my+j;
              x[k]=((x1*Tx0+x2*Txn)/L+(y1*Ty0+y2*Tyn)/W)/2.0;
              int kp1=(i+1)*my+j;
              int km1=(i-1)*my+j;
              if(j==0)          {b[k]+=-Tx0;}
   if(j== (mx-1))     {b[k]+=-Txn;}
   if(i==0)          {b[k]+=-Ty0;}
   else if(i==(my-1))  {b[k]+=-Tyn;}
   A[k][k]+=-4.0;
   if((i==0)&&(j==0)) {A[kp1][k]+=1.0;A[k][k+1]+=1.0;}
   else if((i==0)&&(j==(mx-1))) {A[kp1][k]+=1.0;A[k][k-1]+=1.0;}
   else if((i==(my-1))&&(j==0)) {A[km1][k]+=1.0;A[k][k+1]+=1.0;}
   else if((i==(my-1))&&(j==(mx-1))) {A[km1][k]+=1.0;A[k][k-1]+=1.0;}
   else if(j==0) {A[kp1][k]+=1.0;A[km1][k]+=1.0;A[k][k+1]+=1.0;}
   else if(i==0) {A[k][k+1]+=1.0;A[k][k-1]+=1.0;A[kp1][k]+=1.0;}
   else if(j==(mx-1)) {A[kp1][k]+=1.0;A[km1][k]+=1.0;A[k][k-1]+=1.0;}
   else if(i==(my-1)) {A[km1][k]+=1.0;A[k][k-1]+=1.0;A[k][k+1]+=1.0;}
   else {A[km1][k]+=1.0;A[kp1][k]+=1.0;A[k][k-1]+=1.0;A[k][k+1]+=1.0;}
   }
}
Text.print(A,"A");
Text.print(b,"B");
double T[][]=new double[ny+1][nx+1];
double a[]=conjugate_gradient1.conjugate_gradient(A,b);
for(i=0;i<=ny;i++)
{ for(j=0;j<=nx;j++)
{if(j==0) {T[i][j]=Tx0;}
  else if(i==0) {T[i][j]=Ty0;}
  else if(j==nx) {T[i][j]=Txn;}
  else if(i==ny) {T[i][j]=Tyn;}
  else
  {int k=(i-1)*my+(j-1);
   T[i][j]=a[k];
  }
  }

}
```

```
  return T;
}
 public static void main(String args[]) throws IOException,RemoteException, VisADException
 {  double a[][]=HT_planecoz(1.0,1.0,0.05,20.0,20.0,20.0,100.0);
   Text.print(a);
            ContourPlot pp=new ContourPlot(a,0.0,1.0,0.0,1.0,40, false,"x","y","T=f(x,y)");
            pp.plot();
            Plot3D.data(a,0.0,1.0,0.0,1.0,400,"x","y","T=f(x,y)");
 }
}
```



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0
20.0	20.21952...	20.43337...	20.63606...	20.82250...	20.98813...	21.12901...	21.24192...	21.32434...	21.37450...	21.39134...	21.37450...	21.32434...	21.24192...	21.12901...	20.98813...	20.82250...	20.63606...	20.43337...	20.21952...	20.0
20.0	20.44472...	20.87790...	21.28838...	21.66583...	22.00101...	22.28601...	22.51433...	22.68095...	22.78233...	22.81636...	22.78233...	22.68095...	22.51433...	22.28601...	22.00101...	21.66583...	21.28838...	20.87790...	20.44472...	20.0
20.0	20.68148...	21.34511...	21.97372...	22.55141...	23.06408...	23.49969...	23.84844...	24.10281...	24.25752...	24.30943...	24.25752...	24.10281...	23.84844...	23.49969...	23.06408...	22.55141...	21.97372...	21.34511...	20.68148...	20.0
20.0	20.93610...	21.84736...	22.70997...	23.50203...	24.20421...	24.80022...	25.27692...	25.62433...	25.83549...	25.90633...	25.83549...	25.62433...	25.27692...	24.80022...	24.20421...	23.50203...	22.70997...	21.84736...	20.93610...	20.0
20.0	21.21555...	22.39826...	23.51678...	24.54251...	25.45052...	26.22007...	26.83469...	27.28209...	27.55380...	27.64491...	27.55380...	27.28209...	26.83469...	26.22007...	25.45052...	24.54251...	23.51678...	22.39826...	21.21555...	20.0
20.0	21.52784...	23.01336...	24.41638...	25.70071...	26.83530...	27.79484...	28.55969...	29.11553...	29.45271...	29.56570...	29.45271...	29.11553...	28.55969...	27.79484...	26.83530...	25.70071...	24.41638...	23.01336...	21.52784...	20.0
20.0	21.88245...	23.71094...	25.43468...	27.00864...	28.39513...	29.56428...	30.49372...	31.16765...	31.57580...	31.71246...	31.57580...	31.16765...	30.49372...	29.56428...	28.39513...	27.00864...	25.43468...	23.71094...	21.88245...	20.0
20.0	22.29104...	24.51328...	26.60276...	28.50405...	30.17230...	31.57344...	32.68324...	33.48553...	33.97039...	34.13255...	33.97039...	33.48553...	32.68324...	31.57344...	30.17230...	28.50405...	26.60276...	24.51328...	22.29104...	20.0
20.0	22.76841...	25.44838...	27.95902...	30.23252...	32.21656...	33.87395...	35.18027...	36.12085...	36.68767...	36.87697...	36.68767...	36.12085...	35.18027...	33.87395...	32.21656...	30.23252...	27.95902...	25.44838...	22.76841...	20.0
20.0	23.33424...	26.55281...	29.55242...	32.25043...	34.58749...	36.52552...	38.04305...	39.12992...	39.78248...	40.0	39.78248...	39.12992...	38.04305...	36.52552...	34.58749...	32.25043...	29.55242...	26.55281...	23.33424...	20.0
20.0	24.01575...	27.87621...	31.44742...	34.62928...	37.35746...	39.59759...	41.33649...	42.57330...	43.31232...	43.55806...	43.31232...	42.57330...	41.33649...	39.59759...	37.35746...	34.62928...	31.44742...	27.87621...	24.01575...	20.0
20.0	24.85255...	29.48887...	33.73175...	37.46184...	40.61546...	43.17089...	45.13203...	46.51446...	47.33545...	47.60759...	47.33545...	46.51446...	45.13203...	43.17089...	40.61546...	37.46184...	33.73175...	29.48887...	24.85255...	20.0
20.0	25.90558...	31.49498...	36.52888...	40.87085...	44.47166...	47.33849...	49.50627...	51.01706...	51.90741...	52.20141...	51.90741...	51.01706...	49.50627...	47.33849...	44.47166...	40.87085...	36.52888...	31.49498...	25.90558...	20.0
20.0	27.27479...	34.05658...	40.01794...	45.02101...	49.06182...	52.20515...	54.53751...	56.14011...	57.07573...	57.38324...	57.07573...	56.14011...	54.53751...	52.20515...	49.06182...	45.02101...	40.01794...	34.05658...	27.27479...	20.0
20.0	29.13701...	37.43860...	44.46531...	50.13344...	54.54947...	57.88279...	60.29850...	61.93013...	62.87216...	63.18009...	62.87216...	61.93013...	60.29850...	57.88279...	54.54947...	50.13344...	44.46531...	37.43860...	29.13701...	20.0
20.0	31.83464...	42.09549...	50.27126...	56.49796...	61.11982...	64.47803...	66.84357...	68.40976...	69.30268...	69.59279...	69.30268...	68.40976...	66.84357...	64.47803...	61.11982...	56.49796...	50.27126...	42.09549...	31.83464...	20.0
20.0	36.10607...	48.83748...	58.02627...	64.46733...	68.95381...	72.06596...	74.18798...	75.56266...	76.33603...	76.58571...	76.33603...	75.56266...	74.18798...	72.06596...	68.95381...	64.46733...	58.02627...	48.83748...	36.10607...	20.0
20.0	43.75216...	59.12209...	68.52901...	74.39130...	78.16211...	80.64404...	82.27973...	83.31687...	83.89305...	84.07799...	83.89305...	83.31687...	82.27973...	80.64404...	78.16211...	74.39130...	68.52901...	59.12209...	43.75216...	20.0
20.0	59.78047...	75.36973...	82.57637...	86.40674...	88.65930...	90.06834...	90.97003...	91.53205...	91.84131...	91.94015...	91.84131...	91.53205...	90.97003...	90.06834...	88.65930...	86.40674...	82.57637...	75.36973...	59.78047...	20.0
20.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	20.0



Sonlu fark denklemleri daha yüksek dereceler için de yapılabilir. Fakat daha üst derece fark denklemleri kullanıldığında sınır değerlerinin hesaplanması çok daha zor olabilir (birden fazla sınır tabaka seviyesi oluşur). Bu durumun üstesinden gelmek için ara sınır tabakalarda daha düşük seviyedeki fark denklemlerinden yararlanılabilir. İkinci derece merkezi fark denklemleri (n=4) kullanırsak:

$$\frac{\partial^2 T(x,y)}{\partial x^2} \cong \frac{-T_{m-2,n}+16T_{m-1,n}-30T_{m,n}+16T_{m+1,n}-T_{m+2,n}}{12\Delta x^2}$$

$$\frac{\partial^2 T(x,y)}{\partial y^2} \cong \frac{-T_{m,n-2}+16T_{m,n-1}-30T_{m,n}+16T_{m,n+1}-T_{m,n+2}}{12\Delta y^2}$$

Bu durumda diferansiyel denklem aşağıdaki formu alır.

$$\frac{-T_{m-2,n}+16T_{m-1,n}-30T_{m,n}+16T_{m+1,n}-2T_{m+2,n}}{12\Delta x^2}+\frac{-T_{m,n-2}+16T_{m,n-1}-30T_{m,n}+16T_{m,n+1}-2T_{m,n+2}}{12\Delta y^2}=0$$

Bu durumda sınırı değerlerini alttaki gibi oluşturmamız gerekir.

x=0 (m=0) $\quad T_{0,n}=T_1$

$x=\Delta x$ (m=1) $\quad \frac{T_{2,n}-2T_{1,n}+T_1}{\Delta x^2}+\frac{T_{1,n+1}-2T_{1,n}+T_{1,n-1}}{\Delta y^2}=0$

x=L (m=M) $\quad T_{M,n}=T_2$

$x = L - \Delta x$ (m=M-1) $\frac{T_{M-2,n} - 2T_{M-1,n} + T_2}{\Delta x^2} + \frac{T_{M-1,n+1} - 2T_{M-1,n} + T_{M-1,n-1}}{\Delta y^2} = 0$

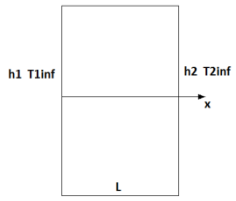
y=0 (n=0) $T_{m,0} = T_3$

$y = \Delta y$ (n=1) $\frac{T_{m+1,1} - 2T_{m,1} + T_{m-1,1}}{\Delta x^2} + \frac{T_{m,2} - 2T_{m,1} + T_3}{\Delta y^2} = 0$

y=W (n=N) $T_{m,N} = T_4$

$y = W - \Delta y$ (n=N-1) $\frac{T_{m+1,N} - 2T_{m,N} + T_{m-1,N}}{\Delta x^2} + \frac{T_{m,N-2} - 2T_{m,N-1} + T_4}{\Delta y^2} = 0$

İkinci bir örnek problem tanımlayalım. Bu problemde ısı iletimi x ve zaman boyutunda verilmiştir.



$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{1}{\alpha}\frac{\partial T(x,t)}{\partial t}$$

Sınır şartları

$$-k\frac{\partial T(x,t)}{\partial x} = h_1(T_{1\infty} - T(x,t)) \quad \text{x=0} \quad \text{t >0}$$

$$-k\frac{\partial T(x,t)}{\partial x} = h_2(T(x,t) - T_{2\infty}) \quad \text{x=L} \quad \text{t >0}$$

$T(x,t) = F(x)$ t=0 0<=x<=L

Buradaki Tk , h_1 , h_2 , $T_{1\infty}$ ve $T_{2\infty}$ sabit değerlerdir.

$$\phi\frac{T_{m+1}^p - 2T_m^p + T_{m-1}^p}{\Delta x^2} + (1-\phi)\frac{T_{m+1}^p - 2T_m^p + T_{m-1}^p}{\Delta x^2} = \frac{1}{\alpha}\frac{T_m^{p+1} - T_m^p}{\Delta t}$$

$t = p\Delta t \quad x = m\Delta x \; Fo = \frac{\alpha\Delta t}{\Delta x^2} \quad Bi = \frac{h\Delta x}{k} \quad \alpha = \frac{k}{\rho C_p}$

Denklemi düzenlersek

$\phi FoT_{m+1}^p + (1 - 2\phi Fo)T_m^p + \phi FoT_{m-1}^p = -(1-\phi)FoT_{m+1}^{p+1} + (1 + 2(1-\phi)Fo)T_m^{p+1} - (1-\phi)FoT_{m-1}^{p+1}$

$\phi = 1$ açık yöntem (bu yöntemde lineer denklem sistemi çözmemiz gerekmez)

$\phi = 0$ kapalı yöntem

Diğer değerler ϕ karışık çözüm

Sınır şartları:

$$\phi h_1 A(T_{1\infty} - T_0^p) + \phi\frac{kA}{\Delta x}(T_1^p - T_0^p) + (1-\phi)h_1 A(T_{1\infty} - T_0^{p+1}) + \phi\frac{kA}{\Delta x}(T_1^{p+1} - T_0^{p+1}) =$$

$$\rho C_p (A\frac{\Delta x}{2})\frac{T_0^{p+1} - T_0^p}{\Delta t}$$

$$\phi h_2 A(T_{2\infty} - T_{k-1}^{p}) + \phi \frac{kA}{\Delta x}(T_{k-2}^{p} - T_{k-1}^{p}) + (1-\phi)h_2 A(T_{2\infty} - T_{k-1}^{p}) + (1-\phi)\frac{kA}{\Delta x}(T_{k-2}^{p} - T_{k-1}^{p}) =$$

$$\rho C_p (A\frac{\Delta x}{2})\frac{T_{k-1}^{p+1} - T_{k-1}^{p}}{\Delta t}$$

Denklemi değiştirirsek

$$2\phi Bi_1 Fo(T_{1\infty} - T_0^{p}) + 2\phi(T_1^{p} - T_0^{p}) + 2(1-\phi)Bi_1 Fo(T_{1\infty} - T_0^{p+1}) + 2(1-\phi)(T_1^{p+1} - T_0^{p+1}) = T_0^{p+1} - T_0^{p}$$

$$2\phi Bi_2 Fo(T_{2\infty} - T_{k-1}^{p}) + 2\phi(T_{k-2}^{p} - T_{k-1}^{p}) + 2(1-\phi)Bi_2 Fo(T_{2\infty} - T_{k-1}^{p}) + 2(1-\phi)(T_{k-2}^{p} - T_{k-1}^{p}) = T_{k-1}^{p+1} - T_{k-1}^{p}$$

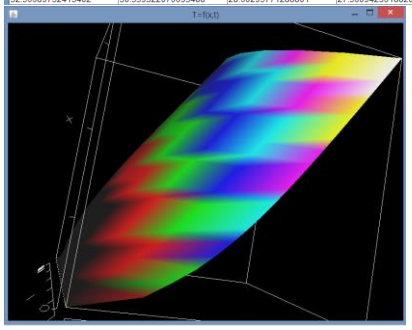
Program 10.2 sonlu farklar denklemleriyle çözüm

```
import java.io.*;

class HT_xt_FDB
{ public static double [] thomas(double a[][],double b[])
  {
  //3 band matrix
  int n=a.length;
  double f[]=new double[n];
  double e[]=new double[n];
  double g[]=new double[n];
  double x[]=new double[n];
  for(int i=0;i<n;i++)     {f[i]=a[i][i];}
  for(int i=0;i<(n-1);i++) {g[i]=a[i][i+1];}
  for(int i=0;i<(n-1);i++) {e[i+1]=a[i+1][i];}
  for(int k=1;k<n;k++)
     {e[k]=e[k]/f[k-1];
      f[k]=f[k]-e[k]*g[k-1];
     }
  for(int k=1;k<n;k++)
     {b[k]=b[k]-e[k]*b[k-1];
     }
  x[n-1]=b[n-1]/f[n-1];
  for(int k=(n-2);k>=0;k--)
      {x[k]=(b[k]-g[k]*x[k+1])/f[k];}
  return x;
  }
  public static double [][] q(double k, double dx,double T[][])
  {  //heat transfer
                int nx=T[0].length;
   int nt=T.length;
   double qi[][]=new double[nt][nx-1];
     for(int p=0;p<nt;p++)
     {  for(int m=1;m<nx;m++)
        { qi[p][m-1]=-k*(T[p][m]-T[p][m-1])/dx;}
     }
     return qi;
  }

  public static double [][] HT_xtcoz(double fi, double L,double t, double dt,double T0[],double alpha,double Bi1,double Bi2,double
T1inf,double T2inf)
  {
  //  L duvar kalınlığı
  //  t maksimum zaman
  //  dt zaman stebi
  //  T1 x=0 t>=0 boundary temperature
  //  T2 x=L t>=0 boundary temperature
  int nx=T0.length; //x step sayısı
  //System.out.println(nx="+nx)
  double A[][]=new double[nx][nx];
  double B[]=new double[nx];
  double dx=L/(nx-1); //x steb uzunluğu
  int nt=(int)(t/dt+0.001);
  double Fo=alpha*dt/(dx*dx);
  double T[][]=new double[nt+1][nx];
```

```
  for(int m=0;m<nx;m++) {T[0][m]=T0[m];}
  for(int p=0;p<nt;p++)
  { //t>0 wall temperatures
              for(int m=0;m<nx;m++)
    {
                 if(m==0) B[m]=(1.0-2.0*Fo*fi-2*Fo*Bi1*fi)*T[p][m]+2.0*fi*Fo*T[p][m+1]+2.0*Bi1*Fo*T1inf;
                 else if(m==(nx-1)) B[m]=(1.0-2.0*Fo*fi-2*Fo*Bi2*fi)*T[p][m]+2.0*fi*Fo*T[p][m-1]+2.0*Bi2*Fo*T2inf;
                 else  B[m]=fi*Fo*T[p][m-1]+(1.0-2.0*Fo*fi)*T[p][m]+fi*Fo*T[p][m+1];
                 if(m==0)
                 {A[m][m]=(1+2.0*Fo*Bi1*(1.0-fi)+2.0*Fo*(1.0-fi));
                  A[m][m+1]=-2.0*(1.0-fi)*Fo;
                 }
                 else if(m==(nx-1))
                 {A[m][m]=(1+2.0*Fo*Bi2*(1.0-fi)+2.0*Fo*(1.0-fi));
                  A[m][m-1]=-2.0*(1.0-fi)*Fo;
                 }
                 else
                 {A[m][m]=(2.0*Fo*(1.0-fi)+1);
                  A[m][m-1]=-(1.0-fi)*Fo;
                  A[m][m+1]=-(1.0-fi)*Fo;
                 }
    }
    T[p+1]=thomas(A,B);
  }
  return T;
  }

  public static void main(String args[])
  {
                 double T0[]=new double[81];
                 double L=0.8;
                 // t=0 daki T0(x) sıcaklık profili
                 int N=T0.length;
                 for(int m=0;m<N;m++) {T0[m]=300-300.0*Math.sin(m*Math.PI/N);}
                 double T1inf=100.0;  //x=0 da t>0 zamanı için duvar sıcaklığı sınır şartı
                 double T2inf=25.0;  //x=L de t>0 zamanı için duvar sıcaklığı sınır şartı
                 double h1=50.0;  //x=0 da t>0 zamanı için duvar sıcaklığı sınır şartı
                 double h2=50.0;  //x=L de t>0 zamanı için duvar sıcaklığı sınır şartı
                 double t=600.0; //sn toplam zaman
                 double dt=1.0; //sn gerçek zaman adımı
                 double dtplot=0.3; //sn her bir plotta bekleme zamanı
                 double alpha=8.4e-5;
                 double x0=0.0;
                  double xn=L;
                 double fi=0.5;
                 double k=200.0;
                 double dx=L/T0.length;
                 double h=250.0;
                 double Bi1=h*dx/k;
                 double Bi2=h*dx/k;
                 double T[][]=HT_xtcoz(fi,L,t,dt,T0,alpha,Bi1,Bi2,T1inf,T2inf);
                 double b[][]=Matrix.T(T);
                 double q[][]=q(k,dx,T);
                 timeplot p=new timeplot(T,x0,xn,dtplot,dt);
                 //timeplot p=new timeplot(q,x0,xn,dtplot,dt);
     p.start();
  }
}
```

data output(veri çıktısı) :

0	1	2	3	4	5	6	7	8	9	10
20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0
22.66629608174702	20.644218547436612	20.155661171071415	20.03772563059055	20.009613020087173	20.004388552648493	20.00961302008717	20.037725630590547	20.15566117107142	20.644218547436616	22.66629608174706
23.640457910611033	21.666587100815804	20.5930734480267	20.19039576162774	20.061395457244608	20.032798526838288	20.06139545724461	20.19039576162774	20.593073448026697	21.666587100815804	23.640457910611037
24.51723160771619	22.400308918331884	21.129985780313074	20.46718937182406	20.189747986548543	20.117504487543684	20.189747986548547	20.46718937182406	21.129985780313078	22.400308918331888	24.517231607716194
25.22293739933925	23.063397042575776	21.636038382646554	20.806404141600587	20.39747442442078	20.278297577840952	20.397474424428083	20.806404141600059	21.636038382646557	23.063397042575783	25.222937399339255
25.849081044645782	23.652576167100847	22.126697032067057	21.171692357722648	20.665855674081133	20.509633095218636	20.665855674081133	21.171692357722655	22.126697032067064	23.652576167100085	25.849081044645779
26.41418523731747	24.19551577279369 8	22.598933391263224	21.555851111936605	20.978264042952265	20.794892313882084	20.978264042952267	21.555851111936612	22.598933391263248	24.19551577279370	26.41418523733175
26.937570219085206	24.70376880940375	23.058006072988686	21.951773225337412	21.32271634264344	21.119568638109527	21.322716342643442	21.95177322533742	23.058006072988686	24.70376880940375	26.937570219085206
27.430542198520293	25.187216289763917	23.506917137170124	22.355923962911 21	21.690018484584993	21.472732737483373	21.690018484584996	22.355923962911213	23.506917137170127	25.187216289763917	27.430542198520293
27.901528557286973	25.652507861691966	23.948297485953557	22.765686724221755	22.073613888725202	21.84624371281408	22.073613888725205	22.765686724221755	23.948297485953557	25.652507861691973	27.901528557286976
28.356449760151207	26.104520342633396	24.384000797976686	23.179270477463 68	22.468725452720022	22.234219587426217	22.468725452720022	23.179270477463685	24.384000797976 69	26.104520342633 4	28.356449760151214
28.799586171613925	26.546741162608 49	24.815392994968374	23.595379881736203	22.871918149574 02	22.632399869519126	22.871918149574025	23.595379881736203	24.815392994968 38	26.546741162608495	28.799586171613928
29.234013647979909	26.981688767490372	25.24345658106938	24.013088354766673	23.280712730799852	23.03771408621582	23.280712730799852	24.013088354766673	25.24345658106939	26.981688767490375	29.234013647979094
29.661949929763 21	27.411176896322075	25.668901951745845	24.431728215328626	23.693323449342056	23.447948176692247	23.693323449342056	24.431728215328622	25.668901951745845	27.411176896322208	29.661949929763207
30.084993122433925	27.836513357158 46	26.092241362152 34	24.85081913614552	24.10846346546257	23.861506911253613	24.10846346546257	24.85081913614552	26.092241362152333	27.836513357158456	30.084993122433932
30.50429538350403	28.258640967491036	26.513844520095788	25.270015346423754	24.525205832376006	24.277240932600535	24.525205832376002	25.270015346423744	26.513844520095788	28.258640967491036	30.50429538350403
30.920687416960 82	28.678239621077008	26.933978269875382	25.689068809806714	24.942882907882 72	24.694322332083374	24.942882907882716	25.689068809680671	26.933978269875375	28.678239621077005	30.920687416960 82
31.334768298766257	29.095799703958637	27.35283533799651	26.107798527523162	25.361013948896684	25.112154854797843	25.361013948896684	26.10779852752316	27.352835337996506	29.09579970395863	31.334768298766 25
31.746970170234032	29.511675034023316	27.770555026263946	26.526078149366818	25.77925290215533	25.530309201984423	25.779252902155328	26.526078149366807	27.770555026263935	29.511675034023302	31.746970170234025
32.157604867544165	29.926121004769826	28.187238134369156	26.943814771451986	26.19735078381384	25.94847639594109	26.197350783813835	26.943814771451972	28.187238134369142	29.926121004769808	32.157604867544144
32.56689752413402	30.339322076693488	28.602957712888 01	27.360942361882024	26.61512856519235	26.36643417201944	26.615128565192347	27.360942361882017	28.602957712887996	30.339322076693474	32.566897524134

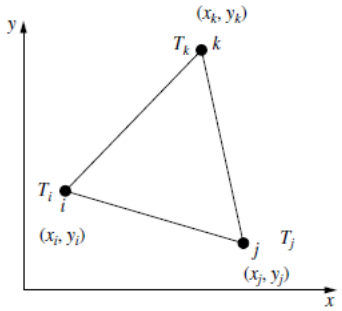


11.0Sonlu elemanlar metodu ile kısmi diferansiyel denklemlerin çözülmesi

11.1 sonlu elemanlar

Sonlu elemanlar yöntemi bir yüzey eğri uydurma yöntemi olarak görülebilir. Simpleks adını verdiğimiz elemanlar kullanarak aradeğer hesabı yapabiliriz. Türetilebilen ve integrali alınabilen bu elemanları kullanarak diferansiyel denklem çözümleri de gerçekleştirebiliriz. En basit iki boyutlu sonlu eleman simpleks şekillerinden biri üçgen elemandır. Bu elemanı şu şekilde tanımlanabilir.

$$T(x,y)=\alpha_1+\alpha_2 x+\alpha_3 y$$



Şekil 11.1 Üçgen düzlem sonlu eleman

Burada polinom lineerdir ve x ve y nin fonksiyonudur. Node noktalarının koordinatlarını denkleme koyarak α_1, α_2 ve α_3 katsayılarını bulabiliriz.

$T_i=\alpha_1+\alpha_2 x_i+\alpha_3 y_i$

$T_j=\alpha_1+\alpha_2 x_j+\alpha_3 y_j$

$T_k=\alpha_1+\alpha_2 x_k+\alpha_3 y_k$

Çözüm seti:

$$\alpha_1=\frac{1}{2A}\left[(x_j y_k-x_k y_j)T_i+(x_k y_i-x_i y_k)T_j+(x_i y_j-x_j y_i)T_k\right]$$

$$\alpha_2=\frac{1}{2A}\left[(y_j-y_k)T_i+(y_k-y_i)T_j+(y_i-y_j)T_k\right]$$

$$\alpha_3=\frac{1}{2A}\left[(x_k-x_j)T_i+(x_i-x_k)T_j+(x_j-x_i)T_k\right]$$

Burada A üçgenin alanıdır ve

$$2A=\det\begin{bmatrix}1 & x_i & y_i\\ 1 & x_j & y_j\\ 1 & x_k & y_k\end{bmatrix}=(x_i y_j-x_j y_i)+(x_k y_i-x_i y_k)+(x_j y_k-x_k y_j)$$

Formülü ile hesaplanabilir. α_1, α_2 ve α_3 değerlerini yerine koyarsak

$$T=N_i T_i+N_j T_j+N_k T_k$$

$$N_i=\frac{1}{2A}\left[a_i+b_i x+c_i y\right]$$

$$N_j=\frac{1}{2A}\left[a_j+b_j x+c_j y\right]$$

$$N_k=\frac{1}{2A}\left[a_k+b_k x+c_k y\right]$$

N değerlerine şekil fonksiyonu ismini veriyoruz. Buradaki katsayılar açık olarak yazılırsa

$$a_i=(x_j y_k-x_k y_j)\qquad a_j=(x_k y_i-x_i y_k)\qquad a_k=(x_i y_j-x_j y_i)$$

$$b_i = (y_j - y_k) \qquad b_j = (y_k - y_i) \qquad b_k = (y_i - y_j)$$

$$c_i = (x_k - x_j) \qquad c_j = (x_i - x_k) \qquad c_k = (x_j - x_i)$$

Eğer şekil fonksiyonları uç nokta koordinatlarında değerlendirilirse (x_i,y_i), noktasında $N_i(x_i,y_i)=1$ olur ve diğer noktaların şekil fonksiyonları $N_i(x_j,y_j)=0$, değerini alır. Genel bir tanım yapılırsa şekil fonksiyonları herhangi bir (x,y) noktası için $N_i+N_j+N_k=1$ olur. Lineer simpleks elemanı veya sonlu eleman bize göreceli olarak basit bir intepolasyon yöntemi vermiş oldu. Herhangi bir (x,y) noktasında sıcaklık için:

$$T = N_i T_i + N_j T_j + N_k T_k = [N_i \;\; N_j \;\; N_k] \begin{bmatrix} T_i \\ T_j \\ T_k \end{bmatrix} = [N]\{T\}$$

Denklemini yazabiliriz. Bu denklemin türevide şekil fonksiyonlarını türeterek basirt bir şekilde oluşturulabilir.

$$\{g\} = \begin{bmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} b_i & b_j & b_k \\ c_i & c_j & c_k \end{bmatrix} \begin{bmatrix} T_i \\ T_j \\ T_k \end{bmatrix} = [B]\{T\}$$: Buradan ısı transferine de geçebiliriz

$$\begin{bmatrix} q_x \\ q_y \end{bmatrix} = -k[B]\{T\} = -\frac{k}{2A} \begin{bmatrix} b_i & b_j & b_k \\ c_i & c_j & c_k \end{bmatrix} \begin{bmatrix} T_i \\ T_j \\ T_k \end{bmatrix}$$

Sonlu elemanın integrali de bir boyutlu simpleks için

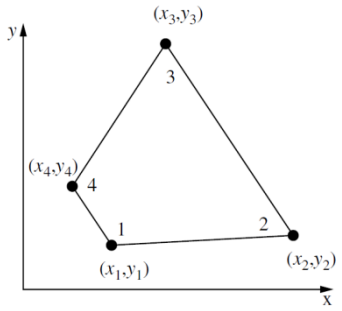
$$\int_l N_i^a N_j^b N_k^c dl = \frac{a!b!c!}{(a+b+c+1)!} 2l$$

İki boyutlu simpleks için

$$\int_A N_i^a N_j^b N_k^c dA = \frac{a!b!c!}{(a+b+c+2)!} 2A$$

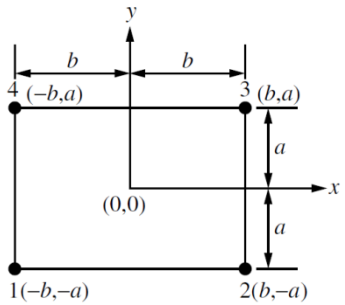
Şeklinde hesaplanabilir. Değişik geometriler kullanarak değişik simpleks elemanları kullanabiliriz. Örneğin dikdörtgen prizma bir simpleks elemanı tanımlamak istersek:

$$T(x,y) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy$$



Şekil 11.2 dikdörtgen prizma sonlu eleman

Bu şeklin özel bir alt şekli de dikdörtgen elemandır.



Şekil 11.3 dikdörtgen sonlu eleman

Bu eleman için şekil fonksiyonu:

$$T(x,y) = N_1T_1 + N_2T_{21} + N_3T_3 + N_4T_4$$

$$N_1 = \frac{1}{4ab}(b-x)(a-y)$$

$$N_2 = \frac{1}{4ab}(b+x)(a-y)$$

$$N_3 = \frac{1}{4ab}(b+x)(a+y)$$

$$N_2 = \frac{1}{4ab}(b-x)(a+y)$$

Türev fonksiyonu:

$$\{g\} = \begin{bmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{bmatrix} = \frac{1}{4ab}\begin{bmatrix} -(a-y) & (a-y) & (a+y) & -(a+y) \\ -(b-x) & -(b+x) & (b+x) & (b-x) \end{bmatrix}\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = [B]\{T\}$$

11.2 Sonlu eleman metodu kullanarak kısmi diferansiyel denklemin çözülmesi

Şimdi genel bir zamandan bağımsız kısmi diferansiyel denklem sistemini sonlu elemanlar yöntemi ile çözmeye çalışalım.

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) + g = 0$$

Sınır şartları

$T=T_b$ yüzey S_1'de

$$k_x \frac{\partial T}{\partial x}\vec{l} + k_y \frac{\partial T}{\partial y}\vec{m} + q = 0 \quad \text{yüzey } S_2\text{'de}$$

$$k_x \frac{\partial T}{\partial x}\vec{l} + k_y \frac{\partial T}{\partial y}\vec{m} + h(T - T_a) = 0 \quad \text{yüzey } S_3\text{'de}$$

Diferansiyel denkelemi çeşitli metodlarla integre edebiliriz ve böylece çözmüş oluruz. En çok kullanılan yöntemlerden biri Galarkin Metodudur. Bu metod alttaki denklemi baz alır.

$\int_\Omega w_k L(T) d\Omega = 0$ burada Ω hacimdir ve w_k node elemanı ve L(T) çözülecek diferansiyel denklemdir. Biz node elemanı olarak şekil fonksiyonunu kullanacağız.

$$\int_\Omega N_k \left[\frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) + g\right] d\Omega = 0$$

Sıcaklık değerini T=[N]{T} denklemi ile tanımladığımızı hatırlayalım . Aynı zamanda kısmi integrasyon kuralı

$\int u dv = uv - \int v du$ so $\int_\Omega \alpha_1 \frac{\partial \alpha_2}{\partial x_1} d\Omega = \int_s \alpha_1\alpha_2 dS - \int_\Omega \frac{\partial \alpha_1}{\partial x_1}\alpha_2 d\Omega$ şeklinde verilebilir. Bunu şekil faktörüyle çarpılmış türev denklemine uygularsak:

$\int_\Omega N_k\left[\frac{\partial}{\partial x}\left(k_x\frac{\partial T}{\partial x}\right)\right]d\Omega=\int_s N_k\left(k_x\frac{\partial T}{\partial x}\right)dS-\int_\Omega\frac{\partial N_k}{\partial x}k_x\frac{\partial N_m}{\partial x}Td\Omega$ Şimdi tüm denkleme uygulanmış formda yazalım

$$-\int_\Omega\left[k_x\frac{\partial N_k}{\partial x}\frac{\partial N_m}{\partial x}+k_y\frac{\partial N_k}{\partial y}\frac{\partial N_m}{\partial y}\right]\{T_m\}d\Omega+\int_\Omega gN_k d\Omega+\int_s N_k\left(k_x\frac{\partial T}{\partial x}+k_y\frac{\partial T}{\partial y}\right)dS=0$$

Sınır şartlarından:

$$\int_s N_k\left(k_x\frac{\partial T}{\partial x}+k_y\frac{\partial T}{\partial y}\right)dS=-\int_s N_k\left(q+h(T-T_a\right)dS=-\int_s N_k qdS-\int_s N_k hN_m\{T_m\}dS+\int_s N_k hT_a dS$$

$$-\int_\Omega\left[k_x\frac{\partial N_k}{\partial x}\frac{\partial N_m}{\partial x}+k_y\frac{\partial N_k}{\partial y}\frac{\partial N_m}{\partial y}\right]\{T_m\}d\Omega+\int_\Omega gN_k d\Omega+-\int_s N_k qdS-\int_s N_k hN_m\{T_m\}dS+\int_s N_k hT_a dS=0$$

$[K]\{T\}=\{f\}$ Burada

$$[K]=-\int_\Omega\left[k_x\frac{\partial N_k}{\partial x}\frac{\partial N_m}{\partial x}+k_y\frac{\partial N_k}{\partial y}\frac{\partial N_m}{\partial y}\right]d\Omega+\int_S hN_k\left(N_m\right)dS$$

$\{f\}=\int_\Omega gN_k d\Omega-\int_s N_k qdS+\int_S hN_k T_a dS$ Denklem matris formunda yazılırsa:

$[K]=-\int_\Omega[B]^T[D][B]d\Omega+\int_S h[N]^T[N]dS$ burada

$$[D]=\begin{bmatrix}k_x & 0\\ 0 & k_y\end{bmatrix}$$

$$\{f\}=\int_\Omega g[N]^T d\Omega-\int_s[N]^T qdS+\int_S h[N]^T T_a dS$$

Şimdi matris terimlerinin yüzey üçgen sonlu elemanlar eşdeğerlerini denkleme yazalım:

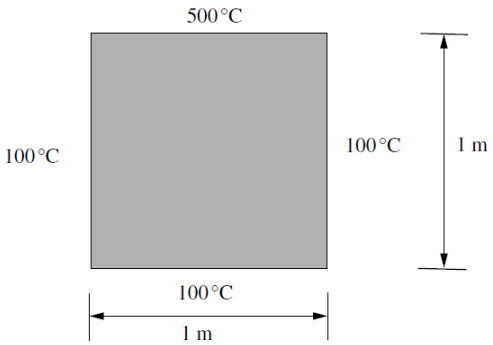
$$\int_l N_i^a N_j^b dl=\frac{a!b!}{(a+b+1)!}l\quad\int_l N_i^2 dl=\frac{2!0!}{(2+0+1)!}l=\frac{2l}{6}\quad\int_l N_i\,N_j dl=\frac{1!1!}{(1+1+1)!}l=\frac{l}{6}$$

$$[K]=\frac{t}{4A}\left\{k_x\begin{bmatrix}b_i^2 & b_ib_j & b_ib_k\\ b_ib_j & b_j^2 & b_jb_k\\ b_ib_k & b_jb_k & b_k^2\end{bmatrix}+k_y\begin{bmatrix}c_i^2 & c_ic_j & c_ic_k\\ c_ic_j & c_j^2 & c_jc_k\\ c_ic_k & c_jc_k & c_k^2\end{bmatrix}\right\}+\frac{htl_{ij}}{6}\begin{bmatrix}2&1&0\\1&2&0\\0&0&0\end{bmatrix}+\frac{htl_{jk}}{6}\begin{bmatrix}0&1&0\\0&2&1\\0&1&0\end{bmatrix}++\frac{htl_{ki}}{6}\begin{bmatrix}2&1&0\\0&0&0\\0&1&2\end{bmatrix}$$

$$\{f\}=\frac{gAt}{3}\begin{Bmatrix}1\\1\\1\end{Bmatrix}-\frac{qtl_{ij}}{2}\begin{Bmatrix}1\\1\\0\end{Bmatrix}-\frac{qtl_{jk}}{2}\begin{Bmatrix}0\\1\\1\end{Bmatrix}-\frac{qtl_{ik}}{2}\begin{Bmatrix}1\\0\\1\end{Bmatrix}+\frac{hT_a tl_{ij}}{2}\begin{Bmatrix}1\\1\\0\end{Bmatrix}+\frac{hT_a tl_{jk}}{2}\begin{Bmatrix}0\\1\\1\end{Bmatrix}+\frac{hT_a tl_{ik}}{2}\begin{Bmatrix}1\\0\\1\end{Bmatrix}$$

PROBLEM: ZANADAN BAĞIMSIZ SICAKLIK SINIRLI İKİ BOYUTLU ISI İLETİMİ

1mx1m boyutlarında kare şeklinde bir levhanın çevre sıcaklıkları üç tarafında 100 C ve bir tarafında 500 C'dir. Sıcaklık profilini hesaplayınız.



Program 10.1 üçgen sonlu eleman

```
public class triangular_element
{
public double t; //Thickness of two dimensional element
public double a[],b[],c[];
public double x[],y[];//coordinates of data points
public double xc,yc; //center point
public double A; //Area
public double N[],B[][]; //Shape factor and derivative
public double Ti[];      //Temperature nodes
public int n_point[]; //point numbers
public int n_element; //element number
public double l[];
public boolean isconv[];
public boolean isheatflux[];
public boolean isheatgen;
public double h[];
public double Tinf[];
public double qflux[];
public double qgen;
public triangular_element(int n_elementi,int n_pointi[],double xi[],double yi[])
{x=new double[3];
y=new double[3];
Ti=new double[3];
l=new double[3];//0:lij,1:ljk,2:lik
isconv=new boolean[3];
isheatflux=new boolean[3];
n_point=new int[3];
xc=0;yc=0;
isheatgen=false;
h=new double[3];
Tinf=new double[3];
qflux=new double[3];
n_element=n_elementi;
for(int i=0;i<3;i++)
{n_point[i]=n_pointi[i];
x[i]=xi[i];y[i]=yi[i];
xc+=x[i];yc+=y[i];
isconv[i]=false;
isheatflux[i]=false;
h[i]=0.0;
Tinf[i]=0.0;
qflux[i]=0.0;
qgen=0.0;
}
xc/=3.0;
yc/=3.0;
a=new double[3];
b=new double[3];
c=new double[3];
N=new double[3];
B=new double[2][3];
```

```
a[0]=x[1]*y[2]-x[2]*y[1];
a[1]=x[2]*y[0]-x[0]*y[2];
a[2]=x[0]*y[1]-x[1]*y[0];
b[0]=y[1]-y[2];
b[1]=y[2]-y[0];
b[2]=y[0]-y[1];
c[0]=x[2]-x[1];
c[1]=x[0]-x[2];
c[2]=x[1]-x[0];
A=0.5*((x[0]*y[1]-x[1]*y[0])+(x[2]*y[0]-x[0]*y[2])+(x[1]*y[2]-x[2]*y[1]));
for(int i=0;i<3;i++)
{ B[0][i]=1.0/(2.0*A)*b[i];
B[1][i]=1.0/(2.0*A)*c[i];
}
l[0]=Math.sqrt(b[2]*b[2]+c[2]*c[2]);
l[1]=Math.sqrt(b[0]*b[0]+c[0]*c[0]);
l[2]=Math.sqrt(b[1]*b[1]+c[1]*c[1]);
}

public double[][] K(double ti,double k[])
{t=ti;
double a0=t/(4.0*A)*k[0];
double a1=t/(4.0*A)*k[1];
double b0=t*h[0]*l[0]/6.0;
double b1=t*h[1]*l[1]/6.0;
double b2=t*h[2]*l[2]/6.0;
double K1[][]={{(a0*b[0]*b[0]+a1*c[0]*c[0]+2*b0+2.0*b2),
        (a0*b[0]*b[1]+a1*c[0]*c[1]+b0),
        (a0*b[0]*b[2]+a1*c[0]*c[2]+b2)},
        {(a0*b[1]*b[0]+a1*c[1]*c[0]+b0),
        (a0*b[1]*b[1]+a1*c[1]*c[1]+2*b0+2*b1),
        (a0*b[1]*b[2]+a1*c[1]*c[2]+b1)},
        {(a0*b[2]*b[0]+a1*c[2]*c[0]+b2),
        (a0*b[2]*b[1]+a1*c[2]*c[1]+b1),
        (a0*b[2]*b[2]+a1*c[2]*c[2]+2*b1+2*b2)}
        };
return K1;
}
public double[] f(double ti,double k[])
{
t=ti;
double a0=qgen*A*t/3.0;
double b0=-qflux[0]*t*l[0]/2.0;
double b1=-qflux[1]*t*l[1]/2.0;
double b2=-qflux[2]*t*l[2]/2.0;
double c0= h[0]* Tinf[0]*t*l[0]/2.0;
double c1= h[1]* Tinf[1]*t*l[1]/2.0;
double c2= h[2]* Tinf[2]*t*l[2]/2.0;
double f1[]={(a0+b0+b2+c0+c2),(a0+b0+b1+c0+c1),(a0+b1+b2+c1+c2)};
return f1;
}


public int setI(int cond,double L0,double L,double W0,double W)
{
int i=100;
        if(cond==0) //x=0
        {if(x[0]==L0 && x[1]==L0) i=0;
        else if(x[1]==L0 && x[2]==L0) i=1;
        else if(x[0]==L0 && x[2]==L0) i=2;
        else i=-1;
        }
        else if(cond==1) //x=L
        {if(x[0]==L && x[1]==L) i=0;
        else if(x[1]==L && x[2]==L) i=1;
        else if(x[0]==L && x[2]==L) i=2;
```

```
                else i=-1;
                }
                else if(cond==2) //y=0
                {if(y[0]==W0 && y[1]==W0) i=0;
                else if(y[1]==W0 && y[2]==W0) i=1;
                else if(y[0]==W0 && y[2]==W0) i=2;
                else i=-1;
                }
                else if(cond==3) //y=W
                {if(y[0]==W && y[1]==W) i=0;
                else if(y[1]==W && y[2]==W) i=1;
                else if(y[0]==W && y[2]==W) i=2;
                else i=-1;
                }
                return i;
}

public void setConvection(int i, double hi, double tinfi)
{if(i>=0 && i<=2)
{
                h[i]=hi;Tinf[i]=tinfi;isconv[i]=true;
}
}

public void setHeatFlux(int i, double qi)
{if(i>=0 && i<=2 ){qflux[i]=qi;isheatflux[i]=true;}  }

public void setHeatGeneration(double qgeni)
{qgen=qgeni;isheatgen=true;}
public void setN(double x,double y)
{for(int i=0;i<3;i++)
{N[i]=1.0/(2.0*A)*(a[i]+b[i]*x+c[i]*y);}
}
public double[]  q(double k)
{ double qi[]=new double[2];
                for(int i=0;i<3;i++)
{qi[0]+=-k*B[0][i]*Ti[i];
qi[1]+=-k*B[1][i]*Ti[i];
}
return qi;
}
public double[]  q(double k,double Ti[])
{  //heat transfer
                setT(Ti);
double qi[]=new double[2];
for(int i=0;i<3;i++)
                {qi[0]+=-k*B[0][i]*Ti[i];
                qi[1]+=-k*B[1][i]*Ti[i];
                }
return qi;
}

public void setN(double z[])
{setN(z[0],z[1]);}
public void setT(double TT[])
{for(int i=0;i<3;i++)
{Ti[i]=TT[i];}
}
public double T(double x,double y)
{ //interpolation
                double TT=0;
setN(x,y);
for(int i=0;i<3;i++)
{TT+=N[i]*Ti[i];}
return TT;
}
```

```
public double T(double T[],double x,double y)
{ //set grid point temperatures and get interpolated value
        setT(T);
setN(x,y);
double TT=0;
for(int i=0;i<3;i++)
{TT+=N[i]*Ti[i];}
return TT;
}
public double T(double T[],double z[])
{ return T(T,z[0],z[1]);}
public static void main(String arg[])
{int n_elementi=0;
int n_pointi[]={0,1,2};
double xi[]={0.0,4,0};
double yi[]={0,0,2.5};
triangular_element fe=new triangular_element(n_elementi,n_pointi,xi,yi);
double p[]={2.0,1.0};
fe.setN(p);
double Ti[]={50,70,100};
fe.setT(Ti);
double T=fe.T(Ti,p[0],p[1]);
double q[]=fe.q(200.0);
}

}
```

Program 10.2 üçgen sonlu elemanın eleman numarasını saptamak için element_number.java

```
public class element_number
{int number;
int ijk;
public element_number(int numberi,int ijki)
{
number=numberi;
if(ijki<=2) ijk=ijki;
else {ijk=0;System.out.println("wrong value");}
}
public boolean equals(element_number en)
{if(en.number==number && en.ijk==ijk) return true;
else return false;
}
public int getNumber() {return number;}
public int getIjk() {return ijk;}
public int[] output()
{int x[]={number,ijk};
return x;
}
}
```

Program 10.3 üçgen sonlu elemanın eleman numarasını saptamak için element_number.java: setElement.java

```
import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class setElement
{   ArrayList<element_number> en;
        int no[];
        int ij[];
        public setElement()
        {en=new ArrayList<element_number>();}
        public void add(int n1,int ij1)
        {element_number en=new element_number(n1,ij1);
        add(en);
        }
```

```
            public void add(element_number s1)
            { //int n=en.size();
            boolean b=false;
            element_number s2;
            Iterator n2=en.iterator();
            while(n2.hasNext()&& !b)
            { s2=(element_number)n2.next();
            b=s2.equals(s1);
            }
            if(!b) en.add(s1);
            }

public int[][] output()
{
            int n=en.size();
            int a[][]=new int[n][2];
            Iterator i=en.iterator();
            int j=0;
            while(i.hasNext()) {element_number el=(element_number)i.next();a[j]=el.output();j++;}
            return a;
}
public static void main(String arg[])
{setElement se=new setElement();
element_number en1=new element_number(11,0);
element_number en2=new element_number(11,1);
element_number en3=new element_number(13,2);
element_number en4=new element_number(13,0);
se.add(en1);
se.add(en2);
se.add(en3);
se.add(en4);
int a[][]=se.output();
Text.print(a);
}
}
```

Program 10.4 : Sonlu eleman gridi oluşturan triangular_grid.java

```
import javax.swing.*;
import org.math.plot.*;
import static java.lang.Math.*;
import static org.math.array.DoubleArray.*;
import visad.*;
import visad.java3d.DisplayImplJ3D;
import visad.java2d.DisplayImplJ2D;
import java.rmi.RemoteException;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class triangular_grid
{
public triangular_element te[];
public setElement se[];
public int N,M,K,K1,K2,Nm1,Mm1;
public double x[][];
public double y[][];
public double T[][];
public double Ti[];
public double xi[];
public double yi[];
public boolean bi[];
public double dx,dy;
public int Bx0[],BxM[],By0[],ByN[];
public double L,W;
public double min(double xi[][])
{ double min=1e200;
```

```
for(int i=0;i<xi.length;i++)
{for(int j=0;j<xi[0].length;j++)
        {if(xi[i][j]<min) min=xi[i][j];}
}
return min;
}
public double max(double xi[][])
{double max=-1e200;
for(int i=0;i<xi.length;i++)
{for(int j=0;j<xi[0].length;j++)
{if(xi[i][j]<max) max=xi[i][j];}
}
return max;
}
public triangular_grid(double xi[][],double yi[][])
{x=xi;
y=yi;
N=x.length-1;
M=x[0].length-1;
T=new double[N+1][M+1];
L=max(xi);
W=max(yi);
setElements();
}

public void generateTriangularGrid(double Li, double Wi,int Ni, int Mi)
{int i,j;
N=Ni;M=Mi;
L=Li;
W=Wi;
x=new double[N+1][M+1];
y=new double[N+1][M+1];
T=new double[N+1][M+1];
setElements();
}

public void setElements()
{ int i,j;
dx=L/M;
dy=W/N;
K=M*N;
K1=K*2;
K2=K1+1;
//grid points
xi=new double[K2];
yi=new double[K2];
Ti=new double[K2];
bi=new boolean[K2];
se=new setElement[K2];
for(i=0;i<K2;i++) {se[i]=new setElement();}
te=new triangular_element[K1];
int k1=-1,k=0,n=0,m=0;
for(n=0;n<=N;n++)
{ for(m=0;m<=M;m++)
{k=(N+1)*n+m;
x[n][m]=m*dx;y[n][m]=n*dy;
xi[k]=x[n][m];
yi[k]=y[n][m];
bi[k]=false;
Ti[k]=0.0;
}
}

for(n=0;n<N;n++)
{for(m=0;m<M;m++)
{ k=(N+1)*n+m;
```

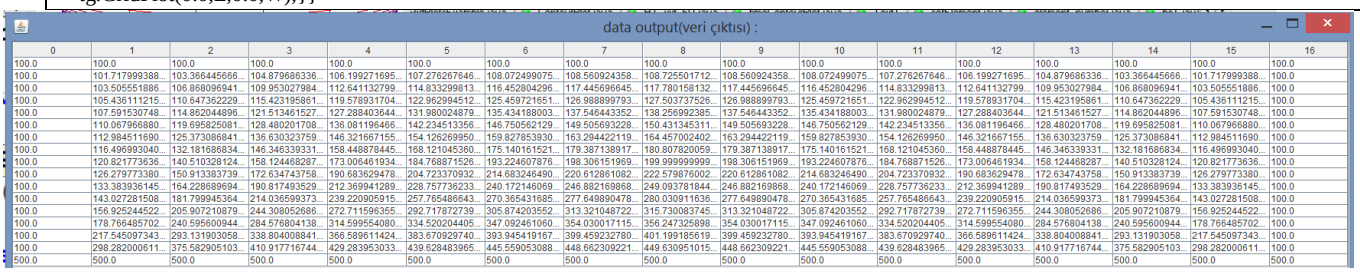
```
                int kp1=(N+1)*(n+1)+m;
                int kp1p1= kp1+1;
                double xi1[]={x[n][m],x[n][m+1],x[n+1][m]};
                double yi1[]={y[n][m],y[n][m+1],y[n+1][m]};

                k1++;
                int n_point1[]={k,(k+1),kp1};
                se[k].add(k1,0);
                se[k+1].add(k1,1);
                se[kp1].add(k1,2);
                te[k1]=new triangular_element(k1,n_point1,xi1,yi1);
                double xi2[]={x[n][m+1],x[n+1][m+1],x[n+1][m]};
                double yi2[]={y[n][m+1],y[n+1][m+1],y[n+1][m]};
                k1++;
                int n_point2[]={k+1,kp1p1,kp1};
                se[k+1].add(k1,0);
                se[kp1p1].add(k1,1);
                se[kp1].add(k1,2);
                te[k1]=new triangular_element(k1,n_point2,xi2,yi2);
    }
    }
    }
    public triangular_grid(double Li, double Wi,int Ni, int Mi)
    {generateTriangularGrid(Li,Wi,Ni,Mi);}
    public int find_element(double z[])
    { return find_element(z[0],z[1]);}
    public int find_element(double x,double y)
    {double R1=Math.sqrt(x*x+y*y);
    int kmin=0;
    double min=1.0e50;
    for(int k=0;k<K1;k++)
    {   te[k].setN(x,y);
                double r0=Math.sqrt((te[k].x[0]-x)*(te[k].x[0]-x)+(te[k].y[0]-y)*(te[k].y[0]-y));
                double r1=Math.sqrt((te[k].x[1]-x)*(te[k].x[1]-x)+(te[k].y[1]-y)*(te[k].y[1]-y));
                double r2=Math.sqrt((te[k].x[2]-x)*(te[k].x[2]-x)+(te[k].y[2]-y)*(te[k].y[2]-y));
                double dr=Math.sqrt(r0*r0+r1*r1+r2*r2);
                boolean b1=(x>=te[k].x[0]) && (x<=te[k].x[2]);
                boolean b2=(y>=te[k].y[0]) && (y<=te[k].y[1]);
                boolean b= b1 && b2;
                if(dr<min) {min=dr;kmin=k;}
    }
    return kmin;
    }
    public void setConvection(int cond,double L0,double L,double W0,double W, double hi, double tinfi)
    {//cond 0: x==L0
    //cond 1: x==L
    //cond 2: y==W0
    //cond 3: y==W
    int j=0;
    for(int i=0;i<K1;i++)
    {  j=te[i].setI(cond,L0,L,W0,W);
                te[i].setConvection(j,hi,tinfi);
    }
    }
    public void setHeatFlux(int cond,double L0,double L,double W0,double W, double qi)
    {//cond 0: x==L0
    //cond 1: x==L
    //cond 2: y==W0
    //cond 3: y==W
    int j=-1;
    for(int i=0;i<K1;i++)
    {  j=te[i].setI(cond,L0,L,W0,W);
                te[i].setHeatFlux(j,qi);
    }
    }
    public void setTemperature(int cond,double L0,double L,double W0,double W, double TTi)
```

```
{//cond 0: x==L0
//cond 1: x==L
//cond 2: y==W0
//cond 3: y==W
for(int i=0;i<K2;i++)
{  if(cond==0)
                {if(xi[i]==L0) {Ti[i]=TTi;bi[i]=true;}}
                else if(cond==1)
                {if(xi[i]==L) {Ti[i]=TTi;bi[i]=true;}}
                else if(cond==2)
                {if(yi[i]==W0) {Ti[i]=TTi;bi[i]=true;}}
                else if(cond==3)
                {if(yi[i]==W) {Ti[i]=TTi;bi[i]=true;}}
}
}
public void setHeatGeneration(int node[],double qgeni)
{  int n=node.length;
for(int i=0;i<n;i++)
{te[node[i]].setHeatGeneration(qgeni);}
}

public void solve(double ti,double k[])
{
double K[][]=new double[K2][K2];
double KK[][]=new double[K2][K2];
double ff[]=new double[K2];
double KN[][]=new double[3][3];
double f[]=new double[K2];
double fN[]=new double[3];
for(int i=0;i<K1;i++)
{KN=te[i].K(ti,k);
fN=te[i].f(ti,k);
int n1=te[i].n_element;
int m[]=te[i].n_point;
for(int j=0;j<3;j++)
{ for(int k1=0;k1<3;k1++)
                {K[m[j]][m[k1]]+=KN[j][k1];KK[m[j]][m[k1]]=K[m[j]][m[k1]];}
                f[m[j]]+=fN[j];
                ff[m[j]]=f[m[j]];
}
}
//redesign K matrix and f vector if any known
//temperature boundary is given
//int j=0;
for(int j=0;j<K2;j++)
{
                for(int i=0;i<K2;i++)
                { if(bi[i])
                {
                if(i==j)
                {f[i]=Ti[i];K[i][j]=1.0;}
                else
                {
                K[i][j]=0.0;}
                //System.out.println("i="+i+"j="+j+"bi="+bi[i]+"Ti="+Ti[i]+"fi="+f[i]);
                }
                }//end of for
} //end of for
//System.out.println("K =\n"+Matrix.toString(K));
//System.out.println("f =\n"+Matrix.toStringT(f));
double TT[]=AXB(K,f);
int k1=0;
for(int n=0;n<=N;n++)
{ for(int m=0;m<=M;m++)
{k1=(N+1)*n+m;
T[n][m]=TT[k1];
```

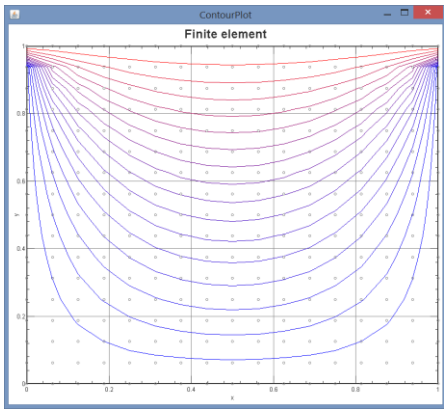
```
}
}
}

//LU decomposition method
public static double[][] LU(double c[][],int indx[],int d[])
{
//returns LU decomposition of matrix c and index indx
double a[][];
int n=c.length;
a=new double[n][n];
double vv[];
vv=new double[n];
double sum,dum,big,temp;
int i,j,k;
int imax;
int nmax=100;
double tiny=1.0e-40;
imax=0;
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
        a[i-1][j-1]=c[i-1][j-1];
}
d[0]=1;
for(i=1;i<=n;i++)
{
big=0.0;
for(j=1;j<=n;j++)
        {
        if(Math.abs(a[i-1][j-1])>big) big=Math.abs(a[i-1][j-1]);
        }
if(big==0) {System.out.println("singular matrix");return a;}
vv[i-1]=1.0/big;
}
for(j=1;j<=n;j++)
{
for(i=1;i<j;i++)
        {
        sum=a[i-1][j-1];
        for(k=1;k<i;k++)
        {
        sum-=a[i-1][k-1]*a[k-1][j-1];
        }
        a[i-1][j-1]=sum;
        }
big=0;
for(i=j;i<=n;i++)
        {
        sum=a[i-1][j-1];
        for(k=1;k<j;k++)
        {
        sum-=a[i-1][k-1]*a[k-1][j-1];
        }
        a[i-1][j-1]=sum;
        dum=vv[i-1]*Math.abs(sum);
        if(dum>=big)
        {
        imax=i;
        big=dum;
        }
        } //end of i=0
if(j != imax)
        {
        for(k=1;k<=n;k++)
        {
```

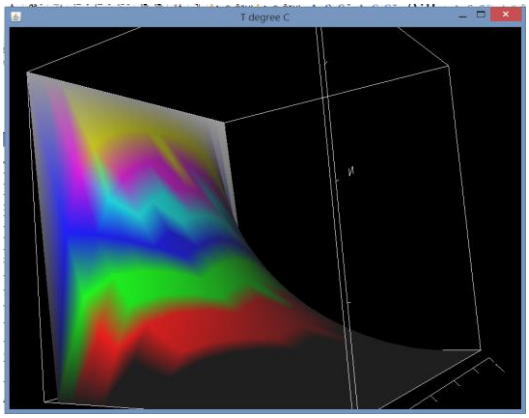
```
                dum=a[imax-1][k-1];
                a[imax-1][k-1]=a[j-1][k-1];
                a[j-1][k-1]=dum;
                }
                d[0]=-d[0];
                vv[imax-1]=vv[j-1];
                } //end of if
    indx[j-1]=imax;
    if(a[j-1][j-1]==0) a[j-1][j-1]=tiny;
    if(j!=n)
    {
    dum=1.0/a[j-1][j-1];
    for(i=j+1;i<=n;i++)
                a[i-1][j-1]*=dum;
    }//endif
    } //end for j=
    return a;
    }
    public static double[] LUaxb(double a[][],double x[],int indx[])
    {
    //solves AX=B system of linear equation of LU decomposed matrix a
    //(calculated by method LU)
    int ii=0;
    int i,j,ll=0;
    double sum=0;
    int n=a.length;
    double b[];
    b=new double[n];
    for(i=1;i<=n;i++)
    {
                b[i-1]=x[i-1];
    }
    for(i=1;i<=n;i++)
    {
    ll=indx[i-1];
    sum=b[ll-1];
    b[ll-1]=b[i-1];
    if(ii!=0)
                {
                for(j=ii;j<=(i-1);j++)
                {
                sum-=a[i-1][j-1]*b[j-1];
                }
                }
    else if(sum!=0) ii=i;
    b[i-1]=sum;
    }
    for(i=n;i>=1;i--)
    {
    sum=b[i-1];
    if(i<n)
                {
                for(j=(i+1);j<=n;j++)
                {
                sum-=a[i-1][j-1]*b[j-1];
                }
                }
    b[i-1]=sum/a[i-1][i-1];
    }
    return b;
    }
    public void ContourPlot()
    {ContourPlot pp=new ContourPlot( x,y,T,16,true,"Finite element","x","y");pp.plot();  }
    public void Plot3D(double L0,double L,double W0,double W) throws RemoteException, VisADException
    {
    Plot3D.plot3D_CD1(T,L0,L,W0,W,200,"x","y","T degree C");
```

```
}
public void GridPlot(double L0,double L,double W0,double W)
{Plot3DPanel plot = new Plot3DPanel("SOUTH");
double dx=(L-L0)/M;
double dy=(W-W0)/N;
double[] xx = increment(L0, dx,L); // x = 0.0:0.1:1.0
double[] yy = increment(W0,dy,W);
plot.addGridPlot("Finite element Grid", xx, yy, T);
JFrame frame = new JFrame("a plot panel");
frame.setSize(600, 600);
frame.setContentPane(plot);
frame.setVisible(true);
}
public static double[] AXB(double a[][],double b[])
{
//Solution of system of linear equations by LU method
// note that the same calculation can be done by divide method.
int n=a.length;
double c[]=new double[n];
int d[]={1};
int indx[]=new int[n];
double e[][]=new double[n][n];
e=LU(a,indx,d);
c=LUaxb(e,b,indx);
return c;
} //end of AXB
public static void main(String arg[]) throws RemoteException, VisADException
{double L=1.0;
double W=1.0;
int N=16,M=16;
double hi=20.0;
triangular_grid tg=new triangular_grid(L,W,M,N);
int cond0=0;//x==0
int cond1=1;//x==L
int cond2=2;//y==0
int cond3=3;//y==W
double Ti0=100.0;
double Ti1=100.0;
double Ti2=100.0;
double Ti3=500.0;
tg.setTemperature(cond0,0.0,L,0.0,W,Ti0);
tg.setTemperature(cond1,0.0,L,0.0,W,Ti1);
tg.setTemperature(cond2,0.0,L,0.0,W,Ti2);
tg.setTemperature(cond3,0.0,L,0.0,W,Ti3);
double k[]={50,50};
tg.solve(1,k);
Text.print(tg.T);
tg.ContourPlot();
tg.Plot3D(0.0,L,0.0,W);
tg.GridPlot(0.0,L,0.0,W);}}
```



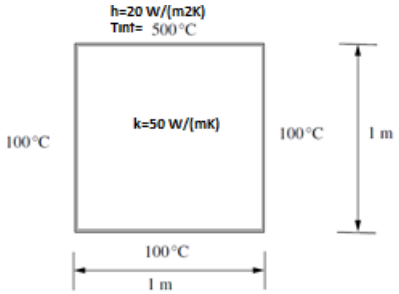
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
100.0	101.717999388...	103.366445666...	104.879686336...	106.199271695...	107.276267646...	108.072499075...	108.560924358...	108.725501712...	108.560924358...	108.072499075...	107.276267646...	106.199271695...	104.879686336...	103.366445666...	101.717999388...	100.0
100.0	103.505551886...	106.868096941...	109.953027984...	112.641132799...	114.833299813...	116.452804296...	117.445696645...	117.780158132...	117.445696645...	116.452804296...	114.833299813...	112.641132799...	109.953027984...	106.868096941...	103.505551886...	100.0
100.0	105.436111215...	110.647362229...	115.423195861...	119.578931704...	122.962994512...	125.459721651...	126.988899793...	127.503737526...	126.988899793...	125.459721651...	122.962994512...	119.578931704...	115.423195861...	110.647362229...	105.436111215...	100.0
100.0	107.591530748...	114.862044896...	121.513461527...	127.288403644...	131.980024879...	135.434188003...	137.546443352...	138.256992385...	137.546443352...	135.434188003...	131.980024879...	127.288403644...	121.513461527...	114.862044896...	107.591530748...	100.0
100.0	110.067966880...	119.695825081...	128.480201708...	136.081196466...	142.234513356...	146.750562129...	149.505693228...	150.431345311...	149.505693228...	146.750562129...	142.234513356...	136.081196466...	128.480201708...	119.695825081...	110.067966880...	100.0
100.0	112.984511690...	125.373086841...	136.630323759...	146.321667155...	154.126269950...	159.827853930...	163.294422119...	164.457002402...	163.294422119...	159.827853930...	154.126269950...	146.321667155...	136.630323759...	125.373086841...	112.984511690...	100.0
100.0	116.496993040...	132.181686834...	146.346339331...	158.448878445...	168.121045360...	175.140161521...	179.387138917...	180.807820059...	179.387138917...	175.140161521...	168.121045360...	158.448878445...	146.346339331...	132.181686834...	116.496993040...	100.0
100.0	120.821773636...	140.510328124...	158.124468287...	173.006461934...	184.768871526...	193.224607876...	198.306151969...	199.999999999...	198.306151969...	193.224607876...	184.768871526...	173.006461934...	158.124468287...	140.510328124...	120.821773636...	100.0
100.0	126.279773380...	150.913383739...	172.634743758...	190.683629478...	204.723370932...	214.683246490...	220.612861082...	222.579876002...	220.612861082...	214.683246490...	204.723370932...	190.683629478...	172.634743758...	150.913383739...	126.279773380...	100.0
100.0	133.383936145...	164.228689694...	190.817493529...	212.369941289...	228.757736233...	240.172146069...	246.882169868...	249.093781844...	246.882169868...	240.172146069...	228.757736233...	212.369941289...	190.817493529...	164.228689694...	133.383936145...	100.0
100.0	143.027281508...	181.799945364...	214.036599373...	239.220905915...	257.765486643...	270.365431685...	277.649890478...	280.030911636...	277.649890478...	270.365431685...	257.765486643...	239.220905915...	214.036599373...	181.799945364...	143.027281508...	100.0
100.0	156.925244522...	205.907210879...	244.308052686...	272.711596355...	292.717872739...	305.874203552...	313.321048722...	315.730083745...	313.321048722...	305.874203552...	292.717872739...	272.711596355...	244.308052686...	205.907210879...	156.925244522...	100.0
100.0	178.766485702...	240.595600944...	284.576804138...	314.599554080...	334.520204405...	347.092461060...	354.030017115...	356.247325898...	354.030017115...	347.092461060...	334.520204405...	314.599554080...	284.576804138...	240.595600944...	178.766485702...	100.0
100.0	217.545097343...	293.131903058...	338.804008841...	366.589611424...	383.670929740...	393.945419167...	399.459232780...	401.199185619...	399.459232780...	393.945419167...	383.670929740...	366.589611424...	338.804008841...	293.131903058...	217.545097343...	100.0
100.0	298.282000611...	375.582905103...	410.917716744...	429.283953033...	439.628483965...	445.559053088...	448.662309221...	449.630951015...	448.662309221...	445.559053088...	439.628483965...	429.283953033...	410.917716744...	375.582905103...	298.282000611...	100.0
500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0





ÖRNEK PROBLEM: ZAMANDAN BAĞIMSIZ ISI İLETİMİ SONLU ELEMANLAR ÇÖZÜMÜ

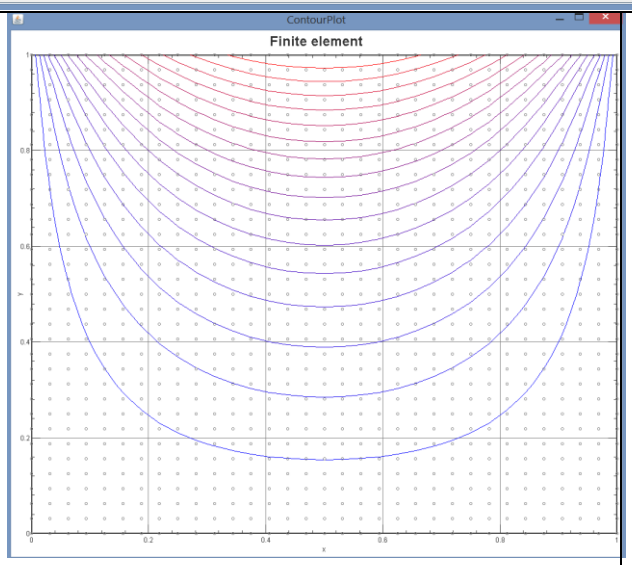
İkinci bir problem daha tanımlayalım. Ancak burada sınır şartı ısı taşınımı sınır şartı ile değiştirilmiştir. Üç yüzey 100 C dir. 4 üncü yüzeyde ise ısı taşınımı sınır şartı verilmiş

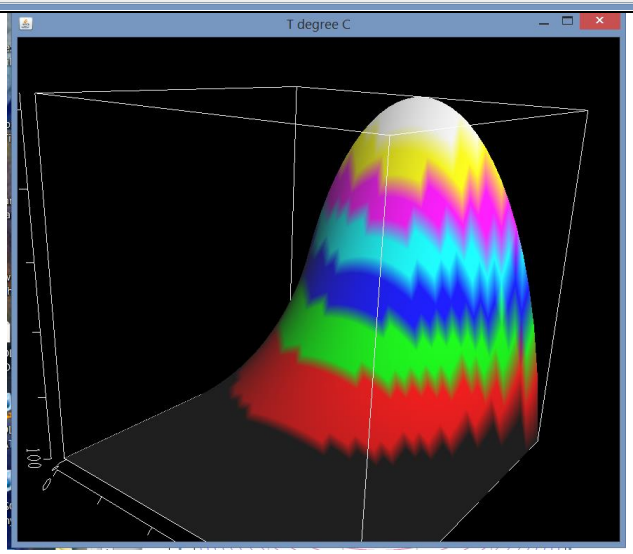


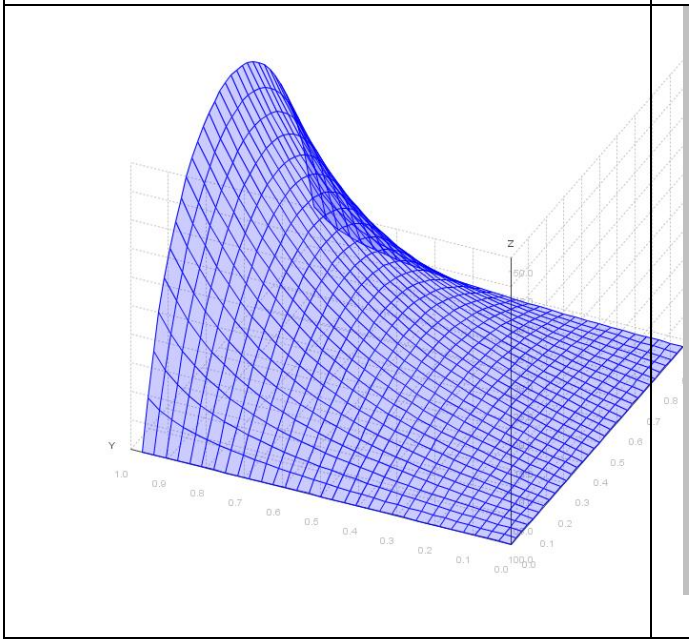

```
public static void main(String arg[]) throws RemoteException, VisADException
{double L=1.0;
double W=1.0;
int N=32,M=32;
double hi=20.0;
triangular_grid tg=new triangular_grid(L,W,M,N);
int cond0=0;//x==0
int cond1=1;//x==L
int cond2=2;//y==0
int cond3=3;//y==W
double Ti0=100.0;
double Ti1=100.0;
double Ti2=100.0;
double Ti3=500.0;
double qi=-5000.0;;
tg.setTemperature(cond0,0.0,L,0.0,W,Ti0);
tg.setTemperature(cond1,0.0,L,0.0,W,Ti1);
tg.setTemperature(cond2,0.0,L,0.0,W,Ti2);
tg.setConvection(cond3,0.0,L,0.0,W,hi,Ti3);
double k[]={50,50};
tg.solve(1,k);
Text.print(tg.T);
tg.ContourPlot();
tg.Plot3D(0.0,L,0.0,W);
tg.GridPlot(0.0,L,0.0,W);
tg.surfacePlot();
}
```

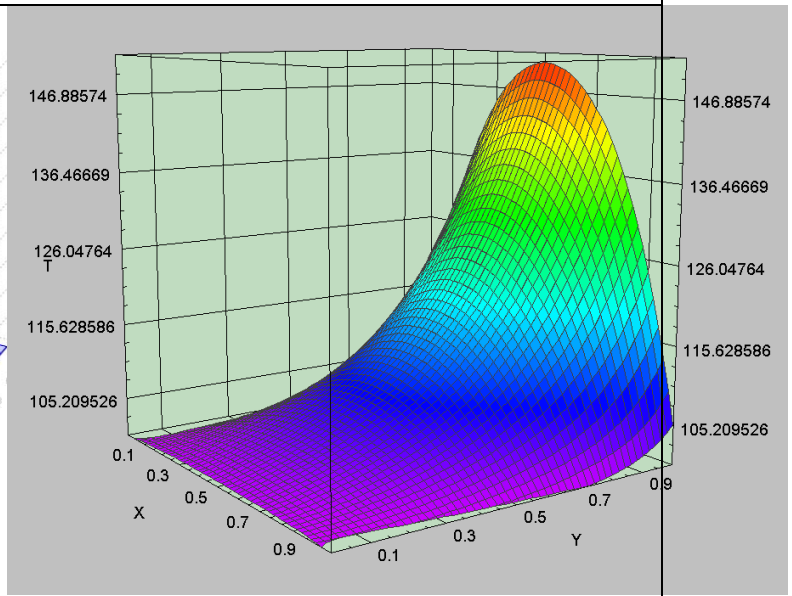
data output(veri çıktısı) :

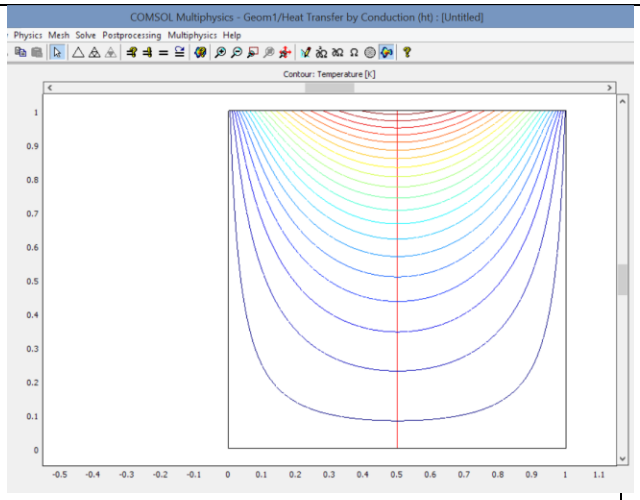
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
100.0	100.04...	100.09...	100.14...	100.18...	100.23...	100.27...	100.31...	100.34...	100.37...	100.40...	100.43...	100.45...	100.46...	100.47...	100.48...	100.48...	100.48...	100.47...	100.46...	100.45...	100.43...	100.40...	100.37...	100.34...	100.310...	100.27...	100.230...	100.18...	100.142...	100.09...	100.047...	100.0
100.0	100.09...	100.19...	100.28...	100.37...	100.46...	100.54...	100.62...	100.69...	100.75...	100.81...	100.86...	100.90...	100.93...	100.96...	100.97...	100.98...	100.97...	100.96...	100.93...	100.90...	100.86...	100.81...	100.75...	100.69...	100.623...	100.54...	100.463...	100.37...	100.285...	100.19...	100.096...	100.0
100.0	100.14...	100.29...	100.43...	100.56...	100.70...	100.82...	100.94...	101.05...	101.14...	101.23...	101.30...	101.37...	101.41...	101.45...	101.47...	101.48...	101.47...	101.45...	101.41...	101.37...	101.30...	101.23...	101.14...	101.05...	100.942...	100.82...	100.700...	100.56...	100.431...	100.29...	100.145...	100.0
100.0	100.19...	100.39...	100.58...	100.76...	100.94...	101.11...	101.27...	101.41...	101.54...	101.66...	101.76...	101.84...	101.91...	101.96...	101.98...	101.99...	101.98...	101.96...	101.91...	101.84...	101.76...	101.66...	101.54...	101.41...	101.270...	101.11...	100.944...	100.76...	100.582...	100.39...	100.196...	100.0
100.0	100.24...	100.49...	100.73...	100.97...	101.19...	101.41...	101.61...	101.79...	101.96...	102.10...	102.23...	102.34...	102.42...	102.48...	102.52...	102.53...	102.52...	102.48...	102.42...	102.34...	102.23...	102.10...	101.96...	101.79...	101.611...	101.41...	101.198...	100.97...	100.738...	100.49...	100.249...	100.0
100.0	100.30...	100.60...	100.90...	101.18...	101.46...	101.72...	101.96...	102.19...	102.39...	102.57...	102.73...	102.86...	102.96...	103.03...	103.07...	103.09...	103.07...	103.03...	102.96...	102.86...	102.73...	102.57...	102.39...	102.19...	101.967...	101.72...	101.463...	101.18...	100.901...	100.60...	100.304...	100.0
100.0	100.36...	100.72...	101.07...	101.41...	101.74...	102.05...	102.34...	102.61...	102.85...	103.06...	103.25...	103.40...	103.52...	103.61...	103.66...	103.68...	103.66...	103.61...	103.52...	103.40...	103.25...	103.06...	102.85...	102.61...	102.343...	102.05...	101.742...	101.41...	101.073...	100.72...	100.362...	100.0
100.0	100.42...	100.84...	101.25...	101.65...	102.03...	102.40...	102.74...	103.05...	103.33...	103.58...	103.80...	103.98...	104.12...	104.22...	104.28...	104.30...	104.28...	104.22...	104.12...	103.98...	103.80...	103.58...	103.33...	103.05...	102.741...	102.40...	102.039...	101.65...	101.256...	100.84...	100.424...	100.0
100.0	100.49...	100.97...	101.45...	101.91...	102.35...	102.77...	103.16...	103.52...	103.85...	104.14...	104.39...	104.59...	104.76...	104.87...	104.94...	104.97...	104.94...	104.87...	104.76...	104.59...	104.39...	104.14...	103.85...	103.52...	103.166...	102.77...	102.355...	101.91...	101.452...	100.97...	100.490...	100.0
100.0	100.56...	101.11...	101.66...	102.18...	102.69...	103.17...	103.62...	104.03...	104.40...	104.73...	105.02...	105.25...	105.44...	105.57...	105.65...	105.68...	105.65...	105.57...	105.44...	105.25...	105.02...	104.73...	104.40...	104.03...	103.623...	103.17...	102.695...	102.18...	101.661...	101.11...	100.561...	100.0
100.0	100.63...	101.26...	101.88...	102.48...	103.06...	103.60...	104.11...	104.58...	105.00...	105.37...	105.70...	105.96...	106.17...	106.32...	106.42...	106.45...	106.42...	106.32...	106.17...	105.96...	105.70...	105.37...	105.00...	104.58...	104.114...	103.60...	103.062...	102.48...	101.888...	101.26...	100.638...	100.0
100.0	100.72...	101.43...	102.13...	102.81...	103.46...	104.07...	104.64...	105.17...	105.65...	106.07...	106.43...	106.73...	106.97...	107.14...	107.24...	107.27...	107.24...	107.14...	106.97...	106.73...	106.43...	106.07...	105.65...	105.17...	104.647...	104.07...	103.460...	102.81...	102.134...	101.43...	100.721...	100.0
100.0	100.81...	101.61...	102.40...	103.16...	103.89...	104.58...	105.22...	105.81...	106.35...	106.82...	107.23...	107.56...	107.83...	108.02...	108.13...	108.17...	108.13...	108.02...	107.83...	107.56...	107.23...	106.82...	106.35...	105.81...	105.226...	104.58...	103.892...	103.16...	102.401...	101.61...	100.811...	100.0
100.0	100.91...	101.81...	102.69...	103.54...	104.36...	105.13...	105.85...	106.51...	107.11...	107.64...	108.09...	108.47...	108.76...	108.97...	109.10...	109.14...	109.10...	108.97...	108.76...	108.47...	108.09...	107.64...	107.11...	106.51...	105.856...	105.13...	104.364...	103.54...	102.693...	101.81...	100.910...	100.0
100.0	101.01...	102.02...	103.01...	103.96...	104.88...	105.74...	106.54...	107.28...	107.94...	108.53...	109.03...	109.45...	109.78...	110.01...	110.15...	110.20...	110.15...	110.01...	109.78...	109.45...	109.03...	108.53...	107.94...	107.28...	106.546...	105.74...	104.880...	103.96...	103.013...	102.02...	101.019...	100.0
100.0	101.13...	102.26...	103.36...	104.43...	105.44...	106.40...	107.30...	108.12...	108.85...	109.50...	110.06...	110.52...	110.89...	111.15...	111.30...	111.35...	111.30...	111.15...	110.89...	110.52...	110.06...	109.50...	108.85...	108.12...	107.301...	106.40...	105.448...	104.43...	103.365...	102.26...	101.138...	100.0
100.0	101.27...	102.52...	103.75...	104.94...	106.07...	107.13...	108.13...	109.03...	109.85...	110.57...	111.19...	111.70...	112.10...	112.38...	112.56...	112.61...	112.56...	112.38...	112.10...	111.70...	111.19...	110.57...	109.85...	109.03...	108.131...	107.13...	106.072...	104.94...	103.754...	102.52...	101.270...	100.0
100.0	101.41...	102.81...	104.18...	105.50...	106.76...	107.94...	109.04...	110.05...	110.95...	111.74...	112.42...	112.98...	113.42...	113.73...	113.92...	113.99...	113.92...	113.73...	113.42...	112.98...	112.42...	111.74...	110.95...	110.05...	109.045...	107.94...	106.762...	105.50...	104.184...	102.81...	101.416...	100.0
100.0	101.57...	103.13...	104.66...	106.13...	107.52...	108.83...	110.05...	111.16...	112.15...	113.02...	113.77...	114.38...	114.86...	115.21...	115.42...	115.49...	115.42...	115.21...	114.86...	114.38...	113.77...	113.02...	112.15...	111.16...	110.054...	108.83...	107.526...	106.13...	104.662...	103.13...	101.579...	100.0
100.0	101.76...	103.50...	105.19...	106.82...	108.37...	109.82...	111.16...	112.39...	113.48...	114.43...	115.25...	115.92...	116.45...	116.82...	117.05...	117.12...	117.05...	116.82...	116.45...	115.92...	115.25...	114.43...	113.48...	112.39...	111.169...	109.82...	108.375...	106.82...	105.195...	103.50...	101.761...	100.0
100.0	101.96...	103.90...	105.79...	107.60...	109.32...	110.92...	112.40...	113.74...	114.94...	115.99...	116.88...	117.61...	118.18...	118.59...	118.83...	118.91...	118.83...	118.59...	118.18...	117.61...	116.88...	115.99...	114.94...	113.74...	112.405...	110.92...	109.321...	107.60...	105.792...	103.90...	101.965...	100.0
100.0	102.19...	104.36...	106.46...	108.47...	110.37...	112.15...	113.77...	115.24...	116.55...	117.69...	118.66...	119.45...	120.07...	120.51...	120.78...	120.87...	120.78...	120.51...	120.07...	119.45...	118.66...	117.69...	116.55...	115.24...	113.777...	112.15...	110.379...	108.47...	106.464...	104.36...	102.196...	100.0
100.0	102.45...	104.88...	107.22...	109.46...	111.56...	113.52...	115.30...	116.91...	118.33...	119.57...	120.62...	121.48...	122.14...	122.62...	122.90...	123.00...	122.90...	122.62...	122.14...	121.48...	120.62...	119.57...	118.33...	116.91...	115.305...	113.52...	111.568...	109.46...	107.225...	104.88...	102.459...	100.0
100.0	102.76...	105.47...	108.09...	110.58...	112.90...	115.05...	117.01...	118.76...	120.30...	121.64...	122.77...	123.69...	124.40...	124.91...	125.22...	125.32...	125.22...	124.91...	124.40...	123.69...	122.77...	121.64...	120.30...	118.76...	117.010...	115.05...	112.909...	110.58...	108.093...	105.47...	102.761...	100.0
100.0	103.11...	106.16...	109.09...	111.85...	114.43...	116.78...	118.91...	120.81...	122.48...	123.92...	125.12...	126.11...	126.87...	127.41...	127.73...	127.84...	127.73...	127.41...	126.87...	126.11...	125.12...	123.92...	122.48...	120.81...	118.918...	116.78...	114.430...	111.85...	109.092...	106.16...	103.113...	100.0
100.0	103.52...	106.97...	110.25...	113.33...	116.16...	118.74...	121.05...	123.10...	124.89...	126.42...	127.71...	128.75...	129.55...	130.12...	130.47...	130.58...	130.47...	130.12...	129.55...	128.75...	127.71...	126.42...	124.89...	123.10...	121.056...	118.74...	116.167...	113.33...	110.255...	106.97...	103.528...	100.0
100.0	104.02...	107.93...	111.62...	115.04...	118.16...	120.96...	123.45...	125.65...	127.55...	129.17...	130.53...	131.63...	132.47...	133.07...	133.43...	133.55...	133.43...	133.07...	132.47...	131.63...	130.53...	129.17...	127.55...	125.65...	123.459...	120.96...	118.161...	115.04...	111.627...	107.93...	104.028...	100.0
100.0	104.65...	109.11...	113.27...	117.06...	120.46...	123.49...	126.16...	128.49...	130.49...	132.20...	133.61...	134.76...	135.64...	136.26...	136.63...	136.75...	136.63...	136.26...	135.64...	134.76...	133.61...	132.20...	130.49...	128.49...	126.163...	123.49...	120.467...	117.06...	113.273...	109.11...	104.651...	100.0
100.0	105.45...	110.60...	115.28...	119.46...	123.15...	126.38...	129.20...	131.64...	133.74...	135.51...	136.97...	138.15...	139.06...	139.69...	140.08...	140.20...	140.08...	139.69...	139.06...	138.15...	136.97...	135.51...	133.74...	131.64...	129.206...	126.38...	123.150...	119.46...	115.286...	110.60...	105.459...	100.0
100.0	106.58...	112.56...	117.80...	122.35...	126.28...	129.69...	132.62...	135.15...	137.30...	139.12...	140.62...	141.82...	142.74...	143.39...	143.78...	143.91...	143.78...	143.39...	142.74...	141.82...	140.62...	139.12...	137.30...	135.15...	132.629...	129.69...	126.287...	122.35...	117.804...	112.56...	106.580...	100.0
100.0	108.30...	115.25...	121.01...	125.85...	129.95...	133.46...	136.46...	139.03...	141.21...	143.05...	144.56...	145.77...	146.70...	147.35...	147.74...	147.87...	147.74...	147.35...	146.70...	145.77...	144.56...	143.05...	141.21...	139.03...	136.466...	133.46...	129.954...	125.85...	121.017...	115.25...	108.301...	100.0
100.0	111.36...	119.14...	125.15...	130.08...	134.21...	137.73...	140.73...	143.30...	145.47...	147.29...	148.80...	150.00...	150.92...	151.57...	151.96...	152.09...	151.96...	151.57...	150.92...	150.00...	148.80...	147.29...	145.47...	143.30...	140.737...	137.73...	134.218...	130.08...	125.159...	119.14...	111.368...	100.0







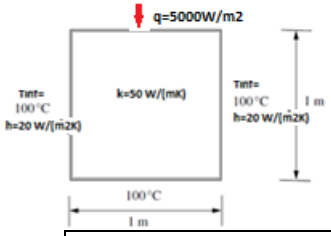




The same problem solved by comsol multiphysics finite element package:

ÖRNEK PROBLEM: ZAMANDAN BAĞIMSIZ ISI İLETİMİ SONLU ELEMANLAR ÇÖZÜMÜ

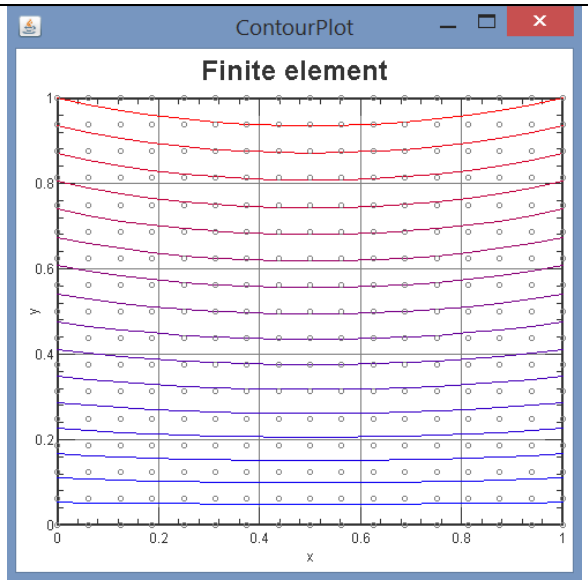
Bu problemde bir yüzey sabit sıcaklıkta verilmiş, 2 yüzeyde ısı taşınım sınır şartı ve bir yüzeyde sabit ısı akısı verilmiş.

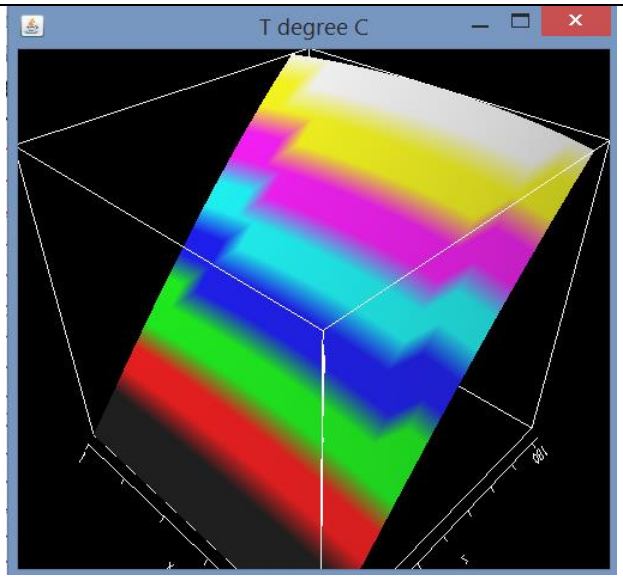



```
public static void main(String arg[]) throws RemoteException, VisADException
{double L=1.0;
double W=1.0;
int N=16,M=16;
double hi=20.0;
triangular_grid tg=new triangular_grid(L,W,M,N);
int cond0=0;//x==0
int cond1=1;//x==L
int cond2=2;//y==0
int cond3=3;//y==W
double Ti0=100.0; //degree C
double Ti1=100.0; //degree C
double Ti2=100.0; //degree C
double qi=-5000.0; //W/m2
tg.setTemperature(cond2,0.0,L,0.0,W,Ti0);
tg.setConvection(cond0,0.0,L,0.0,W,hi,Ti1);
tg.setConvection(cond1,0.0,L,0.0,W,hi,Ti2);
tg.setHeatFlux(cond3,0.0,L,0.0,W,qi);
double k[]={50,50};
tg.solve(1,k);
Text.print(tg.T);
tg.ContourPlot();
tg.Plot3D(0.0,L,0.0,W);
tg.GridPlot(0.0,L,0.0,W);
}
```

data output(veri çıktısı) :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
104.1853...	104.2838...	104.3696...	104.4425...	104.5023...	104.5490...	104.5823...	104.6024...	104.6091...	104.6024...	104.5823...	104.5490...	104.5023...	104.4425...	104.3696...	104.2838...	104.1853...
108.3830...	108.5803...	108.7521...	108.8981...	109.0179...	109.1113...	109.1780...	109.2181...	109.2315...	109.2181...	109.1780...	109.1113...	109.0179...	108.8981...	108.7521...	108.5803...	108.3830...
112.6056...	112.9022...	113.1605...	113.3799...	113.5599...	113.7001...	113.8004...	113.8607...	113.8807...	113.8607...	113.8004...	113.7001...	113.5599...	113.3799...	113.1605...	112.9022...	112.6056...
116.8656...	117.2624...	117.6078...	117.9011...	118.1416...	118.3290...	118.4629...	118.5433...	118.5701...	118.5433...	118.4629...	118.3290...	118.1416...	117.9011...	117.6078...	117.2624...	116.8656...
121.1759...	121.6739...	122.1073...	122.4751...	122.7765...	123.0112...	123.1789...	123.2796...	123.3131...	123.2796...	123.1789...	123.0112...	122.7765...	122.4751...	122.1073...	121.6739...	121.1759...
125.5493...	126.1501...	126.6725...	127.1154...	127.4781...	127.7604...	127.9620...	128.0829...	128.1232...	128.0829...	127.9620...	127.7604...	127.4781...	127.1154...	126.6725...	126.1501...	125.5493...
129.9995...	130.7045...	131.3171...	131.8358...	132.2603...	132.5902...	132.8257...	132.9669...	133.0139...	132.9669...	132.8257...	132.5902...	132.2603...	131.8358...	131.3171...	130.7045...	129.9995...
134.5402...	135.3515...	136.0554...	136.6507...	137.1370...	137.5145...	137.7837...	137.9450...	137.9987...	137.9450...	137.7837...	137.5145...	137.1370...	136.6507...	136.0554...	135.3515...	134.5402...
139.1863...	140.1059...	140.9023...	141.5744...	142.1225...	142.5472...	142.8496...	143.0306...	143.0909...	143.0306...	142.8496...	142.5472...	142.1225...	141.5744...	140.9023...	140.1059...	139.1863...
143.9534...	144.9837...	145.8734...	146.6223...	147.2313...	147.7022...	148.0369...	148.2370...	148.3036...	148.2370...	148.0369...	147.7022...	147.2313...	146.6223...	145.8734...	144.9837...	143.9534...
148.8587...	150.0020...	150.9854...	151.8100...	152.4782...	152.9934...	153.3588...	153.5770...	153.6495...	153.5770...	153.3588...	152.9934...	152.4782...	151.8100...	150.9854...	150.0020...	148.8587...
153.9216...	155.1801...	156.2564...	157.1539...	157.8781...	158.4344...	158.8280...	159.0626...	159.1405...	159.0626...	158.8280...	158.4344...	157.8781...	157.1539...	156.2564...	155.1801...	153.9216...
159.1654...	160.5402...	161.7062...	162.6714...	163.4457...	164.0382...	164.4561...	164.7048...	164.7873...	164.7048...	164.4561...	164.0382...	163.4457...	162.6714...	161.7062...	160.5402...	159.1654...
164.6195...	166.1093...	167.3566...	168.3797...	169.1952...	169.8165...	170.2536...	170.5132...	170.5993...	170.5132...	170.2536...	169.8165...	169.1952...	168.3797...	167.3566...	166.1093...	164.6195...
170.3272...	171.9208...	173.2314...	174.2956...	175.1389...	175.7791...	176.2284...	176.4950...	176.5834...	176.4950...	176.2284...	175.7791...	175.1389...	174.2956...	173.2314...	171.9208...	170.3272...
176.3668...	178.0152...	179.3526...	180.4323...	181.2856...	181.9325...	182.3861...	182.6551...	182.7442...	182.6551...	182.3861...	181.9325...	181.2856...	180.4323...	179.3526...	178.0152...	176.3668...





Şimdi de zamana bağlı ısı değişimi denklemini sonlu elemanlar yöntemiyle nasıl çözebileceğimize bakalım.

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right)+g-\rho C_p \frac{\partial T}{\partial t}=0$$

Sınır şartları

$T=T_b$ yüzey S_1'de

$$k_x \frac{\partial T}{\partial x}\vec{l}+k_y \frac{\partial T}{\partial y}\vec{m}+q=0 \quad t>0 \text{ yüzey } S_2\text{'de}$$

$$k_x \frac{\partial T}{\partial x}\vec{l}+k_y \frac{\partial T}{\partial y}\vec{m}+h(T-T_a)=0 \quad t>0 \text{ yüzey } S_2\text{'de}$$

$T=T_0$ x=0 da

Şimdi Galarkin metodunu bu diferansiyel denkleme uygulayabiliriz.

$\int_\Omega w_k L(T)d\Omega=0$ burada Ω elemanın hacmidir.

$$\int_\Omega N_k\left[\frac{\partial}{\partial x}\left(k_x\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_y\frac{\partial T}{\partial y}\right)+g-\rho C_p\frac{\partial T}{\partial t}\right]d\Omega=0$$

Sunlu elemandaki herhangi bir noktanın sıcaklığı T=[N]{T} zamana göre türev $\frac{\partial T}{\partial t}=[N]\frac{d\{T\}}{dt}$ ve integral denklemi kısmi integrasyonla dönüştürülürse:

$\int udv=uv-\int vdu$ so $\int_\Omega \alpha_1\frac{\partial\alpha_2}{\partial x_1}d\Omega=\int_s \alpha_1\alpha_2 dS-\int_\Omega\frac{\partial\alpha_1}{\partial x_1}\alpha_2 d\Omega$

$\int_\Omega N_k\left[\frac{\partial}{\partial x}\left(k_x\frac{\partial T}{\partial x}\right)\right]d\Omega=\int_s N_k\left(k_x\frac{\partial T}{\partial x}\right)dS-\int_\Omega\frac{\partial N_k}{\partial x}k_x\frac{\partial N_m}{\partial x}Td\Omega$ Bunu diferansiyel denkleme uygularsak:

$$-\int_\Omega\left[k_x\frac{\partial N_k}{\partial x}\frac{\partial N_m}{\partial x}+k_y\frac{\partial N_k}{\partial y}\frac{\partial N_m}{\partial y}\right]\{T_m\}d\Omega-\int_\Omega\rho C_p N_k N_m\left\{\frac{dT_m}{dt}\right\}d\Omega+\int_\Omega gN_k d\Omega+\int_s N_k\left(k_x\frac{\partial T}{\partial x}+k_y\frac{\partial T}{\partial y}\right)dS=0$$

Sınır şartlarından:

$$\int_s N_k\left(k_x\frac{\partial T}{\partial x}+k_y\frac{\partial T}{\partial y}\right)dS=-\int_s N_k\left(q+h(T-T_a\right)dS=-\int_s N_k q dS-\int_s N_k hN_m\{T_m\}dS+\int_s N_k hT_a dS$$

$$-\int_\Omega\left[k_x\frac{\partial N_k}{\partial x}\frac{\partial N_m}{\partial x}+k_y\frac{\partial N_k}{\partial y}\frac{\partial N_m}{\partial y}\right]\{T_m\}d\Omega-\int_\Omega\rho C_p N_k N_m\left\{\frac{dT_m}{dt}\right\}d\Omega-\int_s N_k hN_m\{T_m\}dS+$$

$$+\int_\Omega gN_k d\Omega-\int_s N_k qdS\int_s N_k hT_a dS=0$$

Denklemin matris formuform :

$$[C]\left\{\frac{dT}{dt}\right\}+[K]\{T\}=\{f\}\ \text{burada}$$

$$[C]=\int_\Omega\rho C_p N_k N_m d\Omega$$

$$[K]=-\int_\Omega\left[k_x\frac{\partial N_k}{\partial x}\frac{\partial N_m}{\partial x}+k_y\frac{\partial N_k}{\partial y}\frac{\partial N_m}{\partial y}\right]d\Omega+\int_S hN_k\left(N_m\right)dS$$

$$\{f\}=\int_\Omega gN_k d\Omega-\int_s N_k qdS+\int_S hN_k T_a dS$$

$$[C]=\int_\Omega\rho C_p[N]^T[N]d\Omega$$

$$[K]=-\int_\Omega[B]^T[D][B]d\Omega+\int_S h[N]^T[N]dS\ \text{where}$$

$$[D]=\begin{bmatrix}k_x & 0\\ 0 & k_y\end{bmatrix}$$

$$\{f\}=\int_\Omega g[N]^T d\Omega-\int_s[N]^T qdS+\int_S h[N]^T T_a dS$$

$$\int_l N_i^a N_j^b dl=\frac{a!b!}{(a+b+1)!}l\qquad\int_l N_i^2 dl=\frac{2!0!}{(2+0+1)!}l=\frac{2l}{6}\qquad\int_l N_i\,N_j dl=\frac{1!1!}{(1+1+1)!}l=\frac{l}{6}$$

$$\int_A N_i^a N_j^b N_k^c dA = \frac{a!b!c!}{(a+b+c+2)!}2A \quad \int_A N_i^2 N_j^0 dA = \frac{2!0!}{(2+0+2)!}2A = \frac{2A}{12} \quad \int_A N_i^1 N_j^1 dA = \frac{1!1!}{(1+1+2)!}2A = \frac{A}{12}$$

Görüleceği gibi temel denklem hala zamana bağlı bir diferansiyel denklem. Bunu çözmek için sonlu farklar metoduna başvurabiliriz.

$[C]\left\{\frac{\{T\}^{p+1}-\{T\}^p}{\Delta t}\right\}+\phi[K]\{T\}^p+(1-\phi)[K]\{T\}^{p+1}=\{f\}$ burada $t = p\Delta t$ zaman ve ϕ is açık kapalı metod katsayısı, eğer $\phi = 1$ ise açık metod,eğr $\phi = 0$ ise kapalı metod ve eğer $\phi = 0.5$ ise Crank-Nicolson metodu adını alır. Katsayılar k,h,g, ρ , C_p ve q sabit olarak alınabilir, ancak her zaman stepinde değeri değiştirilebilir. Denlem p+1 stepi için bu stepteki sıcaklık profili çözülecek şekilde organize edilebilir.

$$\left([C]+(1-\phi)\Delta t[K]\right)\{T\}^{p+1}=([C]-\phi\Delta t[K])\{T\}^p+\Delta t\{f\}$$

Şimdi üçgen sonlu elemanlar için şekil faktörü değerlerini denkleme yerleştirebiliriz.

$$[C]=\frac{\rho C_p At}{12}\begin{bmatrix}4&1&1\\1&4&1\\1&1&4\end{bmatrix}$$

$$[K]=\frac{t}{4A}\left\{k_x\begin{bmatrix}b_i^2&b_ib_j&b_ib_k\\b_ib_j&b_j^2&b_jb_k\\b_ib_k&b_jb_k&b_k^2\end{bmatrix}+k_y\begin{bmatrix}c_i^2&c_ic_j&c_ic_k\\c_ic_j&c_j^2&c_jc_k\\c_ic_k&c_jc_k&c_k^2\end{bmatrix}\right\}+\frac{htl_{ij}}{6}\begin{bmatrix}2&1&0\\1&2&0\\0&0&0\end{bmatrix}+\frac{htl_{jk}}{6}\begin{bmatrix}0&1&0\\0&2&1\\0&1&0\end{bmatrix}++\frac{htl_{ki}}{6}\begin{bmatrix}2&1&0\\0&0&0\\0&1&2\end{bmatrix}$$

$$\{f\}=\frac{gAt}{3}\begin{Bmatrix}1\\1\\1\end{Bmatrix}-\frac{qtl_{ij}}{2}\begin{Bmatrix}1\\1\\0\end{Bmatrix}-\frac{qtl_{jk}}{2}\begin{Bmatrix}0\\1\\1\end{Bmatrix}-\frac{qtl_{ik}}{2}\begin{Bmatrix}1\\0\\1\end{Bmatrix}+\frac{hT_atl_{ij}}{2}\begin{Bmatrix}1\\1\\0\end{Bmatrix}+\frac{hT_atl_{jk}}{2}\begin{Bmatrix}0\\1\\1\end{Bmatrix}+\frac{hT_atl_{ik}}{2}\begin{Bmatrix}1\\0\\1\end{Bmatrix}$$

İlk başlangıç sıcaklık profilinin başlangıçta yüklenmesi gerekmektedir her zaman adımında yeni bir sıcaklık profili hesaplanır.

11 Referanslar

1. Chapra, Steven C.; Canale, Raymod P., “Numerical Methods for Engineers”, sixth Edition, 2010,McGraw-Hill, ISBN 978-007-126759-5
2. Pres, W. H.,Teulkolsky, S. A.,Vetterling, W. T., Flannery B. P.; “Numerical Recipes in C”, 1992, Cambridge University Pres, ISBN 0-521-43108-5
3. Borse, G.J.;”Numerical Methods with MATLAB” 1997,PWS Publishing company, ISBN 0-534-93822-1
4. Ralston, A.;Rabinowitz, P;”A First Course in Numerical Analysis”, second Edition, 1978, McGraw-Hill, ISBN 007-051158-6
5. Atkinson, K.E.;”An Introduction to Numerical Analysis”, Second edition, 1989, John Wiley & Sons, ISBN 0-471-62489-6
6. Burden, R.L.,Faires,D.F.;”Numerical Analysis”, Eight Edition, 2005,Thomson Brooks/Cole, ISBN 0-534-39200-8
7. Mathews,J.H.,Fink,K.D.;”Numerical Methods Using MATLAB”,1999,Third Edition, Prentice Hall, ISBN 0-13-270042-5
8. Schilling,R.J.,Haris, S. L.”Applied Numerical Methods for Engineers”,2000, Brooks/Cole, ISBN 0-534-37014-4
9. Dahlquist, G.,Anderson N.;”Numerical Methods,1974,Prentice Hall, ISBN 0-13-627315-7
10. Mathews J. H.,”Numerical Methods for Mathematics, Science and Engineering”, Second Edition, 1992, Prentice-Hall, ISBN 0-13-625047-5
11. Michalewicz, Z., “Genetic Algorithms+Data Structures = Evaluation Programs”, second extended edition, 1992, Springer-Verlag, ISBN 3-540-58090-5
12. Karaoğlu, B.,”Sayisal Fizik”,2004, Seyir Yayincilik, ISBN 975-92544-9-2
13. Bayram, M,”Nümerik Analiz”, 2002, “Aktif Yayinevi”, ISBN 975-6755-50-4
14. Tapramaz, R.,”Sayisal Çözümleme”, 2002, Literatür Yayinlari, ISBN 975-8431-82-X
15. Scheid, F.,”SCHAUM’S OUTLINES Nümerik Analiz”, 1988,Nobel Yayim Dağitim, ISBN 975-591-078-6
16. Fox, L., Mayers D. F., “Computing Methods for Scientist and Engineers”, 1968, Oxford University Press
17. Johnson, R. L.,”Numerical Methods A Software Aproach”, 1982, John Willey & Sons
18. Mitchell, M,” An Introduction to Genetic Algorithms”1999, A Bradford Book The MIT Pres, ISBN 0-262-13316-4
19. Arnold, D. N. , “A Concise Introduction to Numerical Analysis”, 1991
20. Nash, J.C. ,”A Compact Methods for Computers Lineer Algebra and Function Minimisation”1990, Adam Hilger, ISBN 0-85274-318-1
21. Chenney, W.,Kinkaid, D.;”Numerical Mathematics and Computing”, Third Edition, Brooks/Cole publishing
22. Çoban, M. Turhan, “Java Programlama Kılavuzu, Herkes için ”, 2000, Alfa yayinevi, ISBN 975-316-631-1
23. Deitel & Deitel, Java How to Program, Second Edition, Prentice Hall, http://www.deitel.com, second edition,1998, ISBN 0-13-906249-1
22. Naughton, Patrick, t Schildt, Herbert ;”Java 1.1 The Complete Reference”, Second Adition, 1998, Mc Graw Hill, ISBN 0-07-882436-2
23. Zukowski, John; “Mastering Java 1.2 (Renamed Java 2 by sun microsystems)”,1998, Sybex publishing, ISBN 0-7821-2180-2
24. Learning Java, Covers Java 1.3, Patrick Niemeyer & Jonathan Knudsen, O’Reilly , 2000, ISBN 1- 56592-718-4
25. Java Swing, Robert Eckstein, Marc Loy & Dave Wood, O’Reilly , 1998, ISBN 1-56592-455-X
26. JDBC Database Access with Java, The Java Series, Graham Hamilton, Rick Cattell, Maydene Fisher, Addison Wesley, 1997, ISBN 0-201-30995-5
27. Java Algorithms, Scott Robert Ladd, Mc Graw-Hill, 1998, ISBN 0-07-913696-6
28. Fundamentals of Database Systems, Elmasri & Navathe, Benjamin/Cummings, 1994, ISBN 0-8053-1753-8
29. Thinking in Java, Bruce Eckel, Prentice Hall, 2000, ISBN 013-659723-8
30. VisAD graphic package, http://www.ssec.wisc.edu/~billh/visad.html
31. Java Development kit ve 3D graphic package, http://java.sun.com
32. ALGLIN-NET internet site, (Gauss- Hermit integration) http://www.alglib.net/integral/gq/ghermite.php
33. ALGLIN-NET internet site, (Gauss-Laguerre integration) http://www.alglib.net/integral/gq/glaguerre.php
34. Gander, W. ,W. Gautschi, "Adaptive Quadrature - Revisited", BIT, Vol. 40, 2000, page. 84-101. http://www.inf.ethz.ch/personal/gander
35. H.J. Korsch H.-J. Jodl T. Hartmann,“Chaos, A Program Collection for the PC”, Springer publishers, 2008, ISBN 978-3-540-74866-3

36. Forman S. Acton, “Numerical Methods that work”, Mathematical Association of America, 1990, ISBN 0-88385-450-3
37. Alfio Quarteroni, Riccardo Sacco, Fausto Saleri, “Numerical Mathematics”, Springer and Verlag, 1991, ISBN 0-387-98959-5
38. Perviz Moin, “Fundamentals of Enginering Numerical Analysis”, Cambridge publications, 2010, ISBN 978-0-521-88432-7
39. J.H. Verner, Numerically optimal Runge--Kutta pairs with interpolants. Numerical Algorithms. **53**, (2010) pp. 383--396.
40. J.H. Verner, Improved Starting methods for two-step Runge--Kutta methods of stage-order p-3, Applied Numerical Mathematics, **10**, (2006) pp. 388--396.
41. J.H. Verner, Starting methods for two-step Runge--Kutta methods of stage-order 3 and order 6, J. Computational and Applied Mathematics, **185**, (2006) pp. 292--307.
42. T. Macdougall and J.H. Verner, Global error estimators for Order 7,8 Runge--Kutta pairs, *Numerical Algorithms* **31**, (2002) pp. 215--231.
43. Z. Jackiewicz and J.H. Verner, Derivation and implementation of two-step Runge-Kutta pairs. *Japan Journal of Industrial and Applied Mathematics* **19** (2002), pp. 227--248.
44. R.W. Lewis, P. Nithiarasu, K. N. Seetheramu, Fundamentals of the Element Method for Heat and Fluid Flow, John Willey & Sons, 2004, ISBN 0-470-84788-3
45. Computational Methods for Algebraic Spline Surfaces , Tor Dokken, Bert Jüttler, 2000, ISBN 3-540-23274-5, Springer Berlin Heidelberg New York
46. Peter Lancaster, Kestutis Salkauskas, Curve and Surface Fitting, an introduction, Academic Press, Harcourt Brace Jovanovich Publishers, 1986, ISNB 0-12-436060-2
47. Bength Fornberg, “Generation of Finite Difference Formulas on Arbitrary Spaced Grids", Mathematics of Computation, Volume 51, Number 184, October 1988, pages 699-706
48. S. Abbasbandy, Improving Newton-Raphson method for nonlinear equations by modified Adomian decomposition method, Appl. Math. Comput. 145 (2003), pp. 887-893.
49. Shenfeng Li, Rujing Wang, Two fourth order Iterative Methods Based on Continued Fraction for Root-finding Problems, World Academy of Science, Engineering and Technology 60 2011
50. Hasan, Mohammed A., On the Derivation of Higher Order Root-finding Methods, Proceedings of the 2007 American Control Conference, ThA07.3, New York City, USA, July 11-13 2007
51. Xin-She Yang.: A New Metaheuristic Bat-Inspired Algorithm. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence Vol. 284 (2010) 65-74.
52. Xin-She Yang, Engineering Optimisation, An introduction with Metaheuristic applications, John Willey & Sons , 2010, ISBN 978-0-470-58246-6
53. Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search", *Simulation*, 76, 60-68 (2001).
54. Ashrafi, S.M., Dariane, A.B., Performance evaluation of an improved harmony search algorithm for numerical optimization: Melody Search (MS). Eng. Appl. Artif. Intel. (2012), http://dx.doi.org/10.1016/j.engappai.2012.08.005
55. Parikshit Yadav, Rajesh Kumar, S.K. Panda, C.S. Chang, “An Intelligent Tuned Harmony Search algorithm for optimization”, Information Sciences 196(2012) 47-72 elsevier publ.
56. Yang, X. S. and Deb, S., 2009. ‘Cuckoo search via Levy flights’, *Proceeings of World Congress on Nature & Biologically Inspired Computing* (NaBIC 2009,India), IEEE Publications, USA, pp. 210-214.
57. [1] Civicioglu P., Transforming geocentric cartesian coordinates to geodetic coordinates by using differantial search algorithm, Computers & Geosciences 46, 229-247, 2012
58. [2] Vito., T., Elio, T., Kevin, M.P., et al., Swarm cognition: an interdisciplinary approach to the study of self-organising biological collectives, Swarm Intelligence 5, 3-18, 2011
59. [1] Wu, B., Qian, C., Ni, W., Fan, S., Hybrid harmony search and artificial bee colony for global optimization problems, Computers and Mathematics with Applications 64, 2621-2634, 2012
60. [2] Pan, Q., Suganhtan, P.H., Tasgetiren M.F., et al., A self – adaptive global best harmony search algorithm for continuous optimization problems, Appl. Math. Comput. 216, 830-848, 2010
61. [3] Kennedy, J., Eberhart, R.C., Shi, Y., Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, 2001
62. [1] J, Sun., B, Feng., W, Xu., “Particle Swarm Optimization with Particles Having Quantum Behavior”, *IEEE Proc. Of Congress on Evolutionary Computation*, 2004
63. [2] J, Sun., W, Xu., B, Feng., “Global Search Strategy of quantum Behaved Particle Swarm Optimization”, *IEEE Proc.of Conference on Cybernatics and Intelligent Systems*, Singapore, 2004

64. [3] van den Bergh, F., A, Engelbrecht., A new locally convergent particle swarm optimizer. In Proceedings of the IEEE Conference on systems, Man, and Cybernatics Hammamet, Tunisia, 1-6, 2002
65. [4] He, S., Q, Wu., J, Wen, J. Saunders., R, Baton., A particle swarm optimizer with passive congregation, Biosystems 78, 135-147, 2004
66. [5] Angeline, P.J., Evolutionary optimization versus particle swarm optimization: Phlosophy and performance differences. Proceedings of the 7th Annual Conference on Evoltionary Programming, San Diego, USA , 601-610, 1998a
67. [6] Xie, X.F., W.J. Zhang ve Z.L. Yang, Adaptive particle swarm optimization on individual level , 6th International Conference on Signal Processing, 1215-1218, 2002a
68. [7] He, Q., C, Han., An improved particle swarm optimization algorithm with disturbance term. Comput. Intell. Bioinfo. 4115, 100-108, 2006
69. [8] Padhye, N., Topology optimization of compliant mechanism using multi-objective particle swarm optimization. Proceedings of the Conference Companion on Genetic and Evolutionary Computation, 2008
70. [9] Jiao , B., Z. Lian and X. Gu, A dynamic inertia weight particle swarm optimization algorithm. Chaos, Solitons Fract. 37, 698-705
71. [10] Engelbracht A., Heteregenous particle swarm optimization. Proceedings of the 7th International conference on Swarm Intelligence, 191-202, 2010
72. [11] Chuang, L.Y., S.W. Tsai., C.H. Yang., Chaotic catfish particle warm optimization for solving global numerical optimization problems, Appl. Math. Comput. 217, 6900-6916, 2011
73. [12] Clerc, M., Kennedy, J., "The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space." IEEE Transactions on Evolutionary Computation, vol. 6, no.1, pp.58-73, 2002
74. [13] Sun, J., Fang, W., Wu, X.J., Palade, V., Xu, W.B., Quantum-behaved particle swarm optimization: analysis of the individual particle's behavior and parameter selection, Evolutionary Computation, doi:10.1162/EVCO_a_00049.
75. [14] Su, J., Fang, W., Palade V., Wu, X., Xu, W., Quantum behaved particle swarm optimization with Gaussian distributed local attractor point, Applied Mathematics and Computation 218, 3763-3775, 2011
76. [15] Xi, M., Sun,J., Xu, W., An improved quantum behaved particle swarmoptimization algorithm with weighted mean best position, Applied Mathematics and Computation 205, 751-759, 2008
77. [16] Coelho, L.D.S., A quantum particle swarm optimizer with chaotic mutation operator, Chaos, Solitons & Fractals 37, 1409-1418, 2008
78. [17] Liu B., Wang, L., Jin Y.H., Tang, F., Huang D.X., Directing orbits of chaotic systems by particle swarm optimization, Chaos, Solitons & Fractals 29(2) , 454-461, 2006
79. [18] Zaslavski ,G.M., "The Simplest case of a strange attractor ",*Phys. Lett. A* 69 (3), 145-147, 1978
80. [19] Yang, X., Shi, P., Shen, W., Jiang, K., Pang, S., Multiobjective Quantum-behaved Particle Swarm Optimization with Entopy Based Density Assesment and Chaotic Mutation Operator, Journal of Computational Information Systems 9: 10 , 3873-3881, 2013
81. [20] Tavazoei, M.S., Haeri, M., Comparison of different one dimensional masps as chaotic search pattern in chaos optimization algorithms, Appl. Math. Comput. 187, 1076-1085, 2007
82. [21] Hilborn, R.C., Chaos and nonlinear dynamics: an introduction for scientists and engineers, 2nd ed. New York: Oxford Univ. Press; 2004
83. [22] May, R.M., Simple mathematical models with very complicated dynamics, Nature 261, 459-467, 1976
84. [23] Li, Y., Deng, S., Xiao, D., A ovel Hash algorithm construction based on chaotic neural network, Neural. Comput. Applic. 20, 133-141, 2011
85. [24] Ott, E., Chaos in dynamical systems, Cambridge, U.K. : Cambridge University Press; 2002
86. [25] Tomida, A.G., Matlab Toolbox and GUI for Analyzing One-Dimensional Chaotic Maps, In: International Conference on Computational Sciences and Its Applications ICCSA, IEEE Press, 321-330, 2008
87. Heat Conduction, M. Necati Özışık, Second Edition, John Wiley & Sons, Inc. ISBN 0-471-53256-8
88. Fundamentals of the Finite Element Method for Heat and Fluid Flow, Roland W. Lewis, Perumal Nithiarasu, Kankanhally N. Seetharamu, Wiley publishers, ISBN 0-470-847-88-3
89. Generation of Finite Difference Formulas on Arbitrary Spaced Grids, Bength Fornberg, Mathematics of Computation, Volume 51, Number 184, October 1988, pp 699-706