EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

DOĞALGAZ YAKITLI KATI OKSİTLİ YAKIT PİLİ SİSTEMİNİN BİLGİSAYAR BENZEŞİMİ

Abdullah S. TAZEBAY

Makina Mühendisliği Anabilim Dalı

Bilim Dalı Kodu:625.05.00

Tezin Sunulduğu Tarih:12.06.2008

Tez Danışmanı: Yrd. Doç. Dr. Mustafa Turhan ÇOBAN

Bornova-İZMİR

II

Abdullah S. Tazebay tarafından Yüksek Lisans tezi olarak sunulan "Doğalgaz yakıtlı katı oksitli yakıt pili sisteminin bilgisayar benzeşimi" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 12.06.2008 tarihinde yapılan tez savunma sınavında aday oybirliği ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**: Yrd. Doç. Dr. Mustafa Turhan ÇOBAN**	**.....**
Raportör Üye	**: Prof. Dr. Necdet ÖZBALTA**	**.....**
Üye	**: Doç. Dr. Dilek KUMLUTAŞ**	**.....**

ÖZET

DOĞALGAZ YAKITLI KATI OKSİTLİ YAKIT PİLİ SİSTEMİNİN BİLGİSAYAR BENZEŞİMİ

TAZEBAY, Abdullah S.

Yüksek Lisans Tezi, Makina Mühendisliği Bölümü

Tez Yöneticisi: Yrd. Doç. Dr. Mustafa Turhan ÇOBAN

Haziran 2008, 198 sayfa

Günümüzde verimlilik ve çevre ile ilgili kaygılar yakıt pillerinin mühendisler ve bilim adamları arasında büyük ilgi kazanmasına neden olmuştur. Bu enerji dönüşüm sistemleri içten yanmalı motorlara karşı önemli bir alternatif sunmaktadırlar.

Ancak bu sistemler için uygun Hidrojen depolama tekniklerinin geliştirilememiş olması bu sistemlerin kullanımının yaygınlaşması için önemli bir engel oluşturmaktadır. Yakın gelecekte bu sistemler için uygun ve yeterli bir Hidrojen depolama yönteminin geliştirilmesi mümkün görülmemektedir. Bu sebepten dolayı yakıt pili modülünü besleyecek yakıt dönüştürücülerinin geliştirilmesi daha uygun bir çözüm olarak görülmektedir.

Fakat bu tip yakıt dönüşüm sistemlerinin tasarımı içerdiği hesaplamalar dolayısı ile karmaşıktır ve birçok ısıl ve kimyasal hesabın gerçekleştirilmesi gerektirmektedir. Bu çalışmada bu tip hesaplamaları kolaylaştıracak ve sistemin optimizasyonuna imkân tanıyacak bir bilgisayar programı geliştirilmiştir. Bu bilgisayar programında gerçek gaz davranışlarını modelleyen matematiksel yaklaşımlar kullanılmıştır.

Anahtar sözcükler: Yakıt pili, kimyasal denge analizi, yakıt pili sistemi modellemesi, ototermal yakıt dönüşümü, buharlı yakıt dönüşümü, katı oksitli yakıt pili,

ABSTRACT

COMPUTER SIMULATION OF NATURAL GAS FUELED SOLID OXIDE FUEL CELL

TAZEBAY, Abdullah S.

M.Sc. in Mechanical Engineering

Supervisor: Assistant Professor Mustafa Turhan ÇOBAN

June 2008, 198 pages

Today because of the environmental and efficiency considerations fuel cells are gaining popularity among engineers and scientists. These energy conversion systems can be considered as the most promising alternative of internal combustion engines.

However these systems suffer from appropriate fuel storage methods and it is not expected to find a suitable storage method for Hydrogen in a near future. Because of

this reason today using fuel reformers to feed the fuel cell module is a good alternative solution to commercialize fuel cell.

On the other hand to design such systems is a complex process and needs making several calculations about thermal and chemical properties. In this study computer software was developed to make easy these calculations and required optimizations. This package is developed on the base of mathematical models to determine real gas behaviors.

Keywords: Fuel cell, chemical equilibrium modeling,

X

TEŞEKKÜR

Bölüm Başkanımız Prof. Dr. Ali GÜNGÖR'e, yüksek lisans eğitimim boyunca, herhangi bir konuda, desteğini asla esirgemeyen, bu tez çalışmasının ortaya çıkmasını sağlayan, çalışmayı yönlendiren, kıymetli yorum ve değerlendirmeleriyle tezin biçimlendirilmesine büyük katkı sağlayan Sayın Yrd. Doç. Dr. Mustafa Turhan ÇOBAN' a, tez jürime katılarak bu çalışmaya katkıda bulunan hocalarım Prof. Dr. Necdet ÖZBALTA'ya, ve Doç.Dr. Dilek KUMLUTAŞ'a ve eğitimimi her zaman destekleyen aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

1 GİRİŞ 1

2 temel termodinamik özellikler 4

2.1 Termodinamiğin Temel Kavramları 4

2.1.1 İş.....6

2.1.2 Adyabatik İş 15

2.2 Termodinamiğin Birinci Kanunu 17

2.3 Isı 18

2.3.1 Kapalı Sistemler için Birinci Yasa.....20

2.4 Termodinamiğin İkinci Yasası.....22

2.4.1 Kapalı Sistemler için İkinci Yasa 25

2.5 Temel Denklemler 26

2.5.1 Kapalı ve Açık Sistemler için Temel Denklemler.....26

3 Akışkanların özelliklerinin belirlenmesi 34

3.1 İdeal Gazların Özelliklerinin Belirlenmesi 34

3.2 İdeal Gaz Hal Denklemi 35

3.3 İdeal Gaz Termodinamik Özellikleri 38

3.4 Gerçek Gazların Özellikleri 44

3.4.1 Gerçek Gazların Hal Denklemleri..... 44

3.4.2 Doymuş Buhar Basıncının Hesaplanması.....72

3.5 Suyun Termodinamik Özelliklerinin Belirlenmesi 81

3.5.1 Temel Denklem 82

3.5.2 Termodinamik Özellikler 86

4 Akışkanların fiziksel özellikleri 90

4.1 Akışkanların Viskozitesinin Belirlenmesi.. 90

4.1.1 Gazların Viskozitelerinin Belirlenmesi 90

4.1.2 Sıvı Viskozitesinin Belirlenmesi 93

4.2 Akışkanların Isı İletim Katsayılarının Belirlenmesi 95

4.2.1 Gazların Isı İletim Katsayılarının Belirlenmesi.....95

4.2.2 Sıvıların Isı İletim Katsayılarının Belirlenmesi.....97

4.3 Yüzey Gerilim Katsayısının Belirlenmesi ..98

5 kimyasal dengenin hesaplanması.....101

5.1 Kimyasal Denge.....102

5.2 Kimyasal Dengenin Hesaplama Yöntemleri 107

5.2.1 Gibbs Minimizasyonunda NASA Yöntemi.....110

6 Geliştirilen programlar.....114

6.1 Lee-Kesler 120

6.2 Lee-Kesler Arayüz Programı 127

6.3 Lee-Kesler Karışım Programı 132

6.4 Gibbs Programı 135

6.5 Buharlı Yakıt Dönüştürücü Programı 138

6.6 Ototermal Yakıt Dönüştürücü Programı .. 140

6.7 Pompaların Modellenmesi 142

6.8 Isı Değiştirici Modellemesi 143

6.9 Tüm Sistemin Birlikte Modellenmesi 144

6.10 Örnek Bir Sistemin Benzeşimi 146

7 Sonuç 154

Ekler 154

Ek-1 155

Ek-2 193

ŞEKİLLER DİZİNİ

Şekil Sayfa

Şekil 2-1: Tersinir Adyabatik Eğri 23

Şekil 3-1: (0-1.0 P_r) Genelleştirilmiş sıkıştırılabilirlik faktörü (Prausnitz, 1987) 45

Şekil 3-2: (0-100 P_r) Genelleştirilmiş Sıkıştırılabilirlik faktörü (Prausnitz, 1987).....46

Şekil 3-3: (0-40 P_r) Genelleştirilmiş Sıkıştırılabilirlik Faktörü 47

Şekil 3-4: İki hal arasında gerçekleşebilecek farklı yollar 61

Şekil 5-1: Reaksiyonun oluşumu sırasında Gibbs enerjisinin değişimi.....106

Şekil 6-1: Sıkıştırılabilirlik katsayısının hesaplanması. 125

Şekil 6-2: :Farklı girdiler ile Z nin hesaplanması 126

Şekil 6-3: Lee-Kesler programı arayüz örneği.....128

Şekil 6-4: Lee-Kesler programı hesaplama seçenekleri 129

Şekil 6-5: *T* ve *v* den yola çıkarak özelliklerin belirlenmesi. 130

Şekil 6-6: *T* ve *h* dan yola çıkarak diğer özelliklerin belirlenmesi. 131

Şekil 6-7: Lee-Kesler karışım programında karışımın programa girilmesi. 133

Şekil 6-8: Lee-Kesler karışımda bir örnek çıktı. 134

Şekil 6-9: Kimyasal Denge hesaplama programının arayüzü ve elde ettiği sonuçlar. 136

Şekil 6-10: Kimyasal denge hesabı programının çıkış sıcaklığının hesaplaması..... 137

Şekil 6-11: Sistemin tamamını modelleyen program ara yüzü 145

Şekil 6-12: Akışkan Noktası elemanın özellikleri .. 146

Şekil 6-13:Sistem şeması 147

Şekil 6-14: Yakıt ve Hava bileşenlerinin dosyaya girilmesi. 148

Şekil 6-15: Ototermal yakıt dönüştürücü modelinin girilmesi. 149

Şekil 6-16:Ototermal yakıt dönüştürücünün çıkışı..149

Şekil 6-17:Yakıt pili ile yakıt dönüştürücünün modellenmesi 150

Şekil 6-18: Yakıt pili çıkışı 151

Şekil 6-19: Isı değiştirici çıkışı.....151

Şekil 6-20:Örnek sistem modelinin tamamı.....152

Şekil 6-21:Programdan elde edilen noktalar 153

Şekil 6-22: HEX2 nin sonucu ısı değiştirici boyu...153

ÇİZELGELER DİZİNİ

Çizelge Sayfa

Çizelge 3-1: Çeşitli hal denklemlerinin katsayıları .. 54

Çizelge 3-2: BWR Denkleminde kullanılan basit ve referans gaza ait katsayılar 57

Çizelge 3-3: Denklem (3.100) için sabitler 78

Çizelge 3-4: Buhar denkleminin katsayıları (1-4) 83

Çizelge 3-5: Buhar denklemi katsayıları (4-7) 84

Çizelge 3-6: Buhar denkleminde ki *C* katsayıları..... 86

Çizelge 3-7: Su buharı doyma eğrisi denklemi katsayıları 89

Çizelge 6-1: Daha önceki çalışmalardan alınan program dosyaları..... 115

Çizelge 6-2: Bu çalışma sırasında geliştirilen program dosyaları. .....**Error! Bookmark not defined.**

ÇİZELGELER DİZİNİ (Devam)

Çizelge Sayfa

Çizelge 6-3: Lee-Kesler programında kullanılan değişkenlerin anlamları. 120

Çizelge 6-4: Lee-Kesler programının veritabanında bulunan methanole ait özellikler 122

1 GİRİŞ

Bu tezin amacı, günümüzde birçok araştırmaya konu olan yakıt pili sistemlerinin termodinamik tasarım ve optimizasyonunu kolaylaştıracak bilgisayar programı kodunun geliştirilmesidir.

Yakıt pilleri enerji dönüşüm araçları içinde gelecek vaat eden önemli bir konuma sahiptir. Bu sistemler elektrokimyasal dönüşümle direkt olarak elektrik üretimine imkân vermektedirler. Günümüzde bu sistemlerin ihtiyaç duydukları yakıt gereksinimlerini sağlayabilmek için iki farklı yol bulunmaktadır. Bunlardan ilkinde yakıt saf hidrojen veya metanol, etanol şeklindeki temel hidrokarbon bileşikler halinde yakıt pili modülüne sağlanmalıdır. İkinci seçenekte ise benzin, doğalgaz veya dizel gibi günümüzde yaygın olarak kullanılan kompleks hidrokarbon yakıtlar sistem içinde dönüşüm sürecinden geçirilerek yakıt pilinin ihtiyaç duyduğu hidrojen elde edilir. Bu ikinci yol şu anda yakıt pili sistemlerinin hayata geçirilmesinde en uygun yakıt besleme sistemi olarak görülmektedir. Çünkü günümüzde mevcut olan saf hidrojeni depolama yöntemleri henüz bu sistemlerin ticarileşmesi için ihtiyaç duyulan verimlilikten uzaktır.

Benzin, dizel yahut doğal gaz gibi yakıtların kullanımı sırasında gerçekleştirilmesi gereken dönüşüm işlemi

termokimyasal bir süreçtir ve yakıt pili sisteminin performansını direkt olarak etkiler. Bu dönüşüm işlemi sırasında ısı değiştirgeçleri ve yanma odaları yoğun olarak kullanılır. Bunların yalnız başlarına verimlilikleri dışında sistemin tamamının tasarımından doğacak yanlışlıklar yakıt pillerinin sağladığı verim artışının kaybına sebep olabilir. Bu yüzden bu termokimyasal sistemlerin doğru tasarımı yakıt pili sistemlerinin günümüz enerji dönüşüm araçları karşısındaki başarısını direkt olarak etkilemektedir.

Bu tip sistemler yanma odaları, ısı değiştirgeçleri ve dönüşüm olaylarının gerçekleştiği kimyasal reaktörler barındıran karmaşık sistemlerdir ve bu sistemler ısıl dizaynın yanında detaylı kimyasal dizaynı da gerektirir. Bu tip bir tasarım içinse çok miktarda hesaplamanın yapılması gerekmektedir. Örneğin yanma odasında veya kompleks yakıtın dönüşümünün gerçekleştiği reformer odasında gerçekleşen reaksiyonların sonucunda oluşan ürünlerin gerçeğe uygun bir şekilde modellenebilmesi için çok sayıda hesaplamanın yapılması gerekmektedir. Bu tip hesaplamaların bilgisayar ortamında yapılması hesaplamaların hızlı ve doğru bir şekilde gerçekleştirilmesi için uygun bir yoldur.

Bu tezde bu amaçla bir program geliştirilmeye çalışılmıştır. Bu programın geliştirilmesinde güvenirliği ve kolaylığı ve farklı platformlarda aynı şekilde

çalıştırılabilmeye imkan sağlayan Sun Microsystems firması tarafından geliştirilen ve açık kaynaklı olarak herhangi bir ücrete tabii olmadan internet de sunulan JAVA SDK yazılım geliştirme kiti kullanılmıştır.

2 TEMEL TERMODİNAMİK ÖZELLİKLER

Isıl ve kimyasal sistemlerin denge halindeki modellemeleri termodinamiğin temel kavramlarının ve bu kavramların birbirleriyle olan ilişkilerinin iyi anlaşılmasını gerektirir. Bu nedenle bu temel kavramlardan ve bu kavramları birbirleri ile ilişkilendiren matematiksel ifadelerden söz etmek gerekmektedir.

2.1 Termodinamiğin Temel Kavramları

Termodinamik, bir sistemdeki hal değişimlerinin sebep ve sonuçlarını inceler. Örneğin bir kimyasal reaktördeki reaksiyonun gerçekleştiği sıcaklığın ve sonuçta oluşacak ürünlerin kontrol edilebilmesi için sisteme eklenmesi veya çıkarılması gereken ısı miktarının doğru bir şekilde belirlenmesi termodinamiğin konusudur ve bu çalışmanın temel amacı bu tip hesaplamaları kolay, hızlı ve doğru biçimde yapabilmeyi sağlamaktır. Birçok farklı sistem çeşitli ısı girdisi, çıktısı ve akışkanların hal ve faz değişimlerinden faydalanarak çalışır. Tüm bu sistemler için uygun termodinamik hesaplamalar gerçek sistemin tasarımı için gerekli olan temel parametreleri sağlar.

Sistemdeki hal değişimleri çevre ve sistem arasında gerçekleşen madde veya enerji alış verişinin sonucu olarak ortaya çıkar. Enerji alış verişi ısı veya iş transferinde olduğu

gibi direkt olarak gerçekleşebilir. Ayrıca enerji alış verişi sisteme giren veya çıkan madde tarafından da taşınabilir. Daha önce bahsedilebilirği gibi sistemdeki hal değişimi sadece ölçülebilir basınç (P) ve sıcaklık (T) üzerinden gerçekleşmez. Bu hal değişimi bir kimyasal reaksiyonun oluşmasına veya sistemdeki maddenin hal değişimine sebep olabilir. Yukarıda örnek olarak verilen kimyasal reaktör veya benzeri sistemlerde görülen süreçlerin tasarımı ve yürütülmesi için madde ve enerji transferinin doğru bir şekilde tahmin ve kontrol edilmesi gerekir.

Bu noktada termodinamik bize hal değişimine eşlik eden enerji transferinin belirlenmesinde yardımcı olur. Termodinamik analizi gerçekleştirebilmek için termodinamik sürecin takip ettiği yoldan bağımsız olarak ifade edilebilen ve sistemin ilk ve son durumuna karşılık gelen durum fonksiyonları kullanılır. Bu durum fonksiyonları termodinamik hesaplamaları kolaylaştırır. Çünkü bu fonksiyonlar eksik ifade edilmiş gerçek bir sistemi doğru şekilde ifade edilmiş hal değişimleri olarak modelleme imkânı sunar. Böylece hal değişimleri durum fonksiyonları ile herhangi bir yol boyunca hesaplanabilir. Burada ilk olarak işin ve ısının bu tip durum fonksiyonları ile bağı ve bu ilişkilerin termodinamiğin birinci ve ikinci yasasını nasıl meydana getirdiklerini incelemekte fayda vardır. Bu amaçla sırası ile iş ile ilgili temel kavramlar, bu kavramlardan yola çıkarak kapalı sistemler için

termodinamiğin birinci yasası, yine kapalı sistemler için termodinamiğin ikinci kanunu ve son olarak birinci ve ikinci kanunun açık sistemlere uygulanmasını göreceğiz.

2.1.1 İş

Sistem olarak makroskobik bir cisim düşünüldüğünde bu sistemi belirli bir x mesafesi kadar oynatırken uygulanan bir F kuvveti W büyüklüğünde bir iş yapılmasına sebep olur. Kuvvet hareket doğrultusunda uygulandığında bu işin miktarı şu şekilde hesaplanabilir. Denklem (2.1) (Cengel, 2003)

$$W = \int \mathrm{F} \bullet d\mathrm{x} \tag{2.1}$$

Burada entegralin iç kısımdaki her iki bileşende bir vektörü temsil etmektedir. Dolayısı ile aradaki skaler çarpım bize işin miktarını belirleyen asıl büyüklüğün kuvvet vektörünün yer değiştirme boyunca olan bileşeni olduğunu göstermektedir. Burada gerçekleşen yer değiştirme miktarı diferansiyel bir yer değiştirmeyi ifade ettiğinde sonuçta oluşacak iş miktarı δW ile ifade edilebilir.

$$\delta W = F_x dx \tag{2.2}$$

Daha ileri gitmeden önce W'nin işaretinin belirlenmesi gerekmektedir. Burada herhangi bir seçim

yapılabilir ancak en uygun olanı iş sistem üzerine yapıldığında büyüklüğün pozitif, sistem tarafından yapıldığında ise negatif alınmasıdır.

$W > 0$, iş sistem üzerine yapıldığında

$W < 0$, iş sistem tarafından yapıldığında

Bu seçim tamamen isteğe bağlıdır ve farklı kaynaklarda tersi kabul edilebilir.

Denklem (2.1) tanımlanan iş mekanik durumlar göz önünde bulundurularak örneklendirilebilir. Sistemin dünyanın yer çekimi alanındaki hareketi veya sistemin hızının değişimi yapılan bir iş miktarına karşılık gelir. Yer çekimi kuvveti Newton'un ikinci kanunu ile şu şekilde ifade edilir.

$$F = mg \tag{2.3}$$

kullanılarak z_1 den z_2 ye kadar sistem yüksekliğini değiştirmek için yapılması gereken iş miktarı şu şekilde hesaplanabilir.

$$W = mg\int_{z_1}^{z_2} dz = mg(z_2 - z_1) \equiv \Delta E_p \tag{2.4}$$

Burada z dünyanın merkezinden sistemin merkezine olan uzaklığı ifade etmektedir. Hesaplanan iş miktarı dünyanın çekim potansiyel enerjisindeki değişim miktarı ΔE_p' yi tanımlar. Bu örnekte yapılan iş miktarı sistemi yükseltmek için yapıldığından daha önce belirlediğimiz gibi iş miktarının işareti pozitif olarak alınabilir. Böylece potansiyel enerji artmış olur.

Şimdi de sistemin hızını değiştirmek için gereken iş miktarını hesaplamaya çalışalım. Sistemimiz m kütlesine sahipse ve ilk hız miktarı v ise yine Newton'nun ikinci kanunu gereği dışarıdan etkiyen kuvvetin etkisi şu şekilde hesaplanabilir.

$$F = m\frac{dv}{dt} \tag{2.5}$$

Yine yukarıda olduğu gibi bu kez değişen hız miktarı alınarak v_1 den v_2 ye kadar olan hız değişimi için gereken iş miktarı

$$W = m\int_{x_1}^{x_2} \frac{dv}{dt}dx = m\int_{x_1}^{x_2} \frac{dv}{dt}\frac{dx}{dt}dt \tag{2.6}$$

$$= m\int_{v_1}^{v_2} v dv = \frac{m}{2}(v_2^2 - v_1^2) \equiv \Delta E_k \tag{2.7}$$

Burada hesaplanan iş miktarı da genellikle kinetik enerji değişimi olarak adlandırılır, ΔE_k. Yapılan işin işareti ise daha ince belirlediğimiz prensip gereği sistemin hızının artırılması için dışarıdan hız vektörüne paralele bir kuvvet uygulanması gerektiği için pozitif olmalıdır. Böylece yapılan iş miktarı kinetik enerjide artışa sebep olacaktır. Kinetik ve potansiyel enerji miktarlarının toplamı bize toplam dış enerji miktarını verecektir.

$$\Delta E_{ext} = \Delta E_k + \Delta E_p \tag{2.8}$$

Burada görüldüğü gibi iş miktarındaki değişiklik kavramsal bir değişken olan enerji ile ilişkilendirilmiştir. Enerji miktarındaki değişim belirli bir fiziksel ve ölçülebilir iş miktarına karşılık gelir.

Yukarıda belirtilen mekanik iş formlarına ek olarak birçok farklı iş miktarı bulunmaktadır. Örneğin elektrik akımından doğan yük hareketi, yüzey gerilmesinden doğan yüzey alanı değişimi veya manyetik alanlardan doğan manyetikleşme farklı iş türleri olarak gösterilebilir. Bu tür işler mekanik türdeki işler olarak değerlendirilebilir. Ancak akışkanların termodinamiğinde en yaygın görülen mekanik iş türü sistem sınırındaki değişmeden dolayı gerçekleşen işlerdir. Bu iş türü ile sistemin hacminde değişimler gerçekleşir.

Eğer bu şekil değişimi sırasında sistem sınırına düzgün dağılmış bir kuvvet etkirse sınırı oluşturan alan kullanılarak iş miktarı hesaplanabilir.

$$\delta W = -\frac{F}{A} A dx \tag{2.9}$$

Burada ki $\frac{F}{A}$ ifadesi dış basınca, Adx ise hacim değişimine karşılık gelmektedir. Dolayısı ile (2.9) denklemi şu şekilde yeniden yazılabilir.

$$\delta W = -P_{ext} dV \tag{2.10}$$

Burada P_{ext} dışarıdan etkiyen basıncı dV ise hacimde gerçekleşen değişimi ifade etmektedir. Dışarıdan etkiyen basınç değeri her zaman pozitiftir. Hacim azaldığında dV değeri negatif olacaktır bizim daha önce belirlediğimiz prensip gereği ise dışarıdan sisteme bir iş yapıldığında iş değeri pozitif olmalıdır. Dolayısı ile ifadenin başına bir eksi işareti konulmuştur. Eğer iş sistem tarafından yapılırsa yani dışarıdan gelen basınca karşı sistem de bir hacim artışı gerçekleşirse hacim değişimi pozitif olacaktır ve baştaki eksi işareti bizim belirlediğimiz prensibe göre negatif olacaktır. Sonlu bir V_1 den V_2 ye hacim değişimi için iş şu şekilde hesaplanabilir.

$$W = -\int_{V_1}^{V_2} P_{ext} dV \qquad (2.11)$$

Bu ifade her zaman geçerli olmasına rağmen hacim değişimi sırasında P_{ext} ve V'nin birbirlerine göre değişimlerinin bilinmesini zorunlu kılar. P_{ext} sınır boyunca net basınç farkından doğduğundan sürece bağlı olarak değişir bundan dolayı bir ilişkinin bilinmesi halinde dahi bu ilişki ancak iç basıncın değişmediği bir durum için söz konusu olabilir. Böyle uzun süreli bir süreç de sık karşılaşılan bir durum değildir. Bu yüzden dış basınç yerine sistem basıncını kullanmak daha uygundur ancak iç basınç da tersinmez bir hal değişimi için tanımlanmayabilir. Böyle bir durumda ise tahmini bir $P_{ext}(V)$ ifadesi kullanılabilir. Eğer sistem basıncı P tanımlanabiliyorsa basınç ile hacim arasında hal denklemleri kullanılarak hesaplama yapılabilir. Ancak bu durumda da P ve P_{ext} arasındaki ilişkinin de bilinmesi gerekir. Gerçek bir süreç de bu ikisi arasındaki bağıntı sınırın davranışı ile belirlenir. Buna rağmen şu genellemeler yapılabilir.

Sistemin sıkıştırmaya maruz kaldığını düşünürsek sisteme etkiyen ve iş yapan basıncın sistemin iç basıncından süreç boyunca belirli miktarda fazla olması gerekir.

$$P_{ext} = P + \mathcal{P} \qquad (2.12)$$

Sistemin hacminin genişlemesi durumunda ise bunun tam tersi bir basınç gözlenecektir.

$$P_{ext} = P - \mathcal{P} \tag{2.13}$$

(2.12) ve (2.13) denklemleri sonlu ve tersinmez bir sistem için yazılmıştır. Böyle bir sistem tersinir hale yaklaştırılmak istenir ise yapılacak şey $\mathcal{P}$ basıncını süreç boyunca diferansiyel bir hale getirmek gerekecektir. Tam tersinir bir durumda ise $\mathcal{P}$ =0 olacaktır. Bu durumda

$$P_{ext} = P \tag{2.14}$$

Böylece tersinir bir sistem için iş şu şekilde hesaplanabilir.

$$\delta W_{rev} = -PdV \tag{2.15}$$

Sonlu tersinir bir değişim içinse tersinir iş şu şekilde olacaktır.

$$W_{rev} = -\int_{V_1}^{V_2} PdV \tag{2.16}$$

Son iki denklem idealize edilmiş davranış şekillerini ifade etmektedir ve bu süreçlerin gerçek şartlarda görülmesine imkân yoktur.

Eğer sistemin iki farklı hali tersinir ve tersinmez iki farklı yolla birbirine bağlanabiliyorsa tersinir ve tersinmez iş süreçlerini birbirlerine şu şekilde bağlayabiliriz.

$$\delta W_{irr} = \delta W_{rev} + \delta W_{lost} \tag{2.17}$$

Bu ifadedeki δW_{lost} tersinmezliklerden dolayı kaybedilmiş iş miktarını belirtmektedir.

$$\delta W_{lost} \equiv - \mathcal{P} dV \geq 0 \tag{2.18}$$

Kayıp iş miktarı tersinir bir süreç için sıfırdır. Aksi takdirde yani bir tersinmez iş gerçekleşmesi durumunda ise her zaman pozitiftir. Bu sonuca da şu şekilde ulaşılabilir. Örneğin süreç tersinmez bir sıkıştırma ise $dV < 0$, $\mathcal{P}$ değeri ise P_{ext} değeri P den büyük olacağından pozitif olacaktır. Bu durumda da (2.18) den $\delta W_{lost} > 0$ olacaktır. Tersinmez bir genişleme de ise $dV > 0$ olurken, P_{ext} değeri P den küçük olacağı için $\mathcal{P} < 0$ olacaktır. Dolayısı ile yine (2.18) den $\delta W_{lost} > 0$ olacaktır.

Sonuç olarak her zaman $\delta W_{irr} > \delta W_{rev}$ olacaktır. Bu da şu manaya gelir ki bir hal değişiminde tersinir hal değişimi her zaman tersinmez olana göre daha verimli olacaktır. Bunun yanında şu gözleme de ulaşılabilir ki W_{lost} tersinmezliğin bir ölçüsü olarak görülebilir. W_{lost} değeri sürece bağlı bir değerdir ve sistem özelliklerinden

hesaplanamaz. Bu değer ancak ölçülebilir veya tahmin edilebilir.

Ancak incelediğimiz süreç tersinir bile olsa işin direkt olarak hesaplanması için süreç boyunca sistem basıncının hacme bağlı olarak nasıl değiştiğinin bilinmesi gerekir. Yani (2.16) entegralinin iç kısmını hesaplayabilmemizi sağlayacak süreç boyunca geçerli olan bir P(V) ifadesine ihtiyacımız vardır. Bu entegral ifadesi basınç hacim hal diyagramında alanı ifade eder. Bu alan yalnızca ilk ve son hale bağlı değildir bunların yanında bu alanın büyüklüğü takip edilen yola da bağlı olacaktır. Dolayısı ile iş bir tam diferansiyel değildir. Çünkü entegralinin büyüklüğü takip edilen sürece göre değişmektedir. Bir tam diferansiyelin entegrali ise her zaman yalnızca ilk ve son halin özelliklerine bağlıdır ve sürecin takip ettiği yoldan bağımsızdır.

Bu şekilde yola bağımlılık analizleri karmaşıklaştırır. Çünkü her farklı analizde P'nin V'ye bağlı değişimi yeniden tanımlanmalı ve çözüme bu şekilde gidilmelidir. Dolayısı ile yalnızca sistemin başlangıç ve son haline bağlı olarak değişen tam diferansiyellerin kullanılması hesaplamaları büyük ölçüde kolaylaştıracaktır.

2.1.2 Adyabatik İş

Şimdi içinde saf bir gaz taşıyan bir piston silindir düzeneği düşünüldüğünde. Silindir termal olarak yalıtılmış olsun. Buradaki gazı sistemimiz olarak alırsak ve bu sistemin başlangıçta ki sıcaklığına T_1, hacmine V_1, ve toplam içerdiği mol sayısına da N diyecek olursak bu sistemin bulunduğu termodinamik hali kesin bir şekilde ifade etmiş olunur. Sistem bu başlangıç şartlarında iken pistonun üzerine bir ağırlık yerleştirdiğimizi düşünüldüğünde bu ağırlık gaz sistemimizin üzerine bir P_{ext} dış basınç üretecektir ve bu basınç da sistemimizin hacminde bir değişime yol açacaktır. Sistem kapalı olarak tasarlandığı için sistemimize bulunan gaz mol miktarı N sabit kalacaktır. Ayrıca sistemin temas ettiği tüm duvarlar mükemmel şekilde yalıtıldığı içinde sistem tamamen adyabatik olarak kabul edilebilir. Sürecin sonunda sistem tam dengeye gelene kadar beklenildiğini varsayılsın ve bu durumda sıcaklığı ölçüldüğünde sıcaklığın bir miktar arttığı görülebilir. $T_n > T_1$. Burada gerçekleşen iş miktarı Denklem (2.16) (Çengel, 2003) den şu şekilde hesaplanabilir.

$$W_A = -\int_{V_1}^{V_n} P_{ext} dV = -P_{ext}(V_n - V_1) \qquad (2.19)$$

Bu sürecin gerçek bir tersinmez hal değişimi olduğunu düşünürsek rev yerine adyabatik süreci ifade eden A alt

indisini ve P yerine P_{ext} kullanmamız daha mantıklı olacaktır.

Şimdi benzer deneyi daha farklı bir biçimde ancak ilk ve son halleri korumak şartı ile tekrarlayalım ancak bu kez ağırlığın tamamını bir kere de piston üzerine yerleştirmektense yüklemeyi adım adım ve sistemin dengesini her adımda koruyacak şekilde yapalım. Bu ikinci deney de gerçekleştirilen iş miktarını ise şu şekilde ifade edilebilir.

$$W_B = -\int_{V_1}^{V_n} P_{ext} dV = -\sum_{i=1}^{n-1} P_{ext_i} (V_{i+1} - V_i) \tag{2.20}$$

Açıktır ki her iki sürecin sonunda ulaşılacak hacim miktarı aynı olmak zorundadır. Şu sonucu çıkartabiliriz.

$$W_A = W_B \tag{2.21}$$

Her ne kadar iki sürecin sonucunda elde edilen entegrasyon terimleri birbirinden farklılık gösterse de iki sürecin oluşturduğu eğrinin altında kalan alan aynıdır.

Daha sonra sistemin hainde ki aynı değişikliği farklı süreçler kullanarak sağlayabiliriz. Örneğin sistem bir elektrikli ısıtıcı ile ısıtılabilir veya silindirin içine yerleştirilecek bir pervane vasıtası ile sistem e dışarıdan bir

iş girdisi verilebilir. Hatta bu deneylerden bazıları tersinir bir sürece çok yaklaşabilir. Tüm bu deneylerde takip edilen süreç ne olursa olsun başlangıç ve son haller birbirinin aynı olduğu sürece aynı miktarda işin gerçekleştiğini görürüz. Bu da şu manaya gelir ki kapalı bir sistem üzerinde gerçekleştirilen adyabatik bir iş, sürecin takip ettiği yoldan bağımsızdır.

Daha önce bahsettiğimiz gibi sistemin bulunduğu yeri değiştirmek veya sistemin hızını değiştirmek sistemin sahip olduğu dış enerjiyi değiştirmek anlamına gelmektedir, benzer şekilde sisteme uygulanacak adyabatik bir iş de sistemin iç enerjisinin değişimi olarak görülebilir. Bu enerji U ile ifade edilir. Dolayısı ile şu eşitliğe ulaşılabilir. Denklem (2.22) (Çengel 2003)

$$U = W_{ad} \tag{2.22}$$

İç enerji ölçülemeyen ve yaygın bir özelliktir. Bu şekilde adlandırılır çünkü sistemin sıcaklık, basınç ve mol sayısı gibi özelliklerine bağlıdır.

2.2 Termodinamiğin Birinci Kanunu

Adyabatik ve adyabatik olmayan yollarla gerçekleştirilebilen bir hal değişimini düşünüldüğünde. Analizimizi adyabatik olmayan değişimleri de içerecek

şekilde genişletelim. Daha önce yaptığımız deneyi aynı başlangıç ve son halleri için ısıl yalıtkanlığı kaldırarak tekrar uygulayalım. Böylece sistem ve çevresi yalnızca kuvvet bakımından değil aynı zamanda ısıl bakımdan da etkileşim içinde olacaktır. Yine daha önce olduğu gibi toplamda aynı hal değişimini bize verecek küçük değişimler ile tersinir hal değişimine yaklaşan bir süreç takip edelim. Bu süreç sonucunda yapılması gereken iş miktarı Denklem (2.16) formülünden elde edilebildiği görülecektir ki bu iş miktarı her zaman adyabatik iş miktarından daha fazla olacaktır. Ancak adyabatik olmayan sürecin gerektirdiği iş miktarı takip edilen yola göre değişiklik gösterecektir. Buradan çıkarılacak sonuç adyabatik olmayan işin yola bağlı bir fonksiyon olduğudur.

2.3 Isı

Aynı başlangıç ve son halleri arasında gerçekleşen adyabatik ve adyabatik olmayan işler arasındaki farkın ısıl etki yolu ile transfer edilmiş olması gerekir. Bu enerji miktarını biz Q olarak adlandırırız ve bu miktar sürece bağlı ölçülebilir bir büyüklüktür.

$$Q \equiv W_{ad} - W_{nad} \tag{2.23}$$

Her ne kadar bu ifade tarzı sık rastlanılan bir tarz olmasa da aslında ısı transferi ile ilgili bütün normal

nitelikleri taşımaktadır. Ayrıca bu tarz genel olarak yararlı olan kesin bir nicelik ilişkisi sağlar.

Denklem (2.23) iki farklı hal arasında gerçekleşen farklı iş süreçlerinden yola çıkarak Q miktarının belirlenmesine imkân sağlar. Belirli bir süreçte iki hal arasındaki gerekli iş miktarı ile aynı haller arasındaki adyabatik iş miktarı karşılaştırılarak ısıya dönüşen miktar tespit edilebilir. Eğer tersinir ısı transferi için bir değer istersek adyabatik olmayan sürecin tersinir olması gerekir. Ancak adyabatik iş değeri tersinirlikten bağımsızdır. Eğer sistemin iki hali arasındaki adyabatik iş miktarını hesaplamak istersek 1 den 2 ye olan adyabatik iş miktarı yerine (W_{12}) 2 den 1 e olan adyabatik iş miktarını (W_{21}) hesaplamak yeterlidir. Çünkü adyabatik bir hal değişimi için

$$W_{12} = -W_{21} \tag{2.24}$$

Burada dikkat edilmesi gereken Denklem (2.23) denkleminin herhangi bir sıcaklık ifadesi taşımamasıdır. Sıcaklık ısı yoğunluğunun ifadesidir ve ısının bir ölçüsüdür. Sıcaklık ve ısı ısıl etkileşimin sonucunda bir ısı transferinin oluşması durumunda birbiri ile ilişkilidir. Yine iş de olduğu gibi ısı içinde bir işaret prensibi belirlemek gerekmektedir.

$Q > 0$, eğer ısı çevreden sisteme aktarılıyorsa ve

$Q < 0$, ısı sistemden çevreye aktarılıyorsa

Adyabatik olmayan iş W_{nad} yola bağımlı bir değişken olduğu için ve Q adyabatik olmayan ve olan işin bir lineer birleşimi olduğu için Q nun da yola bağımlı bir fonksiyon olması gerekir.

2.3.1 Kapalı Sistemler için Birinci Yasa

Buraya kadar ulaştığımız sonuçları kapalı sistemler için birinci yasanın ifadesine ulaşmak için kullanabiliriz. Diferansiyel bakış açısı ile birinci yasayı iki parça şeklinde ifade edilebilir.

$$dU = \delta W_{ad} \tag{2.25}$$

Birinci kısımda, adyabatik iş, iç enerji değişimi ile ilişkilendirilebilir.

Daha sonra bu ifadeye denklem (2.23) eklenirse sonuçta birinci yasanın diferansiyel ifadesine ulaşılabilir.

$$dU = \delta Q + \delta W \tag{2.26}$$

Bu iki denklem kapalı bir sisteme etkiyen diferansiyel bir süreç için birinci yasadır ve her zaman doğrudur. Bir hal

değişimi hem tersinir hem de tersinmez iki farklı yol ile gerçekleştirilebiliyorsa Denklem (2.26)'dan yola çıkarak

$$dU = \delta Q + \delta W = \delta Q_{rev} + \delta W_{rev} \tag{2.27}$$

$\delta Q \neq \delta Q_{rev}$ ve $\delta W \neq \delta W_{rev}$ olmasına rağmen toplamları birbirine eşit olacaktır. Buda gösterir ki tersinir bir süreç için hesaplanacak dU miktarı diğer süreçleri içinde aynı olacaktır. (2.27) ifadesi tüm süreç için entegre edilirse

$$\Delta Q = Q + W \tag{2.28}$$

Eşitliği elde edilir. Eşitlikten anlaşılabileceği gibi şayet sistem sınırlarından herhangi bir iş geçişi gerçekleşmiyor ise hal değişimi için gerek ısı miktarı doğrudan iç enerji değişimine eşit olacaktır.

$$\Delta U_{wf} = Q_{wf} \tag{2.29}$$

Durum fonksiyonu olan U'nun mutlak değerini belirlemek mümkün değildir. Ancak ΔU'nun değişimi hesaplanarak iç enerji ile ilgili bir fikir sahibi olunabilir. Dolayısı ile mantıklı olan ΔU'nun hesaplanabilmesi için bir referans noktasının belirlenmesidir. Bu noktada iç enerji değeri 0 olarak kabul edilir ve diğer tüm noktalar bu noktaya göre belirlenir. Dolayısı ile genel ΔU farkı için şu ifade yazılabilir.

$$\Delta U = (U_2 - U_{ref}) - (U_1 - U_{ref}) = (U_2 - U_1) \qquad (2.30)$$

2.4 Termodinamiğin İkinci Yasası

Amacımız termodinamikte kullandığımız yola bağlı fonksiyonları hal fonksiyonları ile ilişkilendirmekti. Birinci yasa bu manada bize iş ve ısı gibi iki önemli etkiyi iç enerji gibi bir tam diferansiyele bağlama imkânı sundu. Ancak birinci yasa bunu yapmanın tek yolu değildir. Şimdi termodinamik süreçlere bir başka açıdan bakmaya çalışalım. Hacim değişimi ile gerçekleşen bir tersinir işe bir bakalım.

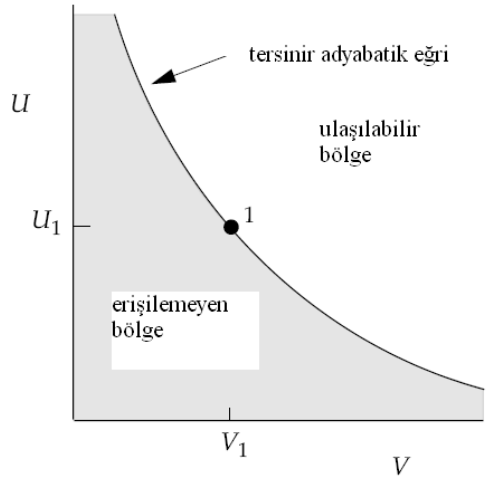
$$\frac{\delta W_{rev}}{P} = -dV \qquad (2.31)$$

Burada her iki tarafı P ile böldüğümüzde ifadenin sol tarafı bir tam diferansiyel olan *dV*'ye eşit olacaktır. Benzer şekilde yola bağımlı bir fonksiyon olan δQ de benzer bir şekilde bir tam diferansiyel haline dönüştürülebilir. Burada yine bir λ faktörü kullanılarak tam diferansiyel hal fonksiyonu elde etmek mümkündür.

$$\frac{\delta Q_{rev}}{\lambda} = dS \qquad (2.32)$$

Şimdi daha önce bahsedilenlere benzer ısıl olarak mükemmel şekilde yalıtılmış bir sistem düşünüldüğünde yine bu sistem yalnızca adyabatik bir iş yapabilir olsun.

Böyle bir sistemin bulunduğu hali tanımlayabilmemiz için yalnızca iki özelliğini tanımlamamız yeterlidir. Bunları V ve U olarak seçebiliriz. Böyle bir sistem tersinir hal değişimleri sırasında Şekil 2.1 de görüldüğü gibi belirli bir eğri takip eder.



Şekil 2-1: Tersinir Adyabatik Eğri

Bu eğri negatif eğime sahiptir.

$$\left(\frac{\partial U}{\partial V}\right)_{Q=0} < 0 \qquad (2.33)$$

Sistem üzerinde gerçekleşecek herhangi bir adyabatik hal değişimi bu eğri üzerinde kalacaktır. Eğer sistem üzerinde tersinmez adyabatik bir iş yapılır ise bu sefer diyagramda eğrinin üst kısmında kalan bölgeye geçilir. Bunu şu şekilde açıklayabiliriz. Eğer adyabatik süreç tersinmez ise yapılan işin bir kısmı tersinmezlikte kaybolacaktır bu durumda örneğin süreci bir sıkıştırma süreci ise hacim tersinir iş sonucu ulaşılacak hacimden fazla olur. Bu durumda nokta olması gerektiğinden daha solda olur ki buda noktayı eğrinin üstünde kalan alana taşır. Yine eğer tersinir işle ulaşılacak hacme tersinmez bir süreçle ulaşılmaya çalışılırsa bu kez de sisteme daha fazla iş yapılması gerekir. Bu durumda da aynı iş miktarı ile bir adyabatik sürecin sebep olacağı iç enerji değişimine ulaşmamız gerekir ki buda noktayı diyagramda eğrinin üst kısmına doğru öteler. Ancak adyabatik bir süreçte hiçbir şekilde yapılan işin eğrinin altında kalan kısma geçmesine imkân yoktur. Eğrinin altında kalan alana ulaşmak için ise adyabatik olmayan sürecin yanında sisteme veya sistemden ısı transferi gerçekleştirilmesi gerekir.

Adyabatik süreç tarafından ulaşılamayacak hallerin varlığı Caratheodory tarafından yapılan çalışmada δQ_{rev}'yi tam diferansiyele dönüştürecek bir entegral çarpanın varlığı için yeterli görülmektedir. Deneysel olarak termodinamik bir hali belirtmek için yalnızca iki bağımsız özelliği tanımlamak yeterli olmaktadır. Dolayısı ile örnek verilen

sistem için δQ_{rev} iki değişkenin cinsinden ifade edilebilir ve entegral faktörünü seçilebilir. En basit çözüm entegral faktörünü pozitif mutlak termodinamik sıcaklık olarak almak olacaktır. Böylece (2.32) denklemi şu şekli alır.

$$dS \equiv \frac{\delta Q_{rev}}{T} \tag{2.34}$$

Bu yeni S fonksiyonu da entropi olarak adlandırılır. Bu özellik de sistemin yaygın özelliklerindendir ve tıpkı U gibi kavramsaldır.

2.4.1 Kapalı Sistemler için İkinci Yasa

Birinci yasaya benzer olarak ikinci yasayı da iki parça halinde ifade edilebilir. İlki yola bağlı Q fonksiyonunu tam diferansiyele dönüştüren ve yeni bir durum fonksiyonu oluşturan ifadedir.

İkinci kısımsa kapalı bir sistemde ki adyabatik bir süreç için mümkün olan ilerleme doğrultusunu vermektedir.

$dS > 0$, tersinmez adyabatik süreç

$dS = 0$, tersinir adyabatik hal değişimi,

$dS < 0$, adyabatik olarak gerçekleşmesi mümkün olmayan hal değişimi.

2.5 Temel Denklemler

Daha önceki bölümlerde ısı ve iş gibi süreç değişkenlerinin S ve U gibi sistem özellikleri ile nasıl ilişkilendirilebilinecekleri görülebilir bu ilişkiler birinci ve ikinci yasa ile elde edilebilir. Şimdi U ve S in nasıl hesaplandığı incelenebilir. Çünkü U ve S direkt olarak ölçülemez ilk olarak ΔU ve ΔS'i nasıl ölçülebilir sistem özellikleri ile ilişkilendirilebilir ona bakalım.

2.5.1 Kapalı ve Açık Sistemler için Temel Denklemler

Açık sistemler için temel denklemlere geçmeden önce kapalı sistemler için denklemlerin incelenmesi faydalı olacaktır.

Örnek olarak homojen bir sistemi ele alalım bu sistem sınırlarından hacim değişimi şeklinde iş giriş ve çıkışına müsaade edilsin ve aynı zamanda sınırlarından ısı geçişi de mümkün olsun bu durumda sistem diferansiyel hal değişimine imkân verecek şekilde tasarlanmış olur. Sistemin özelliği olan U sisteme uygulanan işin tersinir olmasından

etkilenmeksizin aynı miktarda değiştiği varsayılsın. Bu durumda aynı U değişim miktarı için gerekiyorsa daha fazla iş miktarı gerçekleştirilerek aynı hal değişimi, sisteme uygulanabilir.

$$dU = dU_{rev} = dU_{irr} \tag{2.35}$$

Burada ilk yasayı uygulayarak şu sonuca ulaşılabilir.

$$dU = \delta Q_{rev} + \delta W_{rev} = \delta Q_{irr} + \delta W_{irr} \tag{2.36}$$

Tersinir hal değişimi için şu ifadeler yazılabilir. $\delta Q_{rev} = TdS$ ve $\delta W_{rev} = -PdV$ böylece (2.36) denklemi şu hali alır.

$$dU = TdS - PdV = \delta Q_{irr} + \delta W_{irr} \tag{2.37}$$

veya başka bir ifade ile

$$dU = TdS - PdV \tag{2.38}$$

Bu ifade kapalı sistemler için temel denklemdir. Bu denklem tersinir veya tersinmez herhangi bir sürece uygulanabilir. Ancak tersinmez bir süreçte $\delta Q_{irr} < (TdS)$ ve $\delta W_{irr} < (-PdV)$ olduğu için gerçekte yapılan iş ve transfer edilen ısı miktarları bilinmeyen bir yolla ve tersinmezliğin derecesine göre (TdS) ve $(-PdV)$ ifadelerine dağıtılır. Bu

durumda iç enerjiyi U iki birbirinden bağımsız değişken ile bir fonksiyon haline dönüştürülebilir. Bu iki değişkeni S ve V olarak seçersek.

$$T = \left(\frac{\partial U}{\partial S}\right)_{VN} \quad \text{ve} \quad -P = \left(\frac{\partial U}{\partial V}\right)_{SN}$$ bu fonksiyonun bağımsız değişkenleri cinsinden türevleri olacaktır. Ancak mühendislik uygulamaları için S uygun bir değişken değildir çünkü ölçerek bu değişkenin değerini belirli bir hal için tespit etmek mümkün değildir. Ancak S ve V'nin örneğin daha uygun gözüken T ve P ile değiştirilmesi ΔU 'nun hesaplanmasını daha karmaşık bir hale getirecektir. Bu yüzden denklemin kullanışlığını koruyarak S ve V değişkenlerini değiştirmenin bir yolu olan Legendre dönüşümlerini kullanmak uygun olacaktır. Böylece durum fonksiyonlarına ulaşma imkânı doğar. Bu şekilde elde edilecek fonksiyonlar bundan sonraki bütün termodinamik hesaplama ve modellemelere temel oluşturacaktır.

İlk dönüşümde V yi laboratuar ortamında kontrollü ve ölçümü daha kolay olan P ile değiştirmeye çalışalım. Dönüşüm sonucu

$$H = U - (-PV) = U + PV \tag{2.39}$$

olacaktır.

Tanımlanan bu yeni değişken entalpi olarak isimlendirilir. Yaygın kavramsal bir durum fonksiyonudur. Diferansiyel formda ise şu hali alır.

$$dH = TdS + VdP \tag{2.40}$$

Bir başka temel denkleme de S'i T ile yer değiştirerek ulaşılabilir. Dönüşüm sonucu ulaşılacak yeni fonksiyon şöyle olur.

$$A = U - TS \tag{2.41}$$

Bu fonksiyonda Helmholtz enerjisi adı ile anılır ve diferansiyel ifadesi şu şekildedir.

$$dA = -SdT - PdV \tag{2.42}$$

Dördüncü temel denklem ise Legendre dönüşümünün iki kez uygulanması ile elde edilir.

$$G = U - (-PV) - TS = H - TS \tag{2.43}$$

Yine bu ifadede Gibbs enerjisi olarak adlandırılan yeni bir kavramsal ve yaygın sistem özelliğine karşılık gelir. Ancak çok özel haller de elde ettiğimiz bu son iki özellik A ve G bir fiziksel anlam taşırlar ancak bu durumlarda da çok kullanışlı bir rol üstlenirler.

Bütün bu denklemleri özetleyecek olursak

$$dU = TdS - PdV \tag{2.44}$$

$$dH = TdS + VdP \tag{2.45}$$

$$dA = -SdT - PdV \tag{2.46}$$

$$dG = -SdT + VdP \tag{2.47}$$

Denklemlerine kapalı sistemler için ulaşılmış olur. Bu denklemlerin her biri aslında aynı eşitliğin farklı ifadeleridir ve her zaman da geçerlidirler. Elimizde olan probleme göre bu denklemlerin herhangi biri seçilerek çözüm kolaylaştırılabilir.

Şimdi bizim için daha önemli olan açık sistemler için bu temel denklemler geliştirilmeye çalışılabilir. Bir açık sistem çevresi ile yalnızca ısı ve iş değiş tokuşuna değil aynı zamanda kütle giriş çıkışına da maruz kalır. Bu durumda sistem herhangi bir sayıda (Pitzer, 1957…) bileşen içerebilir ve bunların her biri kendilerine ait belirli bir mol miktarına sahip olabilirler {N_1, N_2, N_3,….}. Açık sistemlerde kapalı sistemlerden farklı olarak iç enerji sadece S ve V ye değil aynı zamanda sistemdeki her bir bileşenin mol miktarına bağlıdır. Dolayısı ile açık sistemler için iç enerji fonksiyonu şu şekilde olacaktır.

$$U = U(S, V, N_1, N_2, N_3, ...) \tag{2.48}$$

U bir durum fonksiyonu olduğu için toplam diferansiyeli şu şekilde yazabiliriz.

$$dU = \left(\frac{\partial U}{\partial S}\right)_{VN} dS + \left(\frac{\partial U}{\partial V}\right)_{SN} dV + \sum_i \left(\frac{\partial U}{\partial N_i}\right)_{SVN_{j\neq i}} dN_i \tag{2.49}$$

Burada $N = \sum N_i$ toplam mol sayısını ifade etmektedir. Diğer bağıntıları kullanarak

$$dU = TdS - PdV + \sum_i \left(\frac{\partial U}{\partial N_i}\right)_{SVN_{j\neq i}} dN_i \tag{2.50}$$

formülünü elde edilebilir. Bu denklem açık sistemler için ilk temel denklemdir. Tersinir bir süreçte bu denklemdeki her bir terim basit anlamlarını taşırlar örneğin (TdS) sistem sınırlarından geçen ısı miktarını, (-PdV) sistem hacminde değişiklik yaratan iş miktarını, $\left(\frac{\partial U}{\partial N_i}\right)_{SVN_{j\neq i}} dN_i$ ise *i* bileşenin sistem sınırlarına geçişte sisteme eklediği enerji miktarını vermektedir. Ancak tersinmez bir süreç için bu tür bir yorum yapmak mümkün değildir. Ancak bu denklem tıpkı kapalı sistemlerde olduğu gibi sürecin tersinir veya

tersinmez olmasına bakmaksızın geçerlidir. Tersinmezlik durumunda yalnızca tam diferansiyeli oluşturan bileşenlerin değeri değişiklik gösterecektir. Denklem (2.50) dan yola çıkarak diğer temel denklemleri de elde edilebilir. (Pitzer, 1957…)

$$dH = TdS + VdP + \sum_i \left(\frac{\partial H}{\partial N_i} \right)_{SPN_{j \neq i}} dN_i \tag{2.51}$$

$$dA = -SdT - PdV + \sum_i \left(\frac{\partial A}{\partial N_i} \right)_{TVN_{j \neq i}} dN_i \tag{2.52}$$

$$dG = -SdT + VdP + \sum_i \left(\frac{\partial G}{\partial N_i} \right)_{TPN_{j \neq i}} dN_i \tag{2.53}$$

Burada her denklemin son terimi aslında aynı şeyi ifade etmektedir ve bunlara ortak bir sembol atamak uygun olacaktır

$$\begin{aligned} \left(\frac{\partial U}{\partial N_i} \right)_{SVN_{j \neq i}} &= \left(\frac{\partial H}{\partial N_i} \right)_{SPN_{j \neq i}} \\ = \left(\frac{\partial A}{\partial N_i} \right)_{TVN_{j \neq i}} &= \left(\frac{\partial G}{\partial N_i} \right)_{TPN_{j \neq i}} \end{aligned} \tag{2.54}$$

Bu ortak sembol kimyasal potansiyel olarak adlandırılır ve bazı kaynaklarda $\overline{G}_i$ bazı kaynaklarda ise μ_i olarak ifade edilir.

Kimyasal potansiyel kavramsal bir özelliktir. Fiziksel anlamı i bileşeninden sisteme bir mol ekleyebilmek için yapılması gereken tersinir iş miktarı anlamına gelir. Bu ekleme yapılırken diferansiyel terimden de anlaşılacağı gibi sıcaklık, basınç ve diğer bileşenlerin mol sayıları sabit tutulacaktır. Sonuç olarak bir açık sistem için μ_i kimyasal potansiyel olmak üzere temel denklemler

$$dU = TdS - PdV + \sum_i \mu_i dN_i$$
$$dH = TdS + VdP + \sum_i \mu_i dN_i$$
$$dA = -SdT - PdV + \sum_i \mu_i dN_i$$
$$dG = -SdT + VdP + \sum_i \mu_i dN_i$$

olacaktır.

Isıl sisteminin modellenmesinin doğru bir şekilde yapılabilmesi için sistemde bulunan akışkanların özelliklerinin mümkün olan en iyi şekilde hesaplanabilmesi gerekir. Gerçek gaz ve sıvı akışkanların özeliklerinin belirlenebilmesi için ilk adım ideal gazların davranışının belirlenmesidir.

3 AKIŞKANLARIN ÖZELLİKLERİNİN BELİRLENMESİ

3.1 İdeal Gazların Özelliklerinin Belirlenmesi

İdeal gaz üç farklı karakteristik özelliğe sahiptir. (a) Bir ideal gazın molekülleri arasında kuvvet etkileşimi gerçekleşmez dolayısı ile molekülleri arasında potansiyel enerji bulunmaz. (b) İdeal gazın molekül ve atomları sürekli hareket halindedirler. Dolayısı ile bir ideal gazın kinetik enerjisi ve sıcaklığı vardır. (c) İdeal gazın molekülleri gazı çevreleyen kabın duvarları ile momentum transferi yapar. Böylece ideal gaz basınca ve belirli miktarda bir hacme sahiptir. Bu üç karakteristik özellikleri ideal gazların davranışlarını belirler. Moleküller arasında kuvvet etkileşiminin bulunmaması bir ideal gazın teorik olarak sıfır hacme düşürüldüğünde faz değişiminin gerçekleşmemesine neden olur. Moleküller arasında kuvvet etkileşimimin bulunmaması sebebi ile sıvı ve katı fazın oluşmasını sağlayan kuvvetler ideal gazlarda görülmez. Bu durumda bir ideal gaz her durumda gaz fazında kalır. Moleküller arasında kuvvetlerin bulunmadığının varsayılması büyük bir hataya yol açar ancak ideal gaz yaklaşımının kullanılmasının amacı gerçek gaz davranışına bir yaklaşım yapmak değil, gerçek gazların özelliklerinin bulunabilmesinde ideal gaz özelliklerinin bir temel oluşturmasıdır.

3.2 İdeal Gaz Hal Denklemi

İdeal gaz hal denklemi şu şeklidedir.

$$PV = NR_u T \tag{3.1}$$

P :basınç, Pa

V :hacim, m^3

N :mol sayısı, mol

T :sıcaklık, K

Bu birimler kullanıldığında R_u değeri 8.314 $Pa.m^3/(mol.K)$ olacaktır. Ancak denklemi farklı birimlerle de ifade edilebilir. Termodinamik analizlerde en yaygın kullanılan birimler P (kPa) ve N (kmol) dür. Bu birimler kullanıldığında yine R_u değeri aynı olacaktır.

Bu denklem deneysel olarak elde edilen ve Boyle Charles bağıntısı olarak adlandırılan PvT ilişkisinden ortaya çıkar. Ayrıca moleküller arasındaki kuvvetlerin göz ardı edilmesi durumunda aynı eşitlik istatistiksel mekanikten de çıkartılabilir. Burada kullanılan V sistemdeki toplam hacim miktarını belirtmektedir. N ise yine sistemde ulunan toplam gaz mol miktarını ifade eder. Ancak birçok açıdan denklemi

bu şekilde kullanmak pratik olmamaktadır. Çünkü belirli bir sistem için bu denklemin daha genel bir şekilde mol sayısına bağlı olmaksızın ifade edilmesi sistem üzerinde hesaplamalar yapmak için daha uygun olmaktadır. Bu yüzden $\overline{v} = \frac{V}{N}$ eşitliğini kullanarak denklemi yeniden yazmak daha uygun olacaktır.

$$P\overline{v} = R_u T \tag{3.2}$$

Denklem (3.2) bizim için daha kullanışlı olsa da biz analizlerin çoğunda mol değerlerinden çok sistemin kütlesini kullanacağız. Dolayısı ile ideal gaz denklemini de kütle cinsinden ifade etmek daha iyi olacaktır. Bu durumda R_u yerine $R \equiv \frac{R_u}{M}$ ve $v = \frac{\overline{v}}{M}$ ifadesini kullanabiliriz.

$$Pv = RT \tag{3.3}$$

Bu durumda yeni birimler şu şekilde olacaktır.

P :basınç, kPa

v :hacim, m^3/kg

T :sıcaklık, K

R :gaz sabiti, $kPa.m^3/(kg.K)$

Yine bu denklem bazı durumlarda farklı şekillerde de ifade edilebilir. Bunlardan en yaygın olanları ve bu çalışmada kullanılanları aşağıda özetlenmiştir. (Pitzer, 1957…)

$$P = \rho RT \tag{3.4}$$

$$PV = mRT \tag{3.5}$$

Bunlar içinde en sık karşılaşacağımız ifade şekli Denklem (3.4) dür. Burada da ρ kg/m^3 ve R yine yukarıda verildiği gibi olacaktır.

İdeal gaz denklemi bize bir gazın *PvT* davranışı hakkında en genel bilgiyi verir ancak bir gerçek gaz çok sınırlı koşullar altında ideal gaz davranışı sergiler. Bir gaz ancak çok düşük basınçta veya kritik basıncın üstünde yüksek sıcaklıkta ideal gaz davranışı sergiler. Özellikle kritik nokta civarında bir gazın davranışı ideal gaz davranışından büyük farklılıklar gösterir.

Gerçek gaz davranışını daha iyi modelleyen denklemler geliştirilmiştir ve bu çalışmada geliştirilen bilgisayar programı bu denklemlerden bazılarını kullanmaktadır. Ancak bu denklemlere ileride değinilecektir. Şimdi ideal gazların diğer termodinamik özelliklerinin nasıl hesaplandığını incelenebilir.

3.3 İdeal Gaz Termodinamik Özellikleri

Daha önceki bölümde geliştirdiğimiz genel termodinamik özellik bağıntılarını (2.38)-(2.42) kullanarak ideal gazların bu özelliklerini tespit edilebilir. İlk belirleyeceğimiz özellik iç enerji (2.38) denklemi ile ifade edilmiştir. $dU = TdS - PdV$ denklemi kg başına ifade edilirse.

$$du = Tds - Pdv \tag{3.6}$$

olur. Şimdi daha önce geliştirdiğimiz denklemlere bir yensini ekleyebiliriz. İç enerji u fonksiyonu sistemin bir halini temsil ettiğini düşünürsek sistemin o halini temsil edebilecek diğer parametreler cinsinden iç enerjiyi tanımlamanın mümkün olduğu sonucuna ulaşılabilir. Burada bizi sonuca götürecek u'yu sıcaklık T ve özgül hacim v cinsinden ifade etmektir.

$$du = \left(\frac{\partial u}{\partial T}\right) dT + \left(\frac{\partial u}{\partial v}\right) dv \tag{3.7}$$

Denklem (3.7) de ki $\left(\frac{\partial u}{\partial T}\right)$ terimi termodinamik de sabit hacimde ki ısı kapasitesi olarak ifade edilir.

$$c_v = \left(\frac{\partial u}{\partial T} \right)_v \tag{3.8}$$

$$du = c_v dT + \left(\frac{\partial u}{\partial v} \right) dv \tag{3.9}$$

Isı kapasitesi bize denklem (3.8) den de anlaşılacağı gibi bize gazın sabit hacim altında birim sıcaklık değişiminde gerçekleşecek iç enerji değişimidir. Bu değerin deney ile tespit edilmesi mümkündür. Denklem (3.6) dan *ds* ifadesini çekecek olursak,

$$ds = \frac{du}{T} + Pdv \tag{3.10}$$

elde edilir. Denklem (3.9) (3.10) da yerine konulacak olursa.

$$ds = \frac{c_v}{T} dT + \frac{1}{T} \left[\left(\frac{\partial u}{\partial v} \right) + P \right] dv \tag{3.11}$$

ds için Denklem (3.11)'e ulaşılır. Ancak $\left(\frac{\partial u}{\partial v} \right)$ terimi de kolayca hesaplanabilecek bir terim değildir. Bunun için de yine *du* da olduğu gibi *ds* için sıcaklık *T* ve özgül hacim değişkenlerine bağlı bir diferansiyel ifade yazılabilir.

$$ds = \left(\frac{\partial s}{\partial T}\right) dT + \left(\frac{\partial s}{\partial v}\right) dv \tag{3.12}$$

Burada $\left(\frac{\partial s}{\partial T}\right) = \frac{c_v}{T}$ olacaktır. Aynı şekilde $\frac{1}{T}\left[\left(\frac{\partial u}{\partial v}\right) + P\right] = \left(\frac{\partial s}{\partial v}\right)$ olmalıdır. Eşitliğin sağ tarafı da hesaplanabilir bir değer gibi durmasa da Maxwell eşitlikleri sayesinde $\left(\frac{\partial s}{\partial v}\right)$'yi hesaplanabilir bir değere dönüştürülebilir. Maxwell eşitliklerine göre,

$$\left(\frac{\partial s}{\partial v}\right)_T = \left(\frac{\partial P}{\partial T}\right)_V \tag{3.13}$$

dir ve Denklem (3.13)'ü Denklem (3.11) de yerine yazacak olursak *ds* için şu denkleme ulaşılabilir.

$$ds = \frac{c_v}{T} dT + \left(\frac{\partial P}{\partial T}\right)_V dv \tag{3.14}$$

Benzer şekilde *ds*'e *du*'dan değil de *dh* dan yola çıkarak hesaplarsak bu kez Denklem (2.40)'ı kullanmamız gerekir.

$$dh = c_p dT + \left(\frac{\partial h}{\partial v}\right) dv \tag{3.15}$$

Buradaki tek fark $c_p = \left(\frac{\partial h}{\partial T}\right)_p$ olacaktır. c_p ise sabit basınç altında sıcaklıktaki birim değişikliğe göre entalpi de ki değişiklik miktarını veren sabit basınç ısı kapasitesidir. Bu kez *dh* denkleminden *ds* şu şekilde elde edilir.

$$ds = \frac{c_p}{T} dT - \left(\frac{\partial v}{\partial T}\right)_P dP \tag{3.16}$$

Entropi ifadesinden yola çıkarak *du* içinde hesaplanabilir bir ifade geliştirme imkânımız doğar. Eğer denklem (3.14)'ü tekrar denklem (3.6) da yerine koyacak olursak,

$$du = c_v dT + \left[T\left(\frac{\partial P}{\partial T}\right)_V - P \right] dv \tag{3.17}$$

Aynı şekilde Denklem (3.16) dan yola çıkarak da *dh* için benzer bir ifade geliştirebiliriz.

$$dh = c_p dT + \left[v - T\left(\frac{\partial v}{\partial T}\right)_P \right] dP \tag{3.18}$$

Bu noktadan sonra *u* ve *h* farklarını hesaplayabiliriz. Örneğin 1 ve 2 noktası arasındaki *u* ve *h* farkını hesaplayabiliriz. (3.14),(3.16),(3.17) ve (3.18) denklemlerini 1 den 2 'ye entegre edilirse

$$(s_2 - s_1) = \int_1^2 \frac{c_v}{T} dT + \int_1^2 \left(\frac{\partial P}{\partial T} \right)_V dv \tag{3.19}$$

$$(s_2 - s_1) = \int_1^2 \frac{c_p}{T} dT - \int_1^2 \left(\frac{\partial v}{\partial T} \right)_P dP \tag{3.20}$$

$$(h_2 - h_1) = \int_1^2 c_p dT + \int_1^2 \left[v - T \left(\frac{\partial v}{\partial T} \right)_P \right] dP \tag{3.21}$$

Ayrıca ideal bir gazın *g* (Gibbs) ve *a* (Helhmotz) değerlerini de bu eşitliklerden yol çıkarak hesaplayabiliriz. Denklem (2.42)'ı ekler ve çıkarırsak

$$dg = (vdP + Tds) - sdT - Tds \tag{3.22}$$

parantez içerisinde ki ifade bize *dh*'ı verecektir.

$$dg = dh - sdT - Tds \tag{3.23}$$

Buradan da $dg = dh - d(Ts)$ 'e ulaşılabilir ki bu durumda

$$g \equiv h - Ts \tag{3.24}$$

olacaktır. Benzer şekilde Helhmotz da

$$a \equiv u - Ts \tag{3.25}$$

olacaktır. Şimdi ideal gaz hal denkleminden yola çıkarak (3.19)-(3.21), (3.24) ve (3.25) denklemleri ideal gazlar için elde edilebilir.

$$(u_1 - u_{ref}) = \int_{T_{ref}}^{T_1} c_v dT + \int_{v_{ref}}^{v_1} \left[\frac{TR}{v} - P \right] dv$$

$$(u_1 - u_{ref}) = \int_{T_{ref}}^{T_1} c_v dT \tag{3.26}$$

$$(h_1 - h_{ref}) = \int_{T_{ref}}^{T_1} c_p dT + \int_{P_{ref}}^{P_1} \left[v - \frac{TR}{P} \right] dP$$

$$(h_1 - h_{ref}) = \int_{T_{ref}}^{T_1} c_p dT \tag{3.27}$$

$$(s_1 - s_{ref}) = \int_{T_{ref}}^{T_1} \frac{c_v}{T} dT + \int_{v_{ref}}^{v_1} \frac{R}{v} dv$$

$$(s_1 - s_{ref}) = \int_{T_{ref}}^{T_1} \frac{c_v}{T} dT + R \ln \frac{v_1}{v_{ref}} \tag{3.28}$$

veya c_p den yola çıkarak,

$$(s_1 - s_{ref}) = \int_{T_{ref}}^{T_1} \frac{c_p}{T} dT + R \ln \frac{P_1}{P_{ref}} \tag{3.29}$$

$$(g_1 - g_{ref}) = (h_1 - h_{ref}) - T(s_1 - s_{ref}) \tag{3.30}$$

$$(a_1 - a_{ref}) = (u_1 - u_{ref}) - T(s_1 - s_{ref}) \tag{3.31}$$

3.4 Gerçek Gazların Özellikleri

3.4.1 Gerçek Gazların Hal Denklemleri

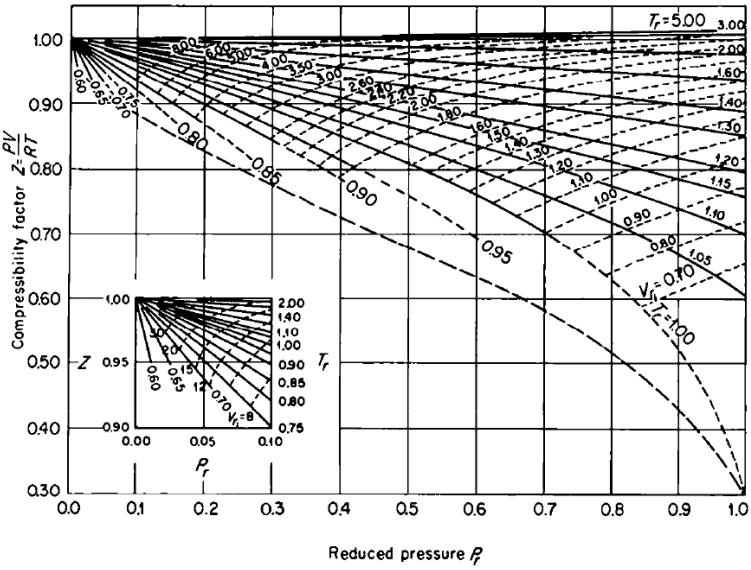
Bir gazın ideal gaz halinden sapması Z sıkıştırılabilirlik faktörü ile ifade edilir. Bu değer

$$Z = \frac{PV}{RT} \tag{3.32}$$

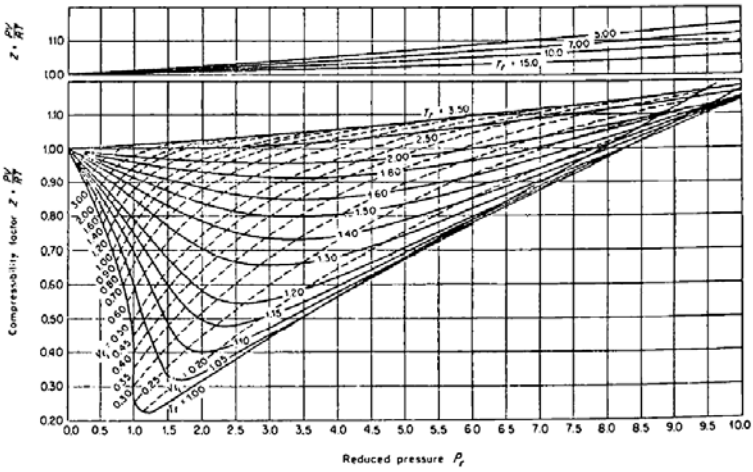
Sıkıştırılabilirlik faktörü genellikle indirenmiş basınç ve indirgenmiş sıcaklık ile ifade edilir. İndirgenmiş basınç $P_r = P/P_c$ ve indirgenmiş sıcaklık ise $T_r = T/T_c$ ile ifade edilir. Burada sıcaklıklar K cinsindendir ve T gazın sıcaklığı T_c ise gazın kritik sıcaklığını ifade etmektedir. Basınç için içinde yine aynı şekilde P_c kritik basıncı ifade etmektedir basınç değerleri için aynı olmaları dışında birim önem arz etmez.

$$Z = f(T_r, P_r) \tag{3.33}$$

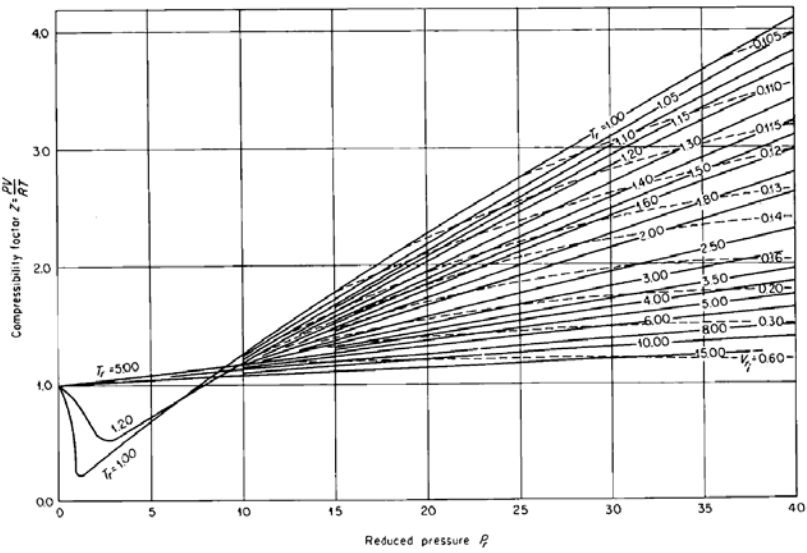
Burada f fonksiyonu deneysel olarak PvT değerlerinden Nelson ve Obert tarafından elde edilmiştir (Prausnitz, 1987). Elde edilen bu eğriler şu şekildedir.



Şekil 3-1: (0-1.0 P_r) Genelleştirilmiş sıkıştırılabilirlik faktörü (Prausnitz, 1987)



Şekil 3-2: (0-100 P_r) Genelleştirilmiş Sıkıştırılabilirlik faktörü (Prausnitz, 1987)



Şekil 3-3: (0-40 P_r) Genelleştirilmiş Sıkıştırılabilirlik Faktörü

Ancak bu grafiklerde

$$V_r = \frac{V}{RT_c / P_c} \tag{3.34}$$

dir. Denklem (3.33) gerçek gazlar için hal denklemine bir örnektir. Bu denklemin önerdiği yaklaşım aynı T_r ve P_r değerleri için bütün gazların aynı sıkıştırılabilirlik faktörüne

sahip olduklarıdır. Buna benzer denklemlere göre eğer hal denklemleri indirgenmiş basınç ve sıcaklık cinsinden yazılırlarsa bütün gazlar için ortak olacaklardır. Bu yaklaşım kısmen doğrudur (Prausnitz, 1987). Yine bu yaklaşıma göre

$$V_r = \frac{V}{V_c} = \frac{(Z / Z_c)(T / T_c)}{P / P_c} = f_1(T_r, P_r) \tag{3.35}$$

Veya

$$Z = Z_c f_2(T_r, P_r) \tag{3.36}$$

Tek atomlu gazlar büyük oranda polar olan sıvılar ve büyük moleküllerden oluşan sıvılar hariç organik bileşikler de Z_c 0.27 ile 0.29 arasında değişir. Eğer bu değeri sabit varsayarsak Denklem (3.36)'i Denklem (3.33)'e dönüşür. Ancak Z_c de kullanılarak ileride gerçek gaz davranışını daha iyi temsil eden denklemler (3.36) den çıkartılabilir.

T_c ve P_c basınç ve sıcaklık parametrelerini boyutsuz hale getirmek için kullanılmaktadır. Bunların dışında da bazı parametreler önerilmiş olsa da bunlar geniş bir kabul görmemiştir.

Denklem (3.33) iki parametreli bir hal denklemidir. Ancak daha yüksek doğruluk sağlamak için üç parametreli bir denklem geliştirebiliriz. Şimdi aynı Z_c değerine sahip

farklı gaz grupları için aynı $Z = f(T_r, P_r)$ fonksiyonları olduğu varsayılsın. Böylece her Z_c için farklı bir Şekil 3.1-Şekil 3.3 setine sahip olunur. Aynı Z_c değerine sahip tüm akışkanlar aynı Z-T_r-P_r davranışını izler (Prausnitz, 1987). Bu yaklaşım daha doğru olacaktır. Burada kullanılabilecek üçüncü parametre Pitzer asentrik faktörüdür. Bu parametre bir molekülün atomik kuvvetler altında sahip olduğu küresel yapıdan uzaklaşmayı ifade eder. Basit ve tam küresel bir molekül için bu katsayı değeri ω=0 olacaktır. Az önce bahsedilen bakış açısına göre aynı ω değerine sahip gazlar aynı Z_c grubuna dâhil olarak değerlendirilir ve böylece aynı $Z = f(T_r, P_r)$ fonksiyonu ile temsil edilebilirler.

Ancak ayrı ω değeri kullanarak farklı Z, T_r, P_r tabloları geliştirmek yerine lineer bir ilişki kullanılabilir.

$$Z = Z^{(0)}(T_r, P_r) + \omega Z^{(1)}(T_r, P_r) \qquad (3.37)$$

Burada $Z^{(0)}$ küresel moleküller için sıkıştırılabilirlik faktörünü temsil ederken $Z^{(1)}$ küresel yapıda ki sapmayı temsil etmektedir (Pitzer, 1957). Yine aynı kaynata Pitzer çeşitli moleküller için asentrik faktörleri tablo halinde sunmuştur.

3.4.1.1 Analitik Hal Denklemleri

Bir analitik hal denklemi basınç, sıcaklık ve molar hacim arasında cebirsel ilişki verir. Bu bölümde yapılan çalışmada kullanılan üç farklı hal denklemi hakkında bilgi verilecektir. Bu hal denklemleri belirli sınırlar altında sıvı hal içinde bize basınç, sıcaklık ve molar hacim ilişkisi sunabilmektedirler. Bu denklemler basitden daha karmaşık ve doğruluğu daha yüksek olana doğru sıralanmıştır. Ancak programlama sırasında karmaşık denklemlerin çözümü daha zor ve yavaş olduğu için doğruluğu ve karmaşıklığı en yüksek olan hal denklemine çözüm kolaylığı sağlayabilmek için daha basit olan diğer denklemlerden faydalanılmıştır.

3.4.1.1.1 Virial Hal Denklemi

Bu hal denklemin geçerliliği istatistiksel mekanikten yola çıkarak gösterilebilir. Bu denklemde basınç değeri sıcaklık ve hacmin tersinin bir seriye açılımı şeklindedir.

$$P = \frac{RT}{V} + \frac{RTB}{V^2} + \frac{RTC}{V^3} + \ldots \tag{3.38}$$

Buradaki *B, C,* ...parametreleri ikinci, üçüncü … viral katsayılar olarak adlandırılır ve saf bir akışkan için yalnızca sıcaklığın bir fonksiyonudurlar. Bu hal denklemi üzerine bir çok çalışma yapılmıştır. Bu hal denklemine olan ilginin bir sebebi *B, C,*...katsayılarının moleküler potansiyel

fonksiyonu ile yakından ilgili olmalarıdır. Üçüncü ve daha yüksek mertebeden katsayılar hakkında çok fazla bilgi mevcut değildir. Bu yüzden bu denklem genellikle ikinci terimden sonrası atılarak kullanılır. Bu yüzden de doğruluğu sınırlıdır. Bu durumda denklemin kullanabileceğimiz kısmı şu şekilde olacaktır.

$$Z = 1 + \frac{BP}{RT} \tag{3.39}$$

ve

$$Z = 1 + \frac{B}{V} \tag{3.40}$$

B katsayısının elde edilebilmesi için şu yaklaşım kullanılır.

$$B = \lim_{1/V \to 0} \left(\frac{\partial Z}{\partial 1 / V} \right)_T \tag{3.41}$$

Denklem (3.39) ve (3.40) düşük yoğunlukta birbirlerine yaklaşırlar ve doğrulukları artar. Ancak yoğunluk kritik yoğunluğun yarısından büyük olduğu durumlarda bu denklem büyük oranda hatalı olacaktır ve bu sınırın dışında kullanılması uygun değildir.

B katsayısının belirlenmesi için farklı türden çalışmalar yapılmıştır bunlara göre polar olmayan moleküller için *B* katsayısı şu eşitliklerden elde edilebilir.

$$\frac{BP_c}{RT_c} = B^{(0)} + \omega B^{(0)} \tag{3.42}$$

$$B^{(0)} = 0.083 - \frac{0.422}{T_r^{1.6}} \tag{3.43}$$

$$B^{(1)} = 0.139 - \frac{0.172}{T_r^{4.2}} \tag{3.44}$$

Bunun dışında bir çok farklı yaklaşım sunulmuştur ancak bunlar daha özellikli bileşikler be molekül grupları için sunulmuştur dolayısı ile bu denklem formunun en genel olanını kullanmak daha uygun olmaktadır.

3.4.1.1.2 Kübik Hal Denklemi

Kübik hal denklemi adını içerdiği birinci ikinci ve üçüncü dereceden hacim teriminden alır. Birçok farklı kübik denklem için genel form şu şekildedir. (Prausnitz, 1987)

$$P = \frac{RT}{V-b} - \frac{a}{V^2 + ubV + \omega b^2} \tag{3.45}$$

Gerçekte bu denklemin kübik hal denklemi olarak adlandırılması ise yukarda ki forma denk olan şu ifadedir.

$$Z^3-(1+B^*-uB^*)Z^2+(A^*+\omega B^{*2}-uB^*-uB^{*2})Z \\ -A^*B^*-\omega B^{*2}-\omega B^{*3}=0 \tag{3.46}$$

$$A^*=\frac{aP}{R^2T^2} \tag{3.47}$$

$$B^*=\frac{bP}{RT} \tag{3.48}$$

En çok bilinen kübik denklemler van der Waals, Redlich-Kwong (RK), Soave (SRK) ve Peng Robinson (PR) denklemleridir. Bu dört denklem için *u* ve *ω* değerleri tam sayıdır. Denklem (3.47) ve (3.48) da görülen *a* ve *b* katsayıları için farklı yaklaşımlar vardır. Tüm bu denklemler için kullanılan katsayılar Çizelge 3-1 de verilmiştir. Bu çalışmada da bu tabloda verilen denklem ve katsayılar kullanılmıştır.

Çizelge 3-1: Çeşitli hal denklemlerinin katsayıları

Denklem	u	ω	b	a
Van der waals	0	0	$\frac{RT_c}{8P_c}$	$\frac{27}{64}\frac{R^2T_c^2}{P_c}$
Redlich-Kwong	1	0	$\frac{0.08664RT_c}{P_c}$	$\frac{0.42748R^2T_c^{2.5}}{P_cT^{1/2}}$
Soave	1	0	$\frac{0.08664RT_c}{P_c}$	$\frac{0.42748R^2T_c^{2.5}}{P_c}\left[1+f_\omega(1-T_r^{1/2})\right]^2$ $f_\omega = 0.48+1.574\omega-0.176\omega^2$
Peng-Robinson	2	-1	$\frac{0.07780RT_c}{P_c}$	$\frac{0.45724R^2T_c^2}{P_c}\left[1+f_\omega(1-T_r^{1/2})\right]^2$ $f_\omega = 0.37464+1.54226\omega-0.26992\omega^2$

Soave ve Peng-Robinson denklemlerinde a parametresi sıcaklığın ve asentrik fonksiyon ω'nın bir fonksiyonudur.

Şekil 3.4 de RK ve SRK denklemlerin den elde edilen hacim değerlerinin doğruluğu görülmektedir. Bu grafikler den de anlaşılacağı gibi T_r ve P_r değerleri 1 civarında iken Soave parametreli RK denklemin doğruluk değeri kabul edilebilir seviyede değildir.

3.4.1.2 Benedict-Webb-Rubin Denklemleri

Benedict-Webb-Rubin denklemleri kübik denklemlerden daha karmaşık ve geniş bir sıcaklık ve basınç aralığı için daha doğrudur. BWR denklemleri için gerekli olan sabitler birçok kaynakta sunulmuştur. Orijinal BWR denkleminin başarısı bu denklemler üzerine çalışmaların yoğunlaşmasına sebep olmuştur.

Lee ve Kesler modifiye edilmiş bir BWR denklemi önermişlerdir ve bu denklem daha önce bahsedilen Pitzer üç parametre yaklaşımını kullanmaktadır. Bu denklemi analitik formda kullanırken dikkatli olmak gerekmektedir. Gerçek bir gaz için sıkıştırılabilirlik faktörü ω=0 olan basit gaz ile referans olarak alınacak *n*-octane akışkanının sıkıştırılabilirlik faktörü arasında olacaktır. Şimdi bir akışkan için Z değerini hesaplamaya çalıştığımızı varsayılsın. İlk önce akışkanın T_r ve P_r kritik değerlerini bulmamız gerekir. Ardından bu değerleri kullanarak basit akışkan (ω=0) için ideal indirgenmiş hacim aşağıdaki denklem aracılığı ile hesaplanır.

$$\frac{P_r V_r^{(0)}}{T_r} = 1 + \frac{B}{V_r^{(0)}} + \frac{C}{(V_r^{(0)})^2} + \frac{D}{(V_r^{(0)})^5} + \frac{c_4}{T_r^3 (V_r^{(0)})^2}\left[\beta + \frac{\gamma}{(V_r^{(0)})^2}\right]\exp\left[-\frac{\gamma}{(V_r^{(0)})^2}\right] \tag{3.49}$$

Burada ki katsayılar şu şekilde hesaplanır.

$$B = b_1 - \frac{b_2}{T_r} - \frac{b_3}{T_r^2} - \frac{b_4}{T_r^3} \tag{3.50}$$

$$C = c_1 - \frac{c_2}{T_r} + \frac{c_3}{T_r^3} \tag{3.51}$$

$$D = d_1 + \frac{d_2}{T_r} \tag{3.52}$$

Denklem (3.52)'de $V_r^{(0)} = P_c V^{(0)} / RT_c$ olmak üzere ve Denklem (3.50), (3.51) ve (3.52) de geçen katsayılar Tablo 3.2 den alınmak üzere $V_r^{(0)}$ bir kez hesaplandığında basit gazın bu sıcaklık ve basınca karşılık gelen sıkıştırılabilirlik faktörü hesaplanır.

$$Z^{(0)} = \frac{P_r V_r^{(0)}}{T_r} \tag{3.53}$$

Daha sonra aynı indirgenmiş sıcaklık ve basınç değerleri ve Denklem (3.49) kullanılarak bu kez $V_r^{(R)}$ yani

referans *n*-octane için çözüm yapılır. Ancak bu kez basit akışkan için kullanılan katsayılar yerine referans sıvının katsayıları kullanılır ki bu katsayılar yine Tablo 3.2 de verilmiştir. Yine aşağıdaki bağıntıdan $Z^{(R)}$ hesaplanır.

$$Z^{(R)} = \frac{P_r V_r^{(R)}}{T_r} \tag{3.54}$$

Çizelge 3-2: BWR Denkleminde kullanılan basit ve referans gaza ait katsayılar

Katsayılar	Basit akışkan	Referans akışkan
b_1	0.1181193	0.2026579
b_2	0.265728	0.331511
b_3	0.154790	0.027655
b_4	0.030323	0.203488
c_1	0.0236744	0.0313385
c_2	0.0186984	0.0503618

c_3	0.0	0.016901
c_4	0.042724	0.041577
d_1 x 10^4	0.155488	0.48736
d_2 x 10^4	0.623689	0.0740336
β	0.65392	1.226
γ	0.060167	0.03754

Bu hesaplamaların ardından ilgili akışkanın sıkıştırılabilirlik faktörü basit ve referans akışkanların yukarıda açıkladığı gibi hesaplanan sıkıştırılabilirlik faktörleri kullanılarak aşağıdaki bağıntı ile hesaplanır.

$$Z = Z^{(0)} + \left(\frac{\omega}{\omega^{(R)}} \right) (Z^{(R)} - Z^{(0)}) \qquad (3.55)$$

Burada $\omega^{(R)} = 0.3978$ dir.

Denklem (3.55) da kullanılan $Z^{(0)}$ ve $Z^{(R)}$ değerleri belirli indirgenmiş basınç ve sıcaklık değerleri için tablo halinde verilebilir. Ancak bu teze konu olan çalışmada hal

denklemi olarak yukarıda geçen Lee-Kesler denklemi kullanılmıştır ve bu denklem her indirgenmiş basınç ve sıcaklık için yeniden hesaplanmış böylece denklemin sahip olduğu doğruluk programa yansıtılmıştır. Aksi takdirde kullanılacak bir tablo için belirtilmemiş ara değerleri hesaplamak doğruluktan fedakârlık etmek olacaktır.

3.4.1.3 Gerçek Gazların Termodinamik Özellikleri

Gerçek gazlar için hal denklemi geliştirildikten sonra gazların termodinamik özellikleri Bölüm 2 de çıkartılan ilişkiler kullanılarak geliştirilebilir.

Daha önceki bölümde geliştirilen termodinamik özelliklerin herhangi birinin iki hal arasında kullanacağı yol arasında hiçbir fark oluşmayacaktır. Örneğin sabit bir bileşime sahip bir karışımın veya saf bir akışkanın P_1, T_1 ile P_2, T_2 arasında ki değişimi için sayısız farklı yol bulunmaktadır.

$$H = f(P,T)$$

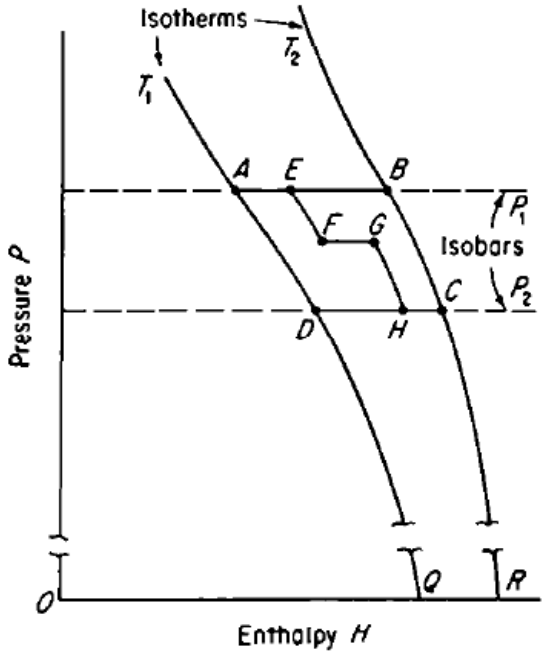
$$dH = \left(\frac{\partial H}{\partial P}\right)_T dP + \left(\frac{\partial H}{\partial T}\right)_P dT$$

$$H_2 - H_1 = \int_{P_1}^{P_2} \left(\frac{\partial H}{\partial P}\right)_{T_1} dP + \int_{T_1}^{T_2} \left(\frac{\partial H}{\partial P}\right)_{P_2} dT \qquad (3.56)$$

$$H_2 - H_1 = \int_{P_1}^{P_2} \left(\frac{\partial H}{\partial P} \right)_{T_2} dP + \int_{T_1}^{T_2} \left(\frac{\partial H}{\partial T} \right)_{P_1} dT \qquad (3.57)$$

Yukarıda ki Denklem (3.56) ve (3.57) aynı ilk ve son haller arasında takip edilebilecek yollardan ikisini vermektedir.

İlk metot da T_1 sıcaklığı sabit tutulup P_1 den P_2 basıncına geçilir ardından bu kez sabit basınç altında T_1 sıcaklığından T_2 sıcaklığına ulaşılır bu sürecin entalpi değişimi Denklem (3.56) de ki gibi hesaplanır. İkinci yolda ise önce sabit P_1 basıncı altında T_1 sıcaklığından T_2 sıcaklığına geçilir ve daha sonra sabit sıcaklıkta P_1 basıncından P_2 basıncına ulaşılır. Bu yoldaki entalpi değişimi de Denklem (3.57) de ki gibi hesaplanabilir. Bunların dışında aynı hal değişimi birbirini takip eden çeşitli izobarik ve izotermal süreçler ile gerçekleştirilebilir. Bu ve buna benzer birkaç yol Şekil 3.3 de gösterilmiştir. (Prausnitz , 1967)



Şekil 3-4: İki hal arasında gerçekleşebilecek farklı yollar

Daha önce Denklem (3.21) de çıkarılan eşitliğe göre (3.57) denkleminin ikinci terimi olan $\int_{T_1}^{T_2}\left(\frac{\partial H}{\partial T}\right)_{P_1} dT$ sabit basınç altında ki sıcaklık değişimidir ve bu entegralin iç kısmı bize c_p'yi verecektir. Ancak farklı basınç değerlerinde c_p değeri de farklılık gösterir ve genellikle elimizde düşük basınç da ve ideal gaz davranışının sergilendiği bölgede ki c_p değerlerine sahip olunur. Bu yüzden hesaplamayı

kolaylaştırmak için hesaplayabildiğimiz değerlere bizi taşıyan bir yol tercih edilir.

Bu durumda tercih edeceğimiz yol Şekil 3.3 den AQRC olacaktır. Bu yol için entalpi değişimi de

$$\Delta H = \int_{P_1}^{P^0} \left(\frac{\partial H}{\partial P} \right)_{T_1} dP + \int_{T_1}^{T_2} c_p^0 dT + \int_{P^0}^{P_2} \left(\frac{\partial H}{\partial P} \right)_{T_2} dP \qquad (3.58)$$

olacaktır. Böylece düşük basınç altında kolaylıkla c_p^0 ideal gaz özgül ısısı kullanılarak hesaplama yapılabilir. Ancak burada da ideal gaz basıncı ile gazın bulunduğu ilk ve son halleri arasında kalan basınçlar arasındaki birinci ve üçüncü entegrallerin hesaplanması gerekmektedir. Bu iki integral fark fonksiyonları olarak adlandırılır ve $(H^0 - H_{P_1})_{T_1}$ ile ifade edilir. Bu durumda yeni denklem

$$\Delta H = (H^0 - H_{P_1})_{T_1} + \int_{T_1}^{T_2} c_p^0 dT - (H^0 - H_{P_2})_{T_2} \qquad (3.59)$$

Denklem (3.59) de ki fark fonksiyonları daha önce ki bölümde geliştirilen hal denklemleri kullanılarak hesaplanabilir. İkinci entegral ise ideal gazın entalpi değişimidir ve akışkana ait c_p fonksiyonunun bilinmesi ile kolayca hesaplanabilir.

3.4.1.4 Fark Fonksiyonlarının Hesaplanması

$\mathcal{L}$ saf bir akışkanın belirli P, T değerleri için termodinamik bir özelliği olsun. Eğer $\mathcal{L}^o$ bu termodinamik özelliğin aynı sıcaklıktaki ancak ideal gaz halindeki değerini temsil ederse $\mathcal{L}$-$\mathcal{L}^o$ fonksiyonu fark fonksiyonu olarak adlandırılır. $\mathcal{L}^o$ referans hal olarak adlandırılır. Referansın tanımını tamamlamak için P^0 veya V^0'ında tanımlanması gerekir. Burada referans olarak P^0 sabit veya V^0 sabit alınabilir. Bunların dışında $P^0=P$ veya $V^0=V$ olarak sistem basıncına bağlı bir referans da seçilebilir.

Bu bölümde ileride görebileceğimiz gibi fark fonksiyonları akışkanın PVT özellikleri kullanılarak elde edilebilir. Burada iki farklı yaklaşım kullanılmaktadır. Akışkanın basıncı hal denkleminde açık olarak belirtilmişse ilk yaklaşım daha uyun iken ikinci yaklaşım sıcaklık ve basınç bağımsız değişken olarak ifade edilmiş ise daha uygun olmaktadır. Bir önceki kısım da ifade edilen denklemler ilk yaklaşıma daha uygundurlar. Eğer elimizdeki hal denklemi $Z = f(T_r, P_r)$ formunda ise bu kez ikinci yaklaşımı kullanmak daha uygun olmaktadır.

İlk form için Helmholtz enerjisi A için fark fonksiyonunu elde etmeye çalışalım ve bu fonksiyondan yola çıkarak diğer tüm özellik fonksiyonlarını Denklem

(2.38)-(2.43) kullanılarak elde edilebilir. Sabit sıcaklık altında Helmholtz enerjisi değişimi şu şekildedir.

$$da = -Pdv \tag{3.60}$$

Yukarıda geçen bu diferansiyel eşitliği sabit sıcaklık altında referans hacim ile özelliklerin belirlenmek istendiği hacim arasında entegre edilecek olursa.

$$a - a^0 = -\int_{v^0}^{v} Pdv \tag{3.61}$$

Ancak bu denklem de ki Ancak bu denklem de ki *P'*nin *v* ye bağlı değişimi referans bölgesi olarak alınan v^0 civarında ideal gaz denklemi ile ilişkili iken *v* de gerçek gaz denklemi bu ilişkiyi sağlar dolayısı ile bu entegrali bu şekli ile hesaplamanın imkânı yoktur. Bunun yerine entegrali ideal ve gerçek gazdan oluşacak şekilde ikiye bölmek gerekmektedir.

$$a - a^0 = -\int_{\infty}^{v} Pdv - \int_{v^0}^{\infty} Pdv \tag{3.62}$$

Bu ifadenin ilk kısmı gerçek gaz için geçerli iken ikinci kısım ideal gaz denklemi kullanılarak bulunabilir. Ancak burada ki bir sorun ideal gaz hal denkleminin entegralinin $v = \infty$ değerinde tanımsız olmasıdır. Oysaki

gerçek gaz denklemleri $v=\infty$ değerinde genellikle tanımlıdırlar ve 0 değerini verirler. Bu tanımsızlıktan kaçmanın yolu bu entegral ifadesinin sağ tarafına $\int_{\infty}^{v} RT/v dv$ terimini ekleyip çıkartmak olacaktır. Bu durumda yeni denklem şu hali alır.

$$a-a^0=-\int_{\infty}^{v}\left(P-\frac{RT}{v}\right)dv-RT\ln\frac{v}{v^0} \tag{3.63}$$

Bu şekilde (3.63) denkleminin sağ tarafında ikinci ifade ideal gaz için bizim istediğimiz hacim ile referans haldeki ideal gazın hacminin farkını ifade eder. Yani ideal gazın iki farklı haldeki hacim farkından doğan Helmholtz enerjisi farkını. İlk terim ise bize gerçek gazla ideal gaz arasındaki ∞ ile v arasında ki hacim farkından doğan Helmholtz enerji farkını verir. Bu noktadan sonra daha önceki bölümde ifade edilen gerçek akışkanlar için hal denklemlerinden biri bu ifade de yerine yazılarak Helmholtz fark fonksiyonu kolayca türetilebilir. Diğer termodinamik özellikler için fark fonksiyonlarını da şu bağıntılardan elde edilebilir.

$$\begin{aligned} s-s^0 &= -\frac{\partial}{\partial T}(a-a^0) \\ &= \int_{\infty}^{v}\left[\left(\frac{\partial P}{\partial T}\right)_v-\frac{R}{v}\right]dv+R\ln\frac{v}{v^0} \end{aligned} \tag{3.64}$$

$$h - h^0 = (a - a^0) + T(s - s^0) + RT(Z - 1) \qquad (3.65)$$

$$u - u^0 = (a - a^0) + T(s - s^0) \qquad (3.66)$$

$$g - g^0 = (a - a^0) + RT(Z - 1) \qquad (3.67)$$

Bu şekilde *P*'nin açık olarak ifade edilebildiği herhangi bir hal denklemi ile yukarıdaki bağıntılar kullanılarak tüm termodinamik özellikler hesaplanabilir.

Yukarıda çıkartılan ifadeler dikkatli bir şekilde incelenirse görülecektir ki $h - h^0$, $u - u^0$ ifadeleri referans basınç veya hacme bağlı değildirler. Ancak $g - g^0$, $s - s^0$ ve $a - a^0$ gibi fonksiyonlar referans hacme bağlıdırlar. Burada referans olarak *P* veya *V* değeri referans alınıp bu denklemlerde kullanılabilir. En yaygını ve bu çalışmada kullanılanı ise *P*=1 bar referans alınmasıdır. Bu durumda *v*=*RT* olacaktır.

İkinci hesaplama biçiminde ise basınç ve sıcaklık değerleri bağımsız değişken olarak verilmişse ve *v* değeri açık olarak elde edilebiliyorsa bu kez Gibbs enerjisinden faydalanılarak fark fonksiyonları hesaplanabilir. Yukarıda takip edilen yol aynen Gibbs için de izlenirse şu eşitliklere ulaşılabilir.

$$g - g^0 = \int_0^P \left(v - \frac{RT}{P} \right) dP + RT \ln \frac{P}{P^0} \tag{3.68}$$

$$s - s^0 = \frac{-\partial}{\partial T}(g - g^0)_P \tag{3.69}$$

$$h - h^0 = (g - g^0) + T(s - s^0) \tag{3.70}$$

$$u - u^0 = (g - g^0) + T(s - s^0) - RT(Z - 1) \tag{3.71}$$

$$a - a^0 = (g - g^0) - RT(Z - 1) \tag{3.72}$$

Yine incelersek görülür ki bu şekilde elde edilen fark fonksiyonları da referans basınç veya hacim değerinden bağımsızdır.

3.4.1.5 Fark Fonksiyonlarının Değerlendirilmesi

Denklem (3.63) ile (3.72) arasında tanımlanan fark fonksiyonları *PvT* verilerinden yola çıkarak hesaplanabilir. Genel olarak analitik bir hal fonksiyonu denklemi *PvT* davranışını ifade etmek için kullanılır.

Daha önce ifade edilen analitik hal denklemlerinden herhangi bir kullanılarak bu özellikler elde edilebilir. Bu çalışmada *PvT* ilişkisi olarak (3.49) ifade edilen Benedict-Webb-Rubin denklemi kullanılmıştır ve bütün

termodinamik özellikler bu ilişkiden yola çıkarak hesaplanmıştır. Bu durumda kullanılan analitik termodinamik özellik denklemleri şu şekilde olmaktadır.

$$\left(\frac{H^0-H}{RT_c}\right)^0 = -T_r\left[Z^{(0)}-1-\frac{b_2+2b_3/T_r+2b_4/T_r^2}{T_r(V_r^{(0)})}\right.$$
$$\left.-\frac{c_2-3c_3/T_r^2}{2T_r(V_r^{(0)})^2}+\frac{d_2}{5T_r(V_r^{(0)})^5}+3E\right] \qquad (3.73)$$

$$E=\frac{c_4}{2T_r^3\gamma}\left\{\beta+1-\left[\beta+1+\frac{\gamma}{(V_r^{(0)})^2}\right]\right.$$
$$\left.\exp\left[-\frac{\gamma}{(V_r^{(0)})^2}\right]\right\} \qquad (3.74)$$

Daha sonra bu kez $V_r^{(0)}$ yerine $V_r^{(R)}$'yi kullanarak ve basit akışkan yerinede referans akışkanın katsayıları kullanılarak $\left(\frac{H^0-H}{RT_c}\right)^0$ elde edilir. Bu iki değer denklem (3.75) de yerine konulduğunda istenen akışkan için entalpi fark fonksiyonunun değerine ulaşılır.

$$\frac{H^0-H}{RT_c}=\left(\frac{H^0-H}{RT_c}\right)^{(0)}+\frac{\omega}{\omega_R}\left[\left(\frac{H^0-H}{RT_c}\right)^{(R)}-\left(\frac{H^0-H}{RT_c}\right)^{(0)}\right] \tag{3.75}$$

$$\left(\frac{S^0-S}{R}\right)^{(0)}=-\ln\frac{P^0}{P}-\ln Z^{(0)}+\frac{b_1+b_3/T_r^2+2b_4/T_r^3}{V_r^{(0)}}+\frac{c_1-2c_3/T_r^3}{2(V_r^{(0)})^2}+\frac{d_1}{5(V_r^{(0)})^5}-2E \tag{3.76}$$

$$\frac{S^0-S}{R}=\left(\frac{S^0-S}{R}\right)^{(0)}+\frac{\omega}{\omega_R}\left[\left(\frac{S^0-S}{R}\right)^{(R)}-\left(\frac{S^0-S}{R}\right)^{(0)}\right] \tag{3.77}$$

Hal denklemleri daha önce kullanıldığı gibi yalnızca gaz fazının değil aynı zamanda sınırlıda olsa sıvı faza ait *PvT* ilişkisinide vermektedir. Dolayısı ile sıvı faza ait hesaplamaların yapılabilmesi olanaklıdır. Eğer sıvı faz için entalpiyi hesaplamak istersek fark fonksiyonunu parçalara ayırmamız daha uygun olacaktır.

$$H^L - H^0 = (H^L - H^{SL}) + (H^{SL} - H^{SV}) + (H^{SV} - H^0) \quad (3.78)$$

Burada,

H^L = *T, P* noktasında ki sıvı entalpiyi

H^0 = *T,* P^0 ideal gaz entalpisini

H^{SL} = *T,* P_{vp} doymuş sıvının entalpisi

H^{SV} = *T,* P_{vp} deki doymuş buharın entalpi değerini temsil etmektedir.

Burada ki $(H^{SV} - H^0)$ değeri yukarı da belirtildiği gibi kolayca hesaplanabilir. Bunun için tek gereken doyma basıncının tespit edilmesidir ki bunun için gerekli yöntemler bir sonraki bölümde incelenecektir. $(H^{SL} - H^{SV})$ ise yoğuşma entalpisidir fonksiyon yine bir sonraki kısımda incelenecektir. Üçüncü ve son parça olan $(H^L - H^{SL})$ ise doyma noktasındaki sıvı ile aşırı soğumuş sıvı fazdaki akışkanın entalpi farkıdır. Yine bu fark da yukarıda belirtildiği gibi Lee-Kesler denklemi sıvı faz için kullanılarak hesaplanabilir.

3.4.1.6 Sıvı Faz Yoğunluk Denklemleri

Bunun yanında direkt olarak sıvı faza ait yoğunluğun hesaplanmasına imkân veren bağıntılarda mevcuttur. Ancak bu çalışmada bu bağıntılar yerine yukarıda belirtildiği gibi sıvı faz için Lee-Kesler denklemi kullanılmıştır. Her ne kadar bu denklemler kullanılmasa da program içinde seçenek olarak bulunmaktadırlar. Kısaca bunlardan bahsedilebilir.

Rackett Yöntemi: Bu denklem Rackett tarafından geliştirilmiş daha sonra Spancer ve Danner tarafından iyileştirilmiştir. Bu denklem doymuş sıvının hacmini belirlemek için uygun bir denklemdir. (Prausnitz, 1987)

$$V_s = \frac{RT_c}{P_c} Z_{RA}^{\left[1+(1-T_r)^{2/7}\right]} \tag{3.79}$$

$$Z_{RA} = 0.2959 - 0.08775\omega \tag{3.80}$$

Z_{RA} her akışkanının kendisine ait bir özelliktir ve Denklem (3.80) ile bulunabilir.

Bhirud Yöntemi: Bhirud doymuş sıvıların hacminin tespiti için aşağıdaki yöntemi önermiştir. Bu denklem özellikle polar olmayan akışkanlar için doğruluk oranı yüksektir. (Prausnitz, 1987)

$$\ln \frac{P_c V_s}{RT} = \ln V^{(0)} + \omega \ln V^{(1)} \tag{3.81}$$

$$\begin{aligned} \ln V^{(0)} &= 1.3964 - 24.076T_r + 102.615T_r^2 \\ &-255.719T_r^3 + 355.805T_r^4 - 256.671T_r^5 \\ &+75.1088T_r^6 \end{aligned} \tag{3.82}$$

$$\begin{aligned} \ln V^{(1)} &= 13.4412 - 135.7437T_r + 533.380T_r^2 \\ &-1091.453T_r^3 + 1231.43T_r^4 - 728.227T_r^5 \\ &+176.737T_r^6 \end{aligned} \tag{3.83}$$

3.4.2 Doymuş Buhar Basıncının Hesaplanması

Bir saf akışkan buharı ile denge halinde ise kimyasal potansiyelin eşitliği bizi her iki fazın sıcaklık ve basınç değerleri ile ilişkili olan Clausıus-Clapeyron denklemine götürür. (Prausnitz, 1987)

$$\frac{dP_{vp}}{dT} = \frac{\Delta H_v}{T \Delta V_v} = \frac{\Delta H_v}{(RT^2 / P_{vp}) \Delta Z_v} \tag{3.84}$$

$$\frac{d \ln P_{vp}}{d(1/T)} = -\frac{\Delta H_v}{R \Delta Z_v} \tag{3.85}$$

Bu denklemlerde ki ΔH_v ve ΔZ_v doymuş buhar ve doymuş sıvının entalpileri ve sıkıştırılabilirlik faktörleri arasındaki farkı temsil etmektedir.

Buhar basıncını belirlemek için geliştirilen denklemlerin birçoğu yukarıda verilen denklemlerin entegrasyonu üzerine geliştirilmiştir. Bunun gerçekleştirilebilmesi için $\Delta H_v / \Delta Z_v$ oranının sıcaklığa bağlı değişiminin ve P_{vp} nin sıcaklıkla değişiminin bilinmesi gerekmektedir.

Burada yapılabilecek en basit yaklaşım $\Delta H_v / \Delta Z_v$ oranının sıcaklıkla değişmediğini ve sabit kaldığını P_{vp} ise aşağıdaki gibi sıcaklıkla ilişkili olduğunu farz etmektir.

$$\ln P_{vp} = A - \frac{B}{T} \tag{3.86}$$

Burada $B = \Delta H_v / R\Delta Z_v$ bu denklem genellikle Clapeyron denklemi olarak adlandırır. Bu denklem kritik sıcaklık civarı dışında küçük sıcaklık aralıkları için iyi sonuçlar verir ancak geniş sıcaklık aralıklarında etkili değildir.

Bir adım ileri gidersek yaygın bir uygulama normal ve kritik kaynama noktalarının beraber kullanılarak bir denklemin geliştirilmesidir. Basınç değerleri bar ve sıcaklık değerleri de K olmak üzere şu denklem kullanılabilir.

$$\ln P_{vpr} = h\left(1 - \frac{1}{T_r}\right) \tag{3.87}$$

$$h = T_{br}\frac{\ln(P_c / 1.01325)}{1 - T_{br}} \tag{3.88}$$

Ancak bu denklem üzerine yapılan çalışmalarda göstermektedir ki bu denklem özellikle T_b civarında gerçeğin üzerinde tahmin de bulunmaktadır. Bu denklem iki parametreli bir denklemdir ve en başta kullandığımız Pitzer yaklaşımını kullanarak üç parametreli ve daha geçerli bir yaklaşım geliştirebiliriz. (Prausnitz, 1987)

$$\ln P_{vpr} = f^{(0)}(T_r) + \omega f^{(1)}(T_r) \tag{3.89}$$

$f^{(0)}$ ve $f^{(1)}$ fonksiyonları deneyler ile tespit ampirik olarak tespit edilmiştirr ancak geniş bir aralık için bu verilere uydurulan eğriler bu fonksiyonlar için kullanılabilir. Aşağıda bu amaç uydurulmuş eğriler verilmektedir.

$$f^{(0)} = 5.92714 - \frac{6.09648}{T_r} - 1.28862\ln T_r + 0.169347T_r^6 \tag{3.90}$$

$$f^{(1)} = 15.2518 - \frac{15.6875}{T_r} - 13.4711 \ln T_r + 0.43577 T_r^6 \qquad (3.91)$$

3.4.2.1 Antonie Denklemi

Antonie denklem (3.86)'e benzer bir ilişki önermiştir. (Prausnitz, 1987)

$$\ln P_{vp} = A - \frac{B}{T + C} \qquad (3.92)$$

C katsayısını normal kaynama noktası ile ilişkilendirecek bazı denklemler üzerine çalışılmıştır ancak bunların hiçbiri yeterli sonuç vermemiştir. Bu yüzden bu denklemi kullanmak için en iyi yöntem Tablo haline getirilmiş deneysel verileri kullanmaktır. Bununla ilgili veriler (Prausnitz, 1987)'de Appendix A de verilmiştir. Ancak bu katsayılarında uygulanabileceği sıcaklık aralığı kısıtlıdır ve bu Arallığa dikkat edilmelidir.

3.4.2.2 Gomez-Thodos Denklemi

Gomez –Nieto ve Thodos doymuş buhar basınçlarının belirlenmesi için aşağıdaki denklemi önermişlerdir. (Gomez-Nieto, 1978).

$$\ln P_{vpr} = \beta\left[\frac{1}{T_r^m} - 1\right] + \gamma[T_r^7 - 1] \tag{3.93}$$

Normal kaynama noktası kullanılarak da *β, γ, m* katsayılarını birbirine bağlayan bir denklem geliştirilebilir.

$$\gamma = ah + b\beta \tag{3.94}$$

$$a = \frac{1 - 1/T_{br}}{T_{br}^7 - 1} \tag{3.95}$$

$$b = \frac{1 - 1/T_{br}^m}{T_{br}^7 - 1} \tag{3.96}$$

Bu noktadan sonra *h* ve diğer katsayılar belirlenmesi özellikleri istenen bileşiğin türüne göre değişmektedir bu katsayılar polar olmayan akışkanlar, su ve alkoller dışında kalan polar akışkanlar için ayrı ayrı yöntemler ile hesaplanmaktadır. Ancak bu teze konu olan programda bu denklem ayrıntılı olarak kullanılmamıştır. Bunun sebebi diğer üç denklemin yeterli doğrulukta sonuç vermesi ve diğer denklemlerinin kullanımının bu denkleme göre daha kolay olmasıdır. Bu yüzden burada yalnızca polar olmayan akışkanlar içim bu denklemin katsayılarının nasıl tespit edileceği belirtilecektir.

Polar olmayan bileşikler için Gomez-Thodos denkleminin katsayıları şu şekilde ifade edilebilir. (Prausnitz, 1987)

$$\beta = -4.267 - \frac{221.79}{h^{2.5}\exp 0.0384h^{2.5}} + \frac{3.8126}{\exp(2272.44/h^3)} + \Delta^* \tag{3.97}$$

He için Δ^*=0.41815, H_2 için Δ^*=0.19904 ve diğer tüm bileşikler için Δ^*=0 dır. (Prausnitz, 1987)

$$m = 0.78425\exp(0.089315h) - \frac{8.5217}{\exp(0.74826h)} \tag{3.98}$$

3.4.2.3 Doymuş Buhar Basıncının İki Referans Akışkan ile Belirlenmesi

Lee-Kesler metodunda biri $\omega=0$ olan bir basit akışkan ile gerçek bir referans akışkanın özellikleri kullanılarak gerçek gaz özellikleri tahmin edilmeye çalışılmıştı. Daha sonra birçok araştırmacı basit akışkan yerine de gerçek bir referans akışkanın kullanılması gerektiğini ileri sürmüşlerdir. Bu şekilde doymuş buhar basıncı için yaklaşım geliştirilmiştir.

$$\ln P_{vpr} = \ln P_{vpr}^{(R1)} + (\ln P_{vpr}^{(R2)} - \ln P_{vpr}^{(R1)}) \frac{\omega - \omega^{(R1)}}{\omega^{(R2)} - \omega^{(R1)}} \quad (3.99)$$

Burada R1 ve R2 indisleri iki farklı gerçek referans gazının ifade etmektedir. Bu gazların her birinin ln P_{vpr} değerleri ise aşağıdaki denklem aracılığı ile hesaplanacaktır. (Prausnitz, 1987)

$$\ln P_{pr} = \frac{a\tau + b\tau^{1.5} + c\tau^3 + d\tau^6}{T} \quad (3.100)$$

Bu denklemde ki τ =1- T_r dir. Diğer katsayılar farklı gerçek gaz referansları için aşağıdaki Çizelge de verilmiştir.

Çizelge 3-3: Denklem (3.100) için sabitler

Madde	T_c K	P_c bar	ω	a	b	c	d
Propan	369.85	42.45	0.153	-6.722	1.33	-2.13	-1.38
Oktan	568.81	24.86	0.398	-7.91	1.38	-3.80	-4.50
Benzen	562.16	48.97	0.212	-6.792	1.33	-2.62	-3.33

Penta- fulora tuluen	566.52	31.24	0.415	-8.05	1.46	-3.82	-2.78

Bu Çizelge aracılığı ile $\omega^{(R1)} < \omega < \omega^{(R2)}$ olacak şekilde seçilen iki referans noktası ile Denklem (3.100) kullanılarak ln P_{vpr} değerleri tespit edilir ve bunlar doymuş buhar basıncı istenen gazın ω değeri kullanılarak bu gazın buhar basıncı Denklem (3.99) dan hesaplanır.

3.4.2.4 Diğer Denklemler

Şimdiye kadar doymuş buhar basıncını belirlemek için önerilen denklemler konuya teorik yaklaşımlardı. Ancak elde deneyler ile elde edilmiş değerler bulunuyorsa bu değerler kullanılarak geliştirilecek korelasyon ve ekstrapolasyon denklemleri doymuş buhar basıncının belirlenmesi amacı ile kullanılabilir. Bu çalışmada da daha güvenilir oldukları için bu tür denklemler tercih edilmiştir. Bu denklemlere verilebilecek en iyi üç örnek şunlardır.

Kullanılan ilk ilgileşim Wagner denklemine benzer bir formdadır. (Prausnitz, 1987)

$$\ln P_{pr} = \frac{a\tau + b\tau^{1.5} + c\tau^{3} + d\tau^{6}}{T} \qquad (3.101)$$

Bir çok gaz için bu denkleme ait katsayılar (Prausnitz, 1987)'da verilmiştir.

Antonie denklemine yakın bir denklem olarak aşağıdaki denklem bu çalışmada da kullanılan korelâsyonların ikincisidir.

$$\ln P_{vpr} = A - \frac{B}{T} + C\ln T + \frac{DP_{vpr}}{T^{2}} \qquad (3.102)$$

Ancak (3.102) denkleminde görülebileceği gibi bu denklemde P_{vpr} açık bir şekilde ifade edilmemiştir. Dolayısı ile lineer olmayan denklemin sayısal yöntemler kullanılarak çözülmesi gerekmektedir. Bu noktada çözümü başarılı bir ilk tahminle kolaylaştırmak ve hızlandırmak mümkündür. Bunun için en iyi yöntem yukarıda belirtilen teorik denklemlerden biri kullanılarak iyi bir başlangıç tahmini yapmaktır. Bu çalışmada bu amaçla Gomez-Thodos denkleminden elde edilen sonuçlar kullanılmıştır.

Kullanılan üçüncü ilgileşim ise Antonie denkleminin kendisidir bazı gazlar için bu denklem yeterli sonuçlar vermektedir. (Prausnitz, 1987)

$$\ln P_{vp} = A - \frac{B}{T + C} \tag{3.103}$$

Bu şekilde elde edilecek fark fonksiyonları aracılığı ile hesaplanan termodinamik özellikler ancak basit hidrokarbonlar ve hafif gazlar için yeterli olabilmektedir. Ancak su gibi bazı akışkanların özelliklerini bu denklemler aracılığı ile belirlemek yeterli doğruluğu sağlayamamaktadır. Bu yüzden su için daha özel denklemeler türetilmiştir.

3.5 Suyun Termodinamik Özelliklerinin Belirlenmesi

Su, büyük oranda polar bir bileşik olması sebebi ile daha çok polar olmayan akışkanların termodinamik modellenmesi için uygun olan yukarıda belirtilen denklemler su için hatalı sonuçlar vermektedir. Birçok sistemde akışkan olarak suyun bulunması bu akışkanın mümkün olduğu kadar doğru bir şekilde modellenmesini zorunlu kılmaktadır. Bu amaçla (Keenan, 1969) dan alınan denklemeler kullanılmıştır.

3.5.1 Temel Denklem

Keenan ve arkadaşları tarafından yürütülen çalışmada suyun özeliklilerini belirlemek için suyun Helmholtz enerjisinden yola çıkarak bir genel denklem belirlenmiş ve diğer bütün termodinamik özellikler tıpkı daha önce Lee-Kesler denkleminin uygulanması sırasında uygulandığı gibi bu Helmholtz fonksiyonundan elde edilmiştir.

Burada kullanılan temle denklem şu şekilde belirtilmiştir.

$$A = A_0(T) + RT[\ln \rho + \rho Q(\rho, T)] \tag{3.104}$$

Burada ρ suyun yoğunluğunu temsil etmektedir. $A_0(T)$ yalnızca sıcaklığa bağlı bir fonksiyondur ve Denklem (3.105) ile ifade edilir. $Q(\rho, T)$ ise T ve ρ ya bağlı bir başka fonksiyondur ve o da Denklem (3.106) ile ifade edilmektedir.

$$A_0 = \sum_{i=1}^{6} C_i / \tau^{i-1} + C_7 \ln T + C_8 \ln T / \tau \tag{3.105}$$

$$Q = (\tau - \tau_c)\sum_{j=1}^{7}(\tau - \tau_{aj})^{j-2}\left[\sum_{i=1}^{8} A_{ij}\,(\tau - \tau_{aj})^{i-1} + e^{-E\rho}\sum_{i=9}^{10} A_{ij}\rho^{i-9}\right] \quad (3.106)$$

Bu denklemlerde ifade edilne T K cinsinden sıcaklığa karşılık gelirken τ ise $1000/T$ yi temsil etmektedir. Ayrıca ρ g/cm^3, R=4.6151 bar cm^3/g K, τ_c= $1000/T_c$, E=4.8 dir. (Keenan, 1969)

$\tau_{aj} = \tau_c$ (j=1), τ_{aj}=2.5 ($j \neq 1$) ve

ρ_{aj}=0.634 (j=1), ρ_{aj}=1.0 ($j \neq 1$)

Çizelge 3-4: Buhar denkleminin katsayıları (1-4)

A_{ij}	1	2	3
1	29.492937	-5.198586	6.833535
2	-132.139170	7.777918	-26.149751

3	274.646320	-33.30102	65.326396
4	-360.938280	-16.25622	-26.181978
5	342.184310	-177.1074	0.000000
6	-244.500420	127.48720	0.000000
7	155.185350	137.46153	0.000000
8	5.972849	155.97836	0.000000
9	-410.30848	337.31180	-137.466180
10	-416.05800	-209.88	-733.968480

Çizelge 3-5: Buhar denklemi katsayıları (4-7)

ij	4	5	6	7
1	-0.156410	-6.397241	-3.966140	-0.690486

2	-0.725461	26.40928	15.45306	2.740742
3	-9.27342	-47.74037	-29.14247	-5.10280
4	4.312584	56.33130	29.56879	3.963609
5	0.000000	0.000000	0.000000	0.000000
6	0.000000	0.000000	0.000000	0.000000
7	0.000000	0.00000	0.000000	0.000000
8	0.0000	0.000000	0.000000	0.000000
9	6.787498	136.8731	79.84797	13.04125
10	10.401	645.8188	399.1757	71.53135

A ve *C* katsayıları Çizelge 3-4, Çizelge 3-5 ve Çizelge 3-6 da verilmiştir.

Çizelge 3-6: Buhar denkleminde ki *C* katsayıları.

C	
1	1857.065
2	3229.12
3	-419.465
4	36.6649
5	-20.5516
6	4.85233
7	46.0
8	-1011.249

3.5.2 Termodinamik Özellikler

Yukarıdaki gibi bir temel denklemin kullanılması bize diğer özelliklerin bu denklemden çıkarılması açısından büyük kolaylık sağlamaktadır. Yukarıda belirtildiği gibi bilinen bir *T, ρ* (*1/v*) için basınç, *P* ve diğer termodinamik özellikleri bulabiliriz.

Denklem (2.42) den bilindiği gibi $dA = -SdT - PdV$ dir buradanda

$$-\left(\frac{\partial A}{\partial V}\right) = P$$

$$-\left(\frac{\partial A}{\partial V}\right)\frac{dV}{d\rho}\frac{d\rho}{dV} = P$$

$$-\left(\frac{\partial A}{\partial \rho}\right)\frac{d\rho}{dV} = P$$

$$\frac{d\rho}{dV} = \left(\frac{dV}{d\rho}\right)^{-1} = \left(\frac{d(1/\rho)}{d\rho}\right)^{-1} = \left(-\frac{1}{\rho^2}\right)^{-1} = -\rho^2$$

olduğu için ;

$$P = \rho^2\left(\frac{\partial A}{\partial \rho}\right) \tag{3.107}$$

olacaktır. Benzer şekilde diğer termodinamik özellikler Denklem (3.104) de ki Helmholtz Denklemi ile uyumlu hale getirilir ise şu eşitliklere ulaşılabilir.

$$u = \left[\frac{\partial(A\tau)}{\partial \tau}\right]_{\rho} \tag{3.108}$$

$$s = -\left(\frac{\partial A}{\partial T}\right)_{\rho} \tag{3.109}$$

Ayrıca

$$h \equiv u + Pv \tag{3.110}$$

$$g \equiv a + Pv \tag{3.111}$$

Özellikleri de elde edilebilir. Şimdi bu denklem (3.108)-(3.111) de A yı yerine yazıp eşitlikleri elde edelim.

$$P = \rho RT\left[1 + \rho Q + \rho^2\left(\frac{\partial Q}{\partial \rho}\right)_\tau\right] \tag{3.112}$$

$$u = RT\rho\tau\left(\frac{\partial Q}{\partial \tau}\right)_\rho + \frac{dA_0\tau}{d\tau} \tag{3.113}$$

$$s = -R\left[\ln\rho + \rho Q - \rho\tau\left(\frac{\partial Q}{\partial \tau}\right)_\rho\right] - \frac{dA_0}{dT} \tag{3.114}$$

$$h = RT\left[\rho\tau\left(\frac{\partial Q}{\partial \tau}\right)_\rho + 1 + \rho^2\left(\frac{\partial Q}{\partial \rho}\right)_\tau\right] + \frac{dA_0\tau}{d\tau} \tag{3.115}$$

Ayrıca suya ait doymuş buhar basıncı değerleri de şu denklemden kolayca hesaplanabilir.

$$P_s = P_c \exp\left[\tau 10^{-5}(T_c - T)\sum_{i=1}^{8} F_i(0.65 - 0.01T)^{i-1}\right] \tag{3.116}$$

P_s yoğuşma (doymuş buhar basıncını), P_c kritik basıncı, T_c kritik sıcaklığı ve T doymuş buhar basıncı istenen sıcaklığı temsili etmektedir. Burada ki F katsayıları Çizelge 3-7' da verilmiştir.

Çizelge 3-7: Su buharı doyma eğrisi denklemi katsayıları

F	
1	-741.9242
2	-29.721
3	-11.55286
4	-0.8685635
5	0.1094098
6	0.439993
7	0.2520658
8	0.05218684

4 AKIŞKANLARIN FİZİKSEL ÖZELLİKLERİ

Termodinamik bir sistemin modellemesi sırasında ısı transfer yüzeylerinin elde edilebilmesi için akışkanlara ait termofiziksel özelliklerinde matematiksel olarak doğru bir şekilde modellenmesi gerekmektedir. Bize gerekli olan özellikler sıvı ve gaz hale ait viskozite, iletim katsayısı ve akışkanın sıvı ve gaz fazı arasındaki yüzey gerilim katsayısıdır.

4.1 Akışkanların Viskozitesinin Belirlenmesi

Akışkanların viskoziteleri hesaplanırken hidrokarbonların bu özelliklerini en genel şekilde temsil edecek yöntemleri kullanmak kullanılabilirliği artıracaktır. Bu yüzden genel olarak viskoziteyi verecek denklemler kullanılmıştır.

4.1.1 Gazların Viskozitelerinin Belirlenmesi

Genel olarak bütün gaz viskozite belirleme teknikleri Chapman-Enskog veya karşılıklı haller yasasını temel alır. Chapman-Enskog denklemi aşağıda verilmiştir. (Prausnitz, 1987)

$$\eta = \frac{26.69(MT)^{1/2}}{\sigma^2 \Omega_v} \tag{4.1}$$

Bu denklem Ω_v çarpışma integrali adlı bir parametre içermektedir ve bu parametre boyutsuz T^* ile ilişkilidir. Bu ilişki araştırmalar sonucunda şu şekilde tanımlanmıştır. (Prausnitz, 1987)

$$\Omega_v = [A(T^*)^{-B}] + C[\exp(-DT^*)] + E[\exp(-FT^*)] \quad (4.2)$$

Burada $T^* = kT/\varepsilon$, A=1.16145, B=0.14874, C=0.52487, D=0.77320, E=2.16178 ve F=1.16145. Bu denklem $0.3 \leq T^* \leq 100$ arasında % 0.064 ortalama hata ile geçerlidir. (Prausnitz, 1987)

Chung ve diğer bazı araştırmacılar aşağıdaki eşitliği ileri sürmüşlerdir. (Prausnitz, 1987)

$$\frac{\varepsilon}{k} = \frac{T_c}{1.2593} \quad (4.3)$$

$$\sigma = 0.809 V_c^{1/3} \quad (4.4)$$

Burada ε/k ve T_c Kelvin cinsindendir ve σ angstrom cinsindendir ve V_c ise cm^3/mol dür.

$$T^* = 1.2593 T_r \quad (4.5)$$

Chung Denklem (4.5) ve Denklem (4.2)'i kullanarak Ω_v 'i hesaplama imkânı doğmaktadır.

Chapman-Enskog denkleminin eşitliğin sağ tarafını F_c ile çarpıp viskozite denklemine moleküler şekil ve polariteyi de bir parametre olarak denkleme ilave edilmiştirr. Bu şekilde geliştirilen gazlar için viskozite denklemi şu şekildedir. (Prausnitz, 1987)

$$\eta = 40.785 \frac{F_c (MT)^{1/2}}{V_c^{2/3} \Omega_v} \qquad (4.6)$$

Burada η = viskozite, μP

M = moleküler ağırlık, g/mol

T = sıcaklık, K

V_c = kritik hacim, cm^3/mol

F_c = 1- 0.2756ω + 0.059035 $\mu_r^2 + \kappa$

Bu denklemdeki ω eksentrik faktörü temsil etmektedir. κ ise polarlığı ifade etmektedir. Bu değer 0 ile 0.2 arasında değişmektedir ve polarlık azaldıkça 0.2 değerine yaklaşır. Biz genellikle polar olmayan akışkanlar ile çalıştığımız için bu değeri 0.2 de sabit alınabilir ancak su

veya asitler gibi polar yapıya sahip akışkanlar için bu değeri uygun almamız gerekmektedir. μ_r katsayısı ise boyutsuz dipol momentini ifade eder. Bu değerde Denklem (4.7) den hesaplanabilir. (Prausnitz, 1987)

$$\mu_r = 131.3 \frac{\mu}{(V_c T_c)^{1/2}} \qquad (4.7)$$

Burada μ : dipol moment , debyesdir.

4.1.2 Sıvı Viskozitesinin Belirlenmesi

Sıvı viskozitelerinin belirlenmesi için başarılı bir teorik temel mevcut değildir. Sıvı viskoziteleri aynı sıcaklık da ki gaz viskozitelerine göre çok daha büyüktür. Sıvı viskozitelerini belirlerken basınç ve sıcaklığın etkisini farklı farklı göz önünde bulunduran denklemler öne sürülmüştür. Bu denklemler içinde en genel ve en kullanışlı olanı tercih edilmiştirr. Bu denklem Przezdziecki ve Sridhar yöntemi olarak bilinmektedir ve bu denklem Hildebrand- Batschinski denkleminin den türetilmiştir. (Prausnitz, 1987)

$$\eta_L = \frac{V_0}{E(V - V_0)} \qquad (4.8)$$

η_L : sıvı vizkositesi ,cP centi Poise

V : molar hacim cm^3/mol

$$E = -1.12 + \frac{V_c}{12.94 + A} \tag{4.9}$$

$$A = 0.10M - 0.23P_c + 0.0424T_f - 11.58(T_f / T_c) \tag{4.10}$$

$$V_0 = 0.0085\omega T_c - 2.02 + \frac{V_m}{0.342(T_f / T_c) + 0.894} \tag{4.11}$$

T_f : donma noktası , K

ω : eksentrik faktörü

V_m : T_f de sıvı molar hacmi

Bu denklemde V_m değerini bulmak her zaman mümkün olmamaktadır. Bu sorunu aşmak için Gunn-Yamada metodunu kullanmak uygundur. Bu metot da bilinen bir V değerinden yola çıkarak T_f deki molar hacim değeri saptanabilir. (Prausnitz, 1987)

$$V(T) = \frac{f(T)}{f(T^R)} V^R \tag{4.12}$$

$$f(T) = H_1(1 - \omega H_2) \tag{4.13}$$

$$H_1 = 0.33593 - 0.33953T_r + 1.51941T_r^2 - 2.02512T_r^3 + 1.11422T_r^4 \quad (4.14)$$

$$H_2 = 0.29607 - 0.09045T_r - 0.04842T_r^2 \quad (4.15)$$

V^R olarak kullanabileceğimiz iyi bir yaklaşımla tespit edilmiş referans bir hacim değerini kullanarak V değerini hesaplamak doğru sonuçlar vermektedir.

4.2 Akışkanların Isı İletim Katsayılarının Belirlenmesi

Isı transferinde sıkça kullanılan bir başka parametrede ısı iletim katsayılarıdır bu katsayının hem gaz hem de sıvılar için mümkün olan en iyi şekilde belirlenmesi gerekir.

4.2.1 Gazların Isı İletim Katsayılarının Belirlenmesi

İdeal gazların ısı iletim katsayıları (Prausnitz, 1987)

$$\frac{vLC_v n}{3} \quad (4.16)$$

Denklemi ile ifade edilir. Bu denklem de yer alan v ortalama molekül hızını, L ortalama serbest yolu, C_v

molekül başına ısı kapasitesini ve *n* moleküllerin yoğunluk sayısını temsil eder. Ancak bu yaklaşım bize yeterli doğrulukta sonuç vermemektedir. Bunun en önemli sebebi moleküllerin hızlarının çok geniş bir spektrumda dağılmış olmaları ve ısı iletiminin yalnızca moleküllerin doğrusal hareketlerinden değil aynı zamanda dönme hareketleri ile de iletiliyor olmasıdır. Poliatomik gazlar için daha uygun ısı iletim katsayıları veren ifadeler geliştirilmiş olmasına rağmen gerçeğe en yakın denklemler ilk olarak Mason ve Monchick analiz yöntemleri ile elde edilmiştirr. Daha sonraları bu analiz üzerine çeşitli çalışmalar yapılmış ve daha başarılı denklemleri geliştirilmiştir. Bu çalışmada da yine genel mantığa uygun olarak kullanım alanı en genel ve doğruluk oranı en iyi olan Chung denklemi tercih edilmiştirr.

Chung ilişkisi şu şekilde ifade edilebilir. (Prausnitz, 1987)

$$\frac{\lambda M'}{\eta C_v} = \frac{3.75\Psi}{C_v / R} \tag{4.17}$$

Burada ;

M' : moleküler ağırlık , g/mol

η : düşük basıncdaki gaz vizkositesi

$$\Psi = 1 + \alpha\{\frac{[0.215 + 0.28288\alpha - 1.061\beta + 0.26665Z]}{[0.6366 + \beta Z + 1.061\alpha\beta]}\}$$

$$\alpha = (C_v / R) - 3/2$$

$$\beta = 0.7862 - 0.7109\omega + 1.3168\omega^2$$

$$Z = 2.0 + 10.5T_r^2$$

β terimi $(f_{int})^{-1}$ üzerinden tanımlanan bir korelasyondur ve yalnızca polar olmayan akışkanlara uygulanır. Polar akışkanlar içinse bu katsayı özel olarak belirlenmelidir. Chung bazı akışkanlar içim bu katsayıyı vermiştir. Eğer özeliği hesaplamak istenene akışkan polar ise ve β katsayısı bilinmiyorsa 0.758 olarak alınabilir. (Prausnitz, 1987)

Z parametresi Mason ve Momnchick'in gerçekleştirdiği analiz de kullandıkları Z_{rot} ile aynı manayı ifade etmektedir. Z nin büyük değerleri için Ψ parametresi şu şekle dönüşür. (Prausnitz, 1987)

$$\Psi = 1 + 0.2665\frac{\alpha}{\beta} \qquad (4.18)$$

4.2.2 Sıvıların Isı İletim Katsayılarının Belirlenmesi

Saf sıvıların ısı iletim katsayılarının belirlenmesi için geliştirilmiş bütün ilişkiler deneysel verilerden yola çıkılarak elde edilmiştir. Bu denklemlerin doğrulukları da sınırlıdır. Normal kaynama noktasının altında sıvıların ısı

iletim katsayıları 0.1 ile 0.17 W/(m.K) arasında olmaktadır. (Prausnitz, 1987)' de bu şekilde elde edilmiş üç farklı denklem değerlendirilmiştir. Bu çalışmada da bu denklemler arasından normal kaynama noktası denklemi sıvıların ısı iletim katsayılarının belirlenmesi için tercih edilmiştir.

Normal kaynama noktası metodu şu şekilde uygulanmaktadır. (Prausnitz, 1987)

$$\lambda_L = \frac{1.11}{M^{1/2}} \frac{[3 + 20(1 - T_r)^{2/3}]}{3 + 20(1 - T_{br})^{2/3}} \qquad (4.19)$$

burada T_r indirgenmiş sıcaklığı, T_{br} ise doyma sıcaklığının indirgenmiş sıcaklığını temsil etmektedir.

4.3 Yüzey Gerilim Katsayısının Belirlenmesi

Yüzey gerilim katsayısı belirleme metotları da ısı iletim metotları gibi deneysel veriler kullanılarak geliştirilmiş denklemlerdir. Prausnitz ve arkadaşları Sıvı ve gazların özelikleri adlı çalışmalarında (Prausnitz, 1987)' de bu denklemler den ikisini incelemişlerdir. Bu çalışmada da bu denklemlerden karşılıklı hal yasası ile elde edilen ilişki kullanılmıştır.

Bu korelasyon Brock ve Bird tarafından geliştirilmiştir (Brock, 1955). Bu korelasyon polar olmayan sıvılar için başarılı sonuçlar vermektedir. (Prausnitz, 1987)

$$\sigma = P_c^{2/3} T_c^{1/3} \left(0.132\alpha_c - 0.279\right)\left(1 - T_r\right)^{11/9} \qquad (4.20)$$

$$\alpha_c = 0.9076\left[1 + \frac{T_{br} \ln(P_c / 1.01325)}{1 - T_{br}}\right] \qquad (4.21)$$

Burada

σ : yüzey gerilim katsayısı , dyn/cm

T_{br} : indirgenmiş kaynama sıcaklığı , K

temsil etmektedir.

Bu iki denklem yüzey gerilimini hesaplamak için kullanılabilir. Yukarıda belirtilen denklem polar olmayan sıvıların yüzey gerilim katsayılarını hesaplamakta başarılı olmaktadır ancak güçlü hidrojen bağı içeren asitler ve alkoller gibi akışkanlar ve kuantum sıvıları (H_2, He, Ne) için pek başarılı değildir. (Prausnitz, 1987)

Polar akışkanların yüzey gerilim katsayılarını elde etmek için kullanılabilecek bir ilişki Stiel faktörü adıyla anılan bir parametre kullanarak daha başarılı sonuçlar

vermektedir (Hakim, 1971). Bu denklem şu şekilde ifade edilmektedir.

$$\sigma = P_c^{2/3} T_c^{1/3} Q_p \left(\frac{1 - T_r}{0.4} \right)^m \tag{4.22}$$

σ : polar sıvı için yüzey gerilim katsayısı dyn/cm

$$Q_p = 0.1560 + 0.365\omega - 1.754\text{X} - 13.57X^2 - 0.506\omega^2 + 1.287\omega\text{X}$$

$$m = 1.210 + 0.5387\omega - 14.61X - 32.07X^2 - 1.656\omega^2 + 22.03\omega X$$

Bu denklemlerde geçen X Stiel faktörü T_r=0.6 da indirgenmiş sıcaklığa karşılık gelen ve P_r(0.6) indirgenmiş basınç değeri için hesaplanır. (Prausnitz, 1987)

$$X = \log P_{vpr}(0.6) + 1.7\omega + 1.552$$

Denklem (4.22) nin güvenirliği konusunda elde fazla veri bulunmamaktadır ancak X değeri bu denklemin geçerliliği üzerinde çok etkilidir ve bu faktörün doğru seçilmesi önemlidir. (Prausnitz, 1987)

5 KİMYASAL DENGENİN HESAPLANMASI

Bir yakıt pili sisteminin ısıl modellenmesi yapılmak istendiğinde karşımıza çıkacak bir başka sorunda sistemin kimyasal dönüşümleri de içermesidir. Sistem belirli bir hidrokarbon yakıtından hidrojenin dönüştürülmesi sürecini içeriyorsa kaçınılmaz bir şekilde yanma reaksiyonlarının modellenmesi gerekmektedir.

Temelde bu tip bir yakıt pili sisteminin en önemli parçası Yakıt Dönüştürücü kısmıdır. Bu kısımda reaksiyonlar gaz fazında ve sürekli akış halinde katalizör yardımı ile gerçekleşir. Böyle bir modellemeyi gerçekleştirmenin iki yolu karşımıza çıkar. Karmaşık yanma reaksiyonu mekanizması belirlenip her bir basamağın reaksiyon kinetiği üzerinden hesaplamaları yapılabilir. Böylece yakıt dönüştürücü sürecinden elde edilecek ürünler belirlenebilir. Ancak bu durumda her bir basamak reaksiyonun kinetik davranışını belirleyen deneysel verilere ihtiyacımız olacaktır. Bu durumda da genel geçer bir modelin uygulanması zorlaşır.

Kimyasal reaksiyonların modellenmesinde kullanabileceğimiz bir başka yaklaşım reaksiyon mekanizmasına bakmasızın sürekli rejimde kimyasal dengeyi kullanarak reaksiyon sonucu oluşacak ürünleri belirlemektir. Ancak burada da reaksiyonun kullanılan

katalizörün yardımı ile sürekli rejim halinde denge haline ulaşması gerekir. Bu tip bir denge modeli, hızlı gerçekleşen reaksiyonlar için uygundur. Dolayısı ile Yakıt pilinin dizi kısmında ve Yakıt Dönüştürücüsünde gerçekleşen yanma reaksiyonları hızlı gerçekleşen reaksiyonlardır. Dolayısı ile kimyasal denge yaklaşımı açısından incelemeye uygundurlar.

5.1 Kimyasal Denge

Yanma reaksiyonları sonucunda oluşacak ürünlerin tespiti için kimyasal denge şartlarının incelenmesi yeterli olacaktır. Kimyasal dengeye ulaşıldığı anda ki kimyasal bileşimin tespiti için takip edilecek iki farklı yöntem bulunmaktadır. Bunların birincisinde daha öncen deneyler ile elde edilmiş reaksiyon denge sabitleri kullanılarak denge halindeki bileşim tespit edilebilir. Ancak bu yöntem genelleştirmeye uygun olmayan ve deneysel veriler ile bizi kısıtlayan bir yöntem olacaktır.

İkinci yöntemde ise denge şartını sağlayan kimyasal bileşim termodinamik özellikler kullanılarak belirlenemeye çalışılır. Burada bizim için daha uygun olan yöntem ikici yöntemdir. Ancak yöntemin uygulanışına geçmeden önce kısaca kimyasal dengenin gerçekleşme şartının incelenmesinde fayda vardır.

Aşağıda ki denkleme göre bir reaksiyonun oluştuğu varsayılsın.

$$aA + bB + ... \rightarrow lL + mM + ... \tag{5.1}$$

veya

$$v_A A + v_B B + ... + v_L L + v_M M + ... = 0 \tag{5.2}$$

Basınç P ve sıcaklık T nin kimyasal reaksiyon boyunca sabit kaldığını düşünüldüğünde. Reaksiyon katsayılarını şu şekilde de ifade edilebilir.

$$\delta n_A = v_A \delta\lambda\,,\ \delta n_B = v_B \delta\lambda\,,\ \ldots,\ \delta n_L = v_L \delta\lambda$$

Sistemin toplam Gibbs serbest enerjisini ise buradan yola çıkarak şu şekilde yazılabilir.

$$\delta G = \sum_A \frac{\partial G}{\partial n_A} \delta n_A = \delta\lambda \sum_A v_A \bar{G}_A \tag{5.3}$$

$$\frac{\delta G}{\delta \lambda} = \Delta \bar{G} \text{ ise}$$

$$\Delta \bar{G} = \sum_A v_A \bar{G}_A = -a\bar{G}_A - b\bar{G}_B - ... \\ +l\bar{G}_L + m\bar{G}_M + ... \tag{5.4}$$

Burada $\Delta\overline{G}$ reaksiyonun serbest enerjisini ifade ederken λ reaksiyonun ilerlemesini bir ölçüsü olarak değerlendirilebilir.

Termodinamiğin ikinci yasasına göre çevre ile ısı alış verişi bulunmayan bir sistemin entropisi her zaman artar.

$$dS_{sys} \geq 0 \tag{5.5}$$

Dolayısı ile adyabatik bir kap içersindeki oluşan bir kimyasal reaksiyonda S_{sys} her zaman artacak yönde reaksiyonu ilerletecektir. Bu değer maksimuma ulaşınca da reaksiyon duracaktır. Bu tanımı çevre ile ısı transferine izin veren bir sisteme uyarlarsak bu kez

$$dS_{sys} \geq \frac{\delta Q}{T} \tag{5.6}$$

olacaktır. Daha önceden bilindiği üzere $\delta Q - PdV = dU$ dir. Bu iki denklemi birleştirirsek

$$dU + PdV - Tds \leq 0 \tag{5.7}$$

Eşitsizliğini elde ederiz. Yeniden Gibbs tanımına dönülürse $G \equiv H - TS$

$$(dG)_{T,P} = dH - TdS - SdT$$
$$= (dU + PdV + VdP) - TdS - SdT \qquad (5.8)$$

Bu denklemde dT ve $dP = 0$ olduğu için

$$(dG)_{T,P} = (dU + PdV) - TdS \qquad (5.9)$$

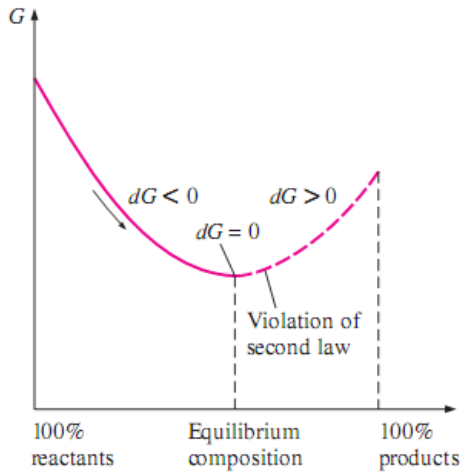
eşitliğine ulaşılabilir. Denklem (5.7) da bize kimyasal reaksiyonun oluşabilmesi için

$$(dG)_{T,P} \leq 0 \qquad (5.10)$$

şartını verir. Dolayısı ile belirli bir sıcaklık ve basınçtaki kimyasal reaksiyon Gibbs enerjisinin azalacağı yönde ilerler ve Gibbs fonksiyonu minimum değerine ulaştığı zaman reaksiyon durur.

$$dG = 0 \qquad (5.11)$$

Bu durum Şekil 5.1 de grafik üzerinde gösterilmiştir.



Şekil 5-1: Reaksiyonun oluşumu sırasında Gibbs enerjisinin değişimi

Bu durumda Denklem (5.4)'nın sıfıra eşit olması gerekmektedir.

$$\Delta G_T = \sum \nu_A G_A^0 = 0 \qquad (5.12)$$

Burada daha öncede adı geçen $g_A^0 = \frac{\partial g_A}{\partial n_A} = \mu_A$ daha önce kimyasal potansiyel olarak adlandırılmış ve açık sistemlerin termodinamik özellikleri arasında belirtilmiştir. Bu özellik kimyasal dengede önemli bir rol oynamaktadır.

Şimdi bu dengenin nasıl hesaplandığını incelenebilir.

5.2 Kimyasal Dengenin Hesaplama Yöntemleri

Bir önceki bölümde ifade edilen kimyasal potansiyel cinsinden bir sistemin Gibbs enerjisi toplamını şu şekilde ifade edilebilir.

$$g = \sum_{j=1}^{NS} \mu_j n_j \tag{5.13}$$

Bu noktadan sonra kolaylık olması açısından Gibbs enerjisi ve kimyasal potansiyeller kmol başına ifade edilecektir.

Denklem (5.13)'de geçen *NS* karışımda bulunan toplam madde sayısını, *g* kmol başına Gibbs serbest enerjisini, μ_j yine kmol başına kimyasal potansiyeli ve n_j ise karışımda ki her bir maddenin mol sayısının toplam karışım kütlesine oranına karşılık gelmektedir.

$$\mu_j = \left(\frac{\partial g}{\partial n_j} \right)_{T,P,n_{i \neq j}} \tag{5.14}$$

Sonuçta kimyasal denge sonucunda oluşacak yeni bileşimi tespit edebilmemiz için Denklem (5.13) ile ifade edilen Gibbs serbest enerjisinin minimum olduğu noktayı tespit etmemi gerekmektedir. Bunun yanında bileşimin

toplam kütlesinin korunumu da sağlanmalıdır. Böyle bir sistem için kütlenin korunumu şu şekilde ifade edilebilir.

$$\sum_{j=1}^{NS} a_{ij} n_j - b_i^0 = 0 \quad (\text{i=1,……,} l\,) \tag{5.15}$$

Buradaki l reaksiyona giren maddelerin içerdikleri elementlerin sayısını temsil etmektedir. b_i^0 reaksiyona giren tüm moleküller içersinde bulunan i'inci elementin miktarlarını ifade etmektedir. a_{ij} ise her j'inci moleküldeki i'inci elementin miktarını temsil etmektedir. Bu şekilde oluşturulan matris sitokometrik matris olarak adlandırılır.

Dolayısı ile matematiksel olarak karşımıza çıkan asıl problem Denklem (5.13) de verilen fonksiyonun Denklem (5.15) de verilen kısıt göz önünde bulundurularak minimize edilmesidir. Problem basit gibi gözükse de μ_j kimyasal potansiyel fonksiyonun non-lineer bir fonksiyon olmasından dolayı bu fonksiyonun minimizasyonu için bazı matematiksel dönüşümler yaparak fonksiyonu lineer hale dönüştürmek gerekmektedir.

Gerçek gazlardan oluşan ideal bir kimyasal sistem için μ_j kimyasal potansiyel şu şekilde ifade edilir.

$$\mu_j = \mu_j^0 + RT \ln \frac{n_j}{n} + RT \ln P \tag{5.16}$$

Bu minimizayonun gerçekleştirilebilmesi için en uygun yol Lagrange çarpanları yöntemini uygulamaktır. (Gordon, 1994).

Bu yöntem Denklem (5.13) ve (5.15) ye uygulandığında şu sonuca ulaşılabilir.

$$G = \sum_{j=1}^{NS} \mu_j n_j + \sum_{i=1}^{l} \lambda_i \sum_{j=1}^{NS} (a_{ij} n_j - b_i^0) \qquad (5.17)$$

Bu denklemin çözümü bize denge halindeki madde bileşimini verecektir. Ancak daha önce bahsedilebilirği gibi bu denklemin çözümü için öncelikle denklemin lineer bir hale sokulması gerekmektedir.

Denklem (5.17)' lineer hale sokmanın bir yolu bu fonksiyonu Taylor serisine açmaktır. Fonksiyon Taylor serisine bilinmeyen ve asıl istediğimiz n_j ve λ_i üzerinden açılabilir. Taylor serine açarak $NS + l$ kadar sayıda lineer denklem elde etmektir. Ancak bu şekilde denklem sisteminin çözümünden n_j ve λ_i nin kendisini değil Δn_j ve $\Delta\lambda_i$ adim değerlerini elde ederiz. Bu adım değerleri de bizi her iterasyonda basamak basamak fonksiyonun minimum noktasına ulaştırır (Gordon, 1994).

Şimdi Gordon tarafından NASA'da yürütülen araştırma sonucu geliştirilen ve bir bilgisayar programı haline getirilen yöntemi incelenebilir.

5.2.1 Gibbs Minimizasyonunda NASA Yöntemi

Gibbs minimizasyonu yöntemlerinin birçoğu bu noktaya kadar aynıdır. Ancak bu noktadan sonra lineer denklem sistemlerinin elde edilişi bakımından çeşitli yöntemler arasında farklılıklar bulunmaktadır. Bunlardan en genel ve kolay uygulanabilir olanı NASA Lewis araştırma merkezinde geliştirilen algoritmadır. Denklem (5.17) da Lagrange katsayıları ile minimizasyonu gerçekleştirebilmek için şu eşitliklerin sağlanması gerekir.

$$\left(\frac{\partial G}{\partial n_j}\right) = \mu_j + \sum_{i=1}^{l} \lambda_i a_{ij} = 0 \qquad (5.18)$$

$$\left(\frac{\partial G}{\partial n_j}\right) = \sum_{j=1}^{NS} (a_{ij} n_j - b_i^0) = 0 \qquad (5.19)$$

Burada $\mu_j = \mu_j^0 + RT \ln \frac{n_j}{n} + RT \ln P$ dir. Bunun yanında yukarıda yaptığımız tanım gereği n_j karışımda ki her bir maddenin mol sayısının toplam karışım kütlesine oranıdır. Dolayısı n_j toplamı bize

$$\frac{1}{M} = n = \sum_{j=1}^{NS} n_j \tag{5.20}$$

İlişkisini verecektir ki burada M karışımın mol kütlesini verir. Şimdi Denklem (5.18)-(5.20) arasındaki eşitliklerde tüm terimler aynı tarafa toplanıp Taylor serisine $\ln n_j$, $\ln n$ ve λ_j üzerinden açılabilir. Elde edilecek denklemler şunlardır.

$$\sum_{j=1}^{NS} a_{kj} n_j \Delta \ln n_j = b_k^0 - b_k \qquad (k=1,\ldots,l) \tag{5.21}$$

$$\Delta \ln n_j - \sum_{i=1}^{l} a_{ij} \pi_i - \Delta \ln n = -\frac{\mu_j}{RT} \qquad (j = 1,\ldots,\mathrm{NG}) \tag{5.22}$$

$$\sum_{j=1}^{NS} n_j \Delta \ln n_j - n \Delta \ln n = n - \sum_{j=1}^{NS} n_j \tag{5.23}$$

Sonuçta NG + l + 1 adet denkleme ve bilinmeyene ulaşılabilir. Ancak çözümü ve minimizasyon için gereken iterasyon sayısını azaltmak için bu denklemleri indirgemek daha uygun olacaktır. İndirgenmiş bu denklemler şu şekilde olacaktır. (Gordon, 1994).

$$\sum_{i=1}^{l}\sum_{j=1}^{NG} a_{kj}a_{ij}n_j\pi_i + \left(\sum_{j=1}^{NG} a_{kj}n_j\right)\Delta \ln n = \\ b_k^0 - b_k + \sum_{j=1}^{NG}\frac{a_{kj}n_j\mu_j}{RT} \qquad (k = 1,...,l) \tag{5.24}$$

$$\sum_{i=1}^{l}\sum_{j=1}^{NG} a_{ij}n_j\pi_j + \left(\sum_{j=1}^{NG} n_j - n\right)\Delta \ln n \\ = n - \sum_{j=1}^{NG} n_j + \sum_{j=1}^{NG}\frac{n_j\pi_j}{RT} \tag{5.25}$$

Denklem (5.24) ve (5.25) kullanılarak minimizasyon gerçekleştirilebilir. Bu denklemler sonucu elde edilen $\ln n_j$, $\ln n$ ve λ_j değerleri aşağıdaki denklemlerde kullanılarak yeni adım için gereken değerler elde edilir. (Gordon, 1994).

$$\ln n_j^{(i+1)} = \ln n_j^{(i)} + \lambda^{(i)}(\Delta \ln n_j)^{(i)} \quad (j = 1,...,NG) \tag{5.26}$$

$$\ln n^{(i+1)} = \ln n^{(i)} + \lambda^{(i)}(\Delta \ln n)^{(i)} \tag{5.27}$$

Bu denklem sistemini belirli bir kimyasal karışım çözerken ilk adım başlangıç tahminlerinin uygun yapılmasıdır. Denklem sistemi karmaşık olsa da temel de basit bir sistem olduğu için basit ve her zaman kullanılabilecek bir başlangıç değeri seçilebilir. Bunun için n = 0.1 seçilebilir bu durumda karışımda ki her bir bileşenin iterasyona başlangıç değeri n_j = 0.1/NG olacaktır.

Yakınsamanın tespiti içinde şu yaklaşım kullanılabilmektedir (Gordon, 1994).

$$\ln \frac{n_j}{n} > 18.420681 \tag{5.28}$$

Ayrıca her iterasyonda adım uzunluğu λ'yı belirli bir algoritma ile belirlemek çözümü hızlandıracak ve doğruluğun artmasına imkân verecektir. λ aşağıda ki gibi belirlenebilir (Gordon, 1994).

$$\lambda_1 = \frac{2}{\max(5\left|\Delta \ln n\right|, \left|\Delta \ln n_j\right|)} \tag{5.29}$$

$$\lambda_2 = \min \left| \frac{-\ln \frac{n_j}{n} - 9.2103404}{\Delta \ln n_j - \Delta \ln n} \right| \tag{5.30}$$

$$\lambda = \min(1, \lambda_1, \lambda_1) \tag{5.31}$$

Bu şekilde birçok sistem en fazla 50 iterasyon sonucunda kimyasal denge halini veren bileşimin elde edilmesi mümkün olmaktadır (Gordon, 1994).

6 GELİŞTİRİLEN PROGRAMLAR

Daha önceki bölümlerde ifade edilen teorik yaklaşımlar kullanılarak öncelikle sistemde yer alan akışkanlar ardından da sistemde kullanılan bileşenler modellenmiştir. Bu çalışma da kullanılan ve geliştirilen program dosyaları Çizelge 6.1 e 6.2 de verilmiştir. Bu çalışmada temle alınan program kodlarının bazıları Yrd. Doç. Dr. M. Turhan Çoban'ın daha önceki konu ile ilgili çalışmalarından alınmıştır. Diğer program kodları bu çalışma sırasında geliştirilmiştir.(Çoban, 2005) Yine bu kodların bazıları kullanılabilirliği ve modülerliği artırmak için daha küçük parçalar halinde bölünmüştür. Bazı program dosyaları bilgisayar programcılığı açısından kodların optimum şekilde çalışabilmesi için esas hesaplamaları gerçekleştiren programlara destek amacı ile geliştirilmiştir ve bunların hesaplamaların kendisi üzerinde bir etkisi bulunmamaktadır.

Çizelge 6-1: Yrd. Doç Dr. M. Turhan Çobanın çalışmalarından ve ders notlarından alınan program kodları

Atom.java	Bir elementin atomsal özelliklerini saklayan sınıf
BasicWindowMonitor .java	MS Windows ortamında arayüzlerin istene şekilde gösterilmesine yardımcı sınıf
complex.java	Kompleks sayılarla gerçekleştirilen işlemleri sağlayan sınıf
Gas1.java	İdeal gazların özelliklerini hesaplayan sınf
Gas1Model.java	İdeal gaz özelliklerini kullanıcıya ulaştıran arayüze yardımcı sınıf
Gas1Table.java	İdeal gaz özelliklerini kullanıcıya ulaştıran arayüz programı
GasData.java	İdeal gaza ait denklem katsayılarını saklayan sınıf
genelModel.java	Diğer sınıflara girdi çıktı fonksiyonlarına ulaşma imkânı veren yardımcı sınıf
girdi.java	Girdi çıktı fonksiyonlarını barındıran sınıf
girdi1.java	Girdi çıktı fonksiyonlarını barındıran sınıfa yardımcı sınıf

Gmix.java	İdeal gaz karışımlarının özelliklerini hesaplayan sınıf
GmixModel.java	İdeal gaz karışımı özelliklerini kullanıcıya sunan arayüze yardımcı sınıf
GmixTable.java	İdeal gaz karışımlarının özelliklerini kullanıcıya sunan arayüz sınıfı
steam.java	Suyun özelliklerini hesaplayan sınıf
Matrix.java	Matrisler üzerindeki işlemleri gerçekleştiren sınıf
Text.java	Dosyadan girdi çıktı sağlayan sınıf

Çizelge 6-2: Bu çalışma sırasında geliştirilen program dosyaları. Buradaki kodların önemli olan bazıları Ek-1 kısmında verilmiştir.

absfluid.java	Tüm akışkanların özelliklerine tek nesne üzerinden ulaşabilmek için yazılmış abstract sınıf
AutoThermalRef.java	Ototermal yakıt dönüştürücü modeli
CalculationException.	Hesaplama sırasında iterasyon sayısının aşılması gibi yanlış sonuçlara sebep olabilecek hataları kullanıcıya bildirmek üzere

java	yazılmış sınıf
CreateRGDB.java	Akışkan özelliklerini modelleyen denklemlere ait katsayıların veritabanının oluşturulması ve bu veri tabanına ekleme yapılmasını sağlayan sınıf
fluidInterface.java	Tüm akışkanların özelliklerine tek nesne üzerinden ulaşabilmek için yazılmış ara sınıf
fluidPoint.java	Tüm akışkanların özelliklerine tek nesne üzerinden ulaşabilmek için yazılmış abstract sınıf
GasAlreadyDefinedException.java	Her hangibir hesaplama sırasında girdi verisi olarak aynı gazın iki veya daha fazla girilmesi durumunda kullanıcıyı uyaran hata sınıfı
GException.java	Tüm hata sınıflarının genel özelliklerini belirleyen hata sınıfı
gibbs.java	Gibbs minimizasyonu yöntemi ile kimyasal dönüşüm sonucunda oluşacak ürünlerin miktarlarını hesaplayan sınıf
gibbsSpecies.java	Gibbs minimizasyonunda kullanılan farklı türdeki akışkanların özelliklerini belirleyen sınıf
HEXfluid.java	Isı değiştiricilerde bulunan akışkanların özelliklerini hesaplayan sınıf

InappropriateArrayException.java	Matematiksel dizi hatalarını kullanıcıya bildiren sınıf
LeeKesler.java	Gerçek akışkanların özelliklerini hesaplayan sınıf
LeeKeslerMix.java	Gerçek akışkan karışımlarının özelliklerini hesaplayan sınıf
LKModel.java	Gerçek akışkanların özelliklerini kullanıcıya bildiren yardımcı sınıf
LKTable.java	Gerçek akışkanların özelliklerini kullanıcıya ileten arayüz programı
LKmixModel.java	Gerçek akışkan karışımlarının özelliklerini kullanıcıya ileten yardımcı program
LKmixTable.java	Gerçek akışkan karışımlarının kullanıcıya ulaşturan arayüz programı
PlateHEX.java	Plakalı ısı değiştiricisini modelleyen program
PlateHEXfluid.java	Plakalı ısı değiştiricisinde bulunan akışkanların özelliklerini hesaplayan program
Pumps.java	Sistemde bulunabilecek pompaların hesaplamalarını gerçekleştiren sınıf

simpleHEX.java	En basit halinde bir ısı değiştiricinin hesaplalamalarını gerçekleştiren sınıf
simplePump.java	Basit pompa hesabını gerçekleştiren sınıf
spDatabase.java	Diğer programların ihtiyaç duyduğu denklem katsayıları ve verileri sağlayan veritabanı erişim programı
spDB.java	Veritabanına erişim programına yardımcı sınıf
SPdbmaker.java	Veritabanını oluşturan sınıfa yardımcı program kodu
steamRef.java	Buharlı yakıt dönüştürücü hesaplamalarını gerçekleştiren sınıf
UnDefinedEntryException.java	Hatalı girilecek bir akışkan için kullanıcıyı uyaracak hata sınıfı
UnDefinedStituationException.java	Diğer hata sınıflarının kapsamına girmeyen program içinde oluşacak hataları kullanıcıya bildiren hata sınıfı

Şimdi bu çalışma kapsamında geliştirilen programlar hakkında daha ayrıntılı bilgi verelim.

6.1 Lee-Kesler

Bu paket gerçek gazların özelliklerini hesaplar programın çalışabilmesi için kullanılan denklemlerin gereksinim duyduğu katsayılar özelliği istenen akışkan için önceden oluşturulmuş veritabanından okunması gerekmektedir. Bu veriler kısaca şunlardır.

Çizelge 6-3: Lee-Kesler programında kullanılan değişkenlerin anlamları.

Özellik	**Açıklaması**
Formula	Akışkanın kimyasal formülü (CH4, CH3NO gibi)
Name	Akışkanın kimyasal adı (fmethan, formamide gibi)
atomList	Molekülün içcerdiği elementleri temsil eden dizi
natom	Atom sayısı
M	Atomik yapıdan hesaplanan moleküler ağırlık
MolWt	Veritabanından okunan moleküler ağırlık, g/mol

Tfp	Normal donma noktası (1 atm), K
Tb	Normal kaynama noktası (1 atm), K
Tc	Kritik sıcaklık, K
Pc	Kritik basınç, bar
Vc	Kritik hacim, cm3/mole
Zc	Kritik sıkıştırılabilirlik faktörü
Omega	Pitzer eksentrik faktörü
Dipm	Dipol moment
CPC	Sabit basınç özgül ısı denklemine ait katsayılar ($a+bT+cT^2+dT^{3)}$
DELHF	Formasyon entalpisi
DELGF	Formasyon Gibbs serbest enerjisi
DELS0	298.2 K, 1.01325 bar daki entropi değeri
Vpeq	Buhar basıncını veren denklem numarası

VPC	Buhar basıncı denklemi katsayıları (a-b/T+cln T+d/T^2)
Tmin	Yukarıda belirtilen buhar basıncı denkleminin doğru bir şekilde uygulanabileceği aralığın alt sınırı
Tmax	Yukarıda belirtilen buhar basıncı denkleminin doğru bir şekilde uygulanabileceği aralığın üst sınırı
LDEN	Sıvı yoğunluğ, g/cm^3
TDEN	Sıvı yoğunluğunun ölçüldüğü sıcaklık, K

Burada geliştirilen programlarda bu değerler otomatik olarak akışkanın seçilmesi ile önceden oluşturulmuş bir veritabanından yardımcı programlar aracılığı ile çağrılmaktadır. Bu veritabanında halihazır da kullanılmak üzere 429 farklı bileşiğe ait veriler bulunmaktadır. Bu veritabanın bir çıktı örneği aşağıdaki gibidir.

Çizelge 6-4: Lee-Kesler programının veritabanında bulunan methanole ait özellikler

Methanol Formula:	CH4O	Methanol Omega=	0.556

Methanol Name:	methanol	Methanol Dipm=	1.7
Methanol natom:	3	Methanol CPC[i]0]=	21.15
atomName[0]=	C	Methanol CPC[i]1]=	0.07092
atomN[0]=	1	Methanol CPC[i]2]=	2.59E-05
atomName[1]=	H	Methanol CPC[i]3]=	-2.85E-08
atomN[1]=	4	Methanol DELHF=	-201300
atomName[2]=	O	Methanol DELGF=	-162600
atomN[2]=	1	Methanol DELS0=	188.956
atom[0]=	C	Methanol VPEq=	1
atom[1]=	H4	Methanol VPC[i]0]=	-8.54796
atom[2]=	O	Methanol VPC[i]1]=	0.76982
MolWt=	32.042	Methanol VPC[i]2]=	-3.1085
Methanol Tfp=	175.5	Methanol VPC[i]3]=	1.54481

Methanol Tb=	337.7	Methanol Tmin=	288
Methanol Tc=	512.6	Methanol Tmax=	512.6
Methanol Pc=	80.9	Methanol LDEN=	0.791
Methanol Vc=	118	Methanol TDEN=	293
Methanol Zc=	0.224		

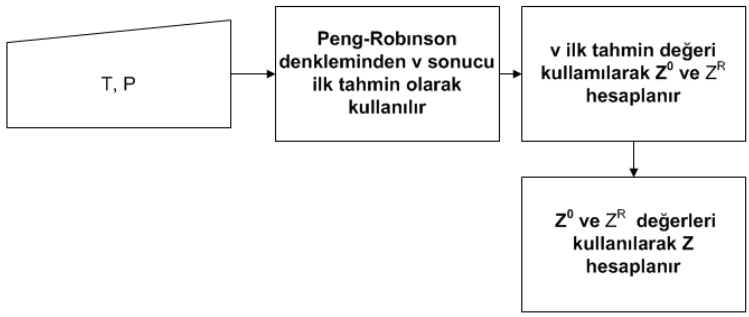
Ayrıca belirli gazlar için daha önce geliştirilmiş olan ideal gaz programından alınan c_p eğrileri kullanılmıştır. Çünkü daha önceki çalışmalarda gazların özgül ısıları için daha hassas eğriler elde edilmiştir.

LeeKesler.java programı gerçek gazların özelliklerini daha önceki bölümlerde anlatılan Lee-Kesler yaklaşımını kullanarak hesaplar. Ancak bu denklemler den *v, T* ve *P* değerleri verildiğinde doğrudan herhangi bir kök bulma yöntemi ile bulunabilir. Fakat bir akışkanın termodinamik hali *T-v, T-s, T-h, P-h* gibi *T, P, h, s* in kombinasyonları şeklinde ifade edilebilir. Bunların en yaygınları *T-v, T-s, T-h, P-h* dır. Bu durumda bu bilinen Lee-Kesler denklemi üzerinden tekrar bir kök bulma işlemi yapılır ve istenen noktadaki diğer termodinamik özellikler hesaplanır. Burada kök bulma yöntemi olarak paranteze alma ve Ridder (Ek-2

de bu yöntemle ilgili ayrıntılar verilmiştir)yöntemleri birlikte kullanılmıştır. Bu sayede nonlineer bu denklemin kökleri kolayca hesaplanabilmektedir.

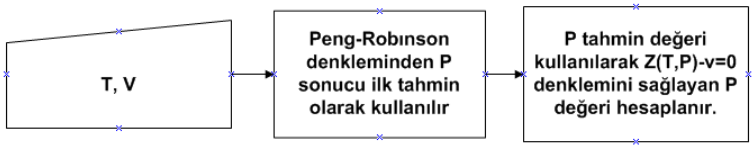
Benzer durum diğer termodinamik özellikler içinde geçerlidir. Aynı hesaplama şekli bu özelliklerde de kullanılmıştır.

Örnek olarak sıkıştırılabilirliğin hesaplanması için genel olarak şu algoritma takip edilir.



Şekil 6-1: Sıkıştırılabilirlik katsayısının hesaplanması.

Eğer başlangıçta *T, v* değerleri verilir ise bu kez yukarıda verilen algoritma bir *Z(T,P)* fonksiyonu gibi düşünülüp bilinene *v* değeri kullanılarak *Z(T,P)*-*v*=0 denklemini sağlayacak *P* değeri hesaplanır.



Şekil 6-2: :Farklı girdiler ile *Z* nin hesaplanması

Z(T,P) hesaplanırken ayrı ayrı Z^0 ve Z^R sıkıştırılabilirlik oranları Denklem (3.49) dan hesaplanırken yine $V_r^{(0)}$ ve $V_r^{(R)}$ nin bir kök bulma yöntemi kullanılarak bulunması gerekir. Detaylar Ekler bölümünde yer almak üzere Parenteze alam ve Ridder kök bulma yöntemleri kullanılarak *Z* değerleri hesaplanır.

Benzer şekilde girdi olarak *T, P* değerleri kullanılarak Denklem (3.75) ile *H,* Denklem (3.77) ile *S* ve bu iki değerden de Gibbs ve Helmholtz serbest enerji değerleri hesaplanabilir.

Denklem (3.102) ile Denklem (3.103) kullanılarak buhar basıncı hesaplanmaktadır.

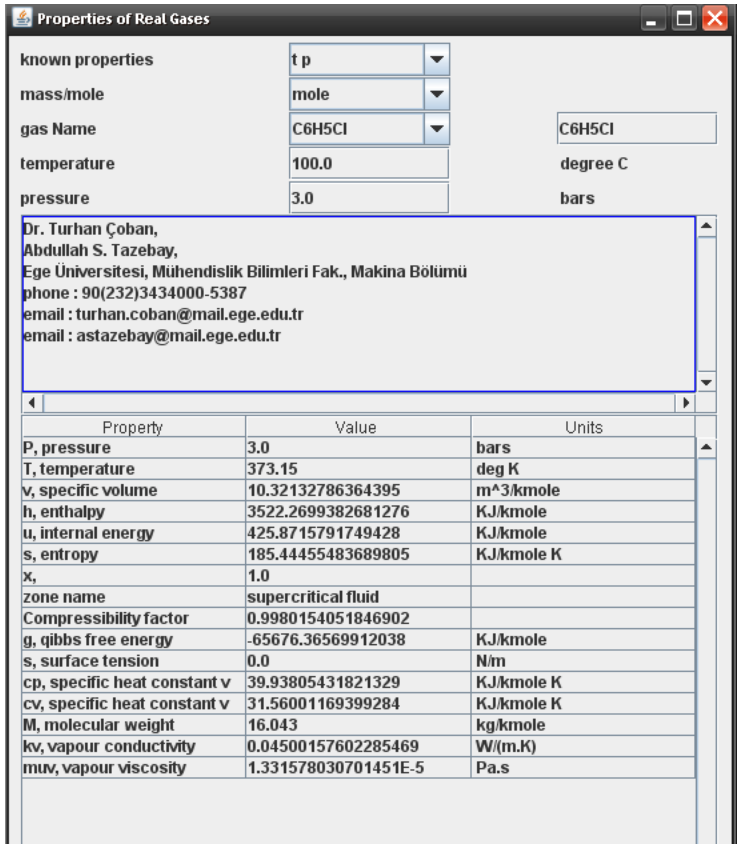
Bölüm 4 de belirtilen denklemler ile de sıvı gaz viskoziteleri, ısı iletim katsayıları ve yüzey gerilim katsayısı belirlenmektedir.

6.2 Lee-Kesler Arayüz Programı

Lee-Kesler program kodu gerçek bir gazın özelliklerini daha önce anlatılan yaklaşım tekniklerini kullanarak hesaplamaya çalışır ve iki farklı sınıfın yardımı ile de bu hesaplamalar kullanıcıya bir arayüz aracılığı ile aktarılır. Bunu gerçekleştiren sınıflar LKModel.java ve LKTable.java dosyalarında bulunmaktadır.

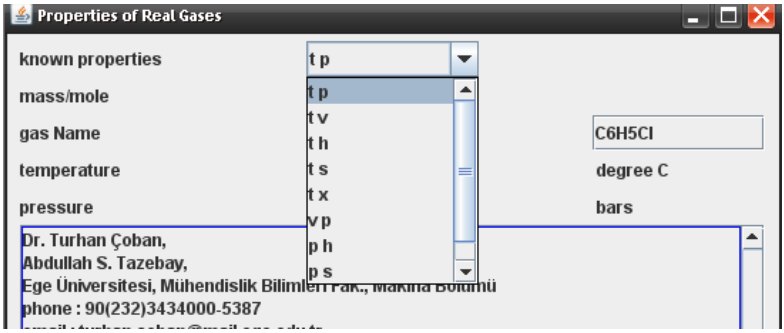
Daha önce belirttiğimiz gibi temel hesaplama tekniği girilen T ve P değerleri kullanılarak ilk olarak v daha sonra da diğer tüm özellikler belirlenir. Böyle elde edilmiş sonuçlar kullanıcı arayüzü vasıtası ile aşağıdaki şekilde ki gibi kullanıcıya aktarılır.

Örneğin C_6H_5Cl (chlorobenzene)'nin T=100 ^{0}C ve P= 3.0 bar da ki özellikleri şu şekilde olacaktır.



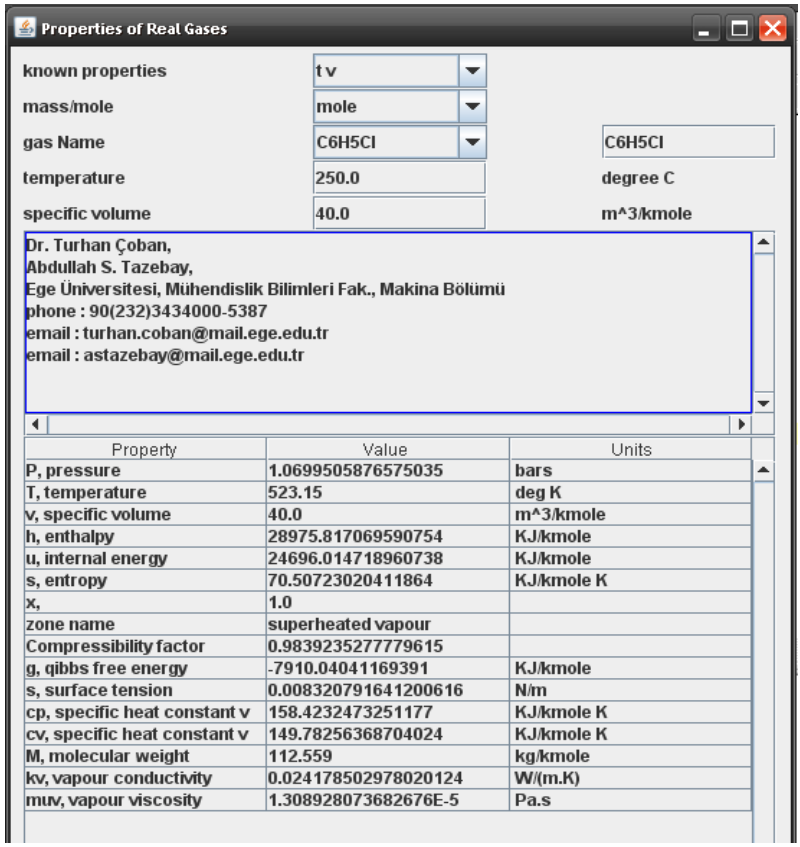
Şekil 6-3: Lee-Kesler programı arayüz örneği

Ancak her zaman elimizde ki bilinen değerler T ve P olmaz bazı durumlarda yalnızca T ve v veya T ve h ile diğer özellikleri belirlememiz gerekebilir. Bir önceki kısımda ki gibi burada gerçekleştirilen hesaplamaların sonuçları da aynı şekilde kullanıcıya sunulmaktadır.



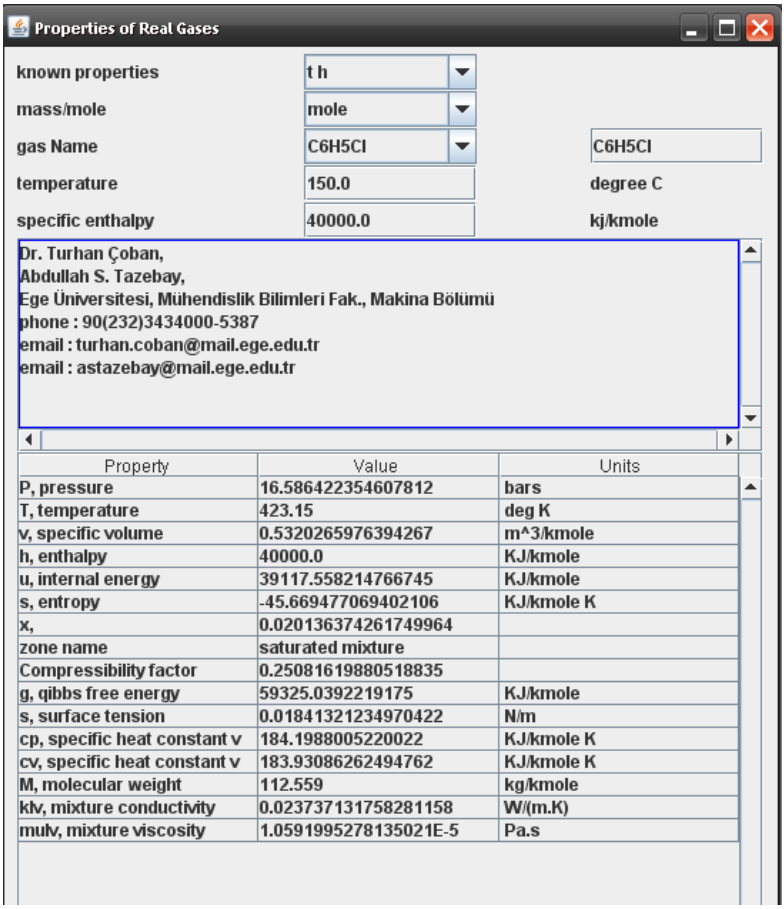
Şekil 6-4: Lee-Kesler programı hesaplama seçenekleri

Örneğin T= 250 ^{0}C ve v=40 m^3/kmole



Şekil 6-5: *T* ve *v* den yola çıkarak özelliklerin belirlenmesi.

Benzer şekilde bu kez T=150 ^{0}C ve h= 40000 kj/kmole için diğer özellikleri belirleyebiliriz.



Şekil 6-6: *T* ve *h* dan yola çıkarak diğer özelliklerin belirlenmesi.

Örneğin yukarıdaki son örnekte görüldüğü gibi bu kez akışkanımız doymuş sıvı buhar karışımında ve x=0.02 kuruluk derecesine sahiptir.

6.3 Lee-Kesler Karışım Programı

Yukarıda özellikleri anlatılan Lee-Kesler programı yalnızca saf akışkanların özelliklerini belirleyebilmektedir. Oysaki bizim sistemlerimizde kullanacağımız akışkanlar çoğunlukla farklı hidrokarbonların veya yanma ürünlerinin karışımı şekilde olacaktır. Bu yüzden saf maddelerden çok karışımların özelliklerini belirleyebilmek bizim için daha önemlidir.

Burada karşımıza çıkan sorun ise karışımdaki maddelerin özelliklerinin birbirlerini etkilemesidir. Bu etkinin göz önüne alınması modelleme açısından sorunlara yol açmaktadır. Ayrıca bu konu üzerine yeterli çalışmalarda yapılmamıştır. Bu yüzden bu etkiyi göz ardı etmek ve karışımı ideal kabul edip özelliklerini bu şekilde belirlemek bizim için daha uygun olacaktır (Prausnitz, 1987).

Bu durumda termofiziksel ve termodinamik özellikler karışımlar için şu genel şekilde belirlenmiştir.

$$x_i = \frac{n_i}{n_t} \tag{5.32}$$

Olmak üzere ve L herhangi bir özelliği ifade etmek üzere karışımın özelliği L_{mix} aşağıdaki gibi olacaktır.

$$L_{mix} = \sum_{i=1}^{NS} x_i L_i \tag{5.33}$$

Bu yaklaşım özellikle yüzey gerilimi gibi bazı termofiziksel özelliklerin belirlenmesinde teorik ve pratik hatalara yol açmaktadır ancak şu anda mevcut yöntemler bu çalışmadaki gibi bir modellememin temel ilkelerine uymadığı için karışım özelliklerinin ideal farz edilmesi daha uygun olmaktadır.

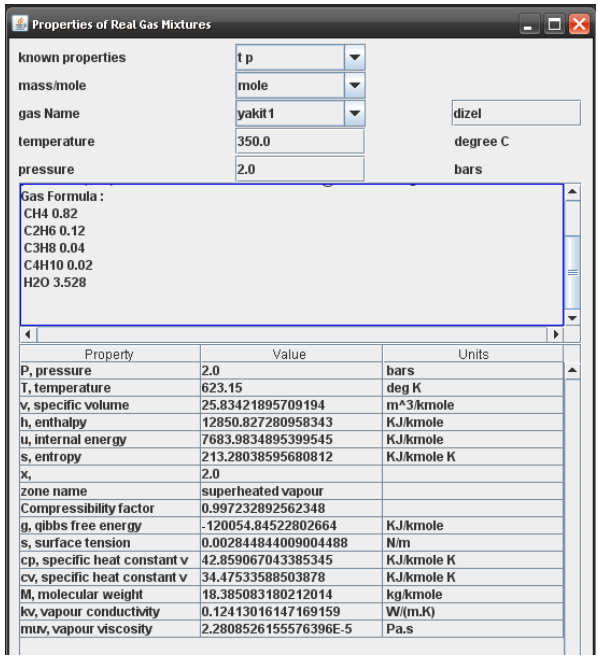
Lee-Kesler karışım programının kullanılabilmesi için karışımı oluşturan maddelerin önceden program çalıştırılmadan RealGasMix.txt metin dosyasına yazılması ve dosyanın kaydedilmesi gerekmektedir. Bu şekilde dosyaya girilmiş bir karışım aşağıdaki şekilde görülmektedir.

```
 95 dizel_5
 96 7
 97 C9H20  0.01715
 98 C10H22 0.03079
 99 C11H24 0.05964
100 C12H26 0.09655
101 C13H28 0.1960
102 C14H30 0.1780
103 C15H32 0.08319
104
105 air
106 3
107 nitrogene 0.7808780878
108 oxygen 0.2095209521
109 carbondioxide 3.00030003e-4
```

Şekil 6-7: Lee-Kesler karışım programında karışımın programa girilmesi.

Burada görüldüğü gibi öncelikle ilk satıra karışımın adı daha sonraki satıra karışımın içerdiği madde sayısı bundan sonra da sırası ile maddelerin kimyasal formülleri ve karışım içindeki mol sayıları girilmelidir. Bu şekilde aşağıdaki yakıt için yapılan giriş ve bu yakıtın T=350 ^{0}C ve P=2 bar daki özellikleri görülebilir.

```
1 yakit1
2 5
3 CH4 0.82
4 C2H6 0.12
5 C3H8 0.04
6 C4H10 0.02
7 H2O 3.528
8
```



Şekil 6-8: Lee-Kesler karışımda bir örnek çıktı.

6.4 Gibbs Programı

gibbs.java programı bir önceki bölümde anlatılan kimyasal reaksiyon hesabını gerçekleştiren program kodudur. Bu kod Denklem (5.24) ile Denklem (5.25) deki denklem sistemlerini çözerek kimyasal reaksiyon sonucu oluşacak ürünleri ve mol sayılarını hesaplar.

Bu program denklem sistemini çözerken Matrix.java programında ifade edilen LU[1] yöntemini kullanmaktadır.

Bu program ile kimyasal dengenin oluşması istenen sıcaklık verilip çıkış mol sayıları ve reaksiyonun entalpisi elde edilebilir. Bunun yanında istendiği takdirde farklı sıcaklık değerleri için çıkış denge halleri alınabilir. Üçüncü seçenekte ise reaksiyona verilecek veya reaksiyondan çekilecek ısı miktarı verilerek denge sonunda ulaşılacak mol sayıları elde edilebilir.

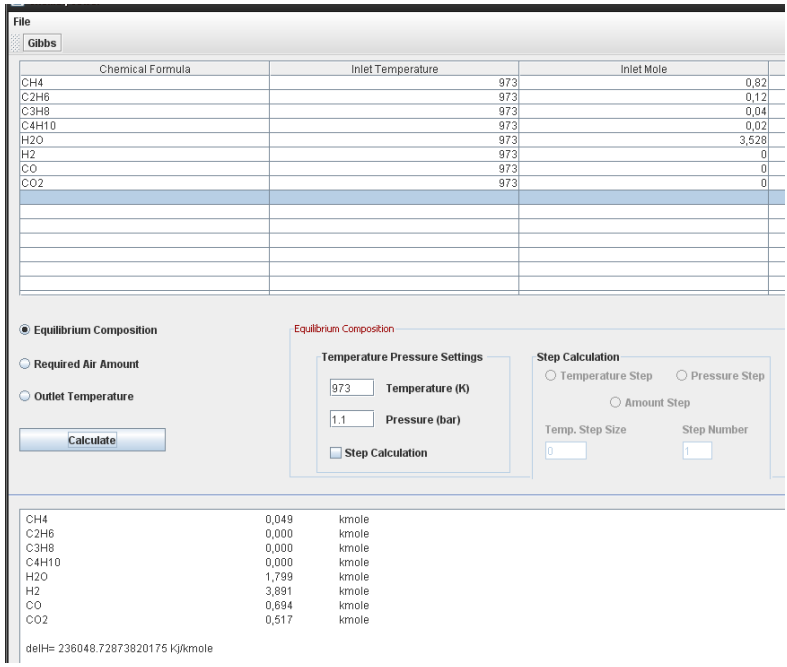
Örnek olarak aşağıdaki kimyasal karışım bir yanma odasına gönderilerek sonuçta çıkacak ürünler tespit edilebilir.

CH_4 0.82 C_2H_6 0.12e
C_3H_8 0.04 C_4H_{10} 0.02

[1] LU yöntemi Ek-2 kısmında detaylı olarak anlatılmıştır.

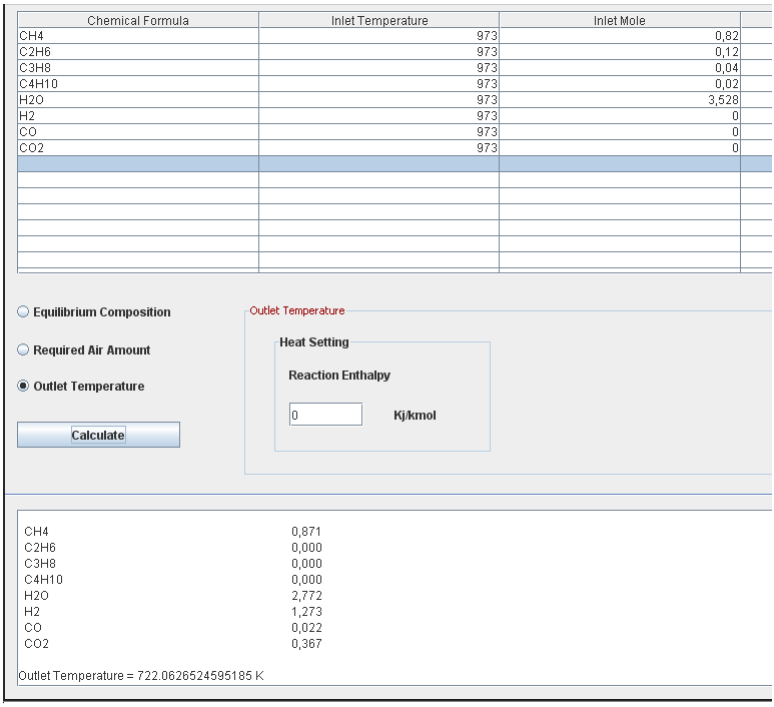
H_2O 3.528 H_2 0.0
CO 0.0 CO_2 0.0

Burada dikkat edilmesi gereken konu reaksiyondan çıkması muhtemel tüm maddelerin de başlangıçta 0.0 mol sayısı ile girenler kısmında gösterilmesi gerektiğidir. Ancak bu şekilde hesaplamak yapmak mümkündür. Şimdi bu örnek sonucunda çıkış mol sayılarını program aracılığı ile tespit edilir. Denge sıcaklığımızı 973 K verilir.



Şekil 6-9: Kimyasal Denge hesaplama programının arayüzü ve elde ettiği sonuçlar.

Şimdi aynı şartlar altında bu kez reaksiyona hiç enerji verilmez ise reaksiyon sonucunun nasıl olacağı incelenebilir.



Şekil 6-10: Kimyasal denge hesabı programının çıkış sıcaklığının hesaplaması

Görüldüğü gibi bu durumda karışımın sıcaklık çıkışı 722 K ne düşmektedir. Ayrıca reaksiyonun ısının yetersizliğinden dolayı C_2H_6, C3H8, C_4H_{10} gibi büyük moleküller tamamen parçalanamamakta ve CH_4 miktarı bu yüzden artmaktadır.

Gibbs.java programı bundan sonra kullanılacak olan ototermal ve buharlı dönüştürücü programları için temel oluşturmaktadır.

6.5 Buharlı Yakıt Dönüştürücü Programı

Bir önceki kısımda bahsedilen Gibbs programı geliştirildikten sonra bir buharlı yakıt dönüştürücü içindeki reaksiyonları modellemek kolaylaşmaktadır. Bir buharlı yakıt dönüştürücünün içinde gerçekleşen reaksiyonlar aşağıdaki gibidir.

$$YAKIT + Su\ Buharı + ısı \longleftrightarrow H_2 + CO_2 + H_2O +$$

Diğer yanma ürünleri

Bu reaksiyon endo termik olduğu için yakıt su buharı karışımının reaksiyon boyunca ısı alması veya her ikisinin ayrı ayrı bir ön ısıtıcı ile çıkış da istenen sıcaklığa ulaşılacak şekilde yüksek sıcaklığa ısıtılması gerekmektedir. Geliştirilen model ile şu hesaplamalar gerçekleştirebilmektedir.

Ürünlerin istenilen sıcaklık da yakıt dönüştürücüyü terk edebilmeleri için verilmesi gereken ısı miktarını tespit etmek mümkündür.

Belirli bir sıcaklık da yakıt dönüştürücüye giren belirli yakıt su buharı karışımının hangi sıcaklıkta yakıt dönüştürücüden çıkacağını belirlemek mümkün olmaktadır.

Örnek bir buharlı yakıt dönüştürücü girişi ve yazılan programdan elde edilen çıkış molleri aşağıda verilmiştir.

Giriş

CH_4:0.92 kmol , N_2:0.75 kmol, C_2H_6:0.32 kmol, C_3H_8:0.1 kmol, C_4H_{10}:0.04 kmol

T=500 ^{0}C de reaksiyonun oluştuğunu ve girenlerinde bu sıcaklıkta olduklarını düşünüldüğünde ayrıca P=1.1 bar sonuç şu şekilde elde edilmektedir.

Çıkış CH_4:1.8342815765018918 kmol, N_2:0.750 kmol, C_2H_6:1.378314E-6 kmol, C_3H_8:7.21358E-12 kmol, C_4H_{10}:4.071974E-147 kmol, H_2O:4.628599334 kmol, H_2:0.102833 kmol, CO:3.066804E-5 kmol, CO_2:0.185684 kmol ve reaksiyonun denge sıcaklığını sabit tutabilmek için gereken ısı miktarı Q=9228.632 kj/kmol olarak çıkmaktadır.

Bu buharlı bir yakıt dönüştürücü için tipik bir gereksinimdir ve daha yüksek sıcaklıklara ihtiyaç duyulduğu açıktır. Bu program arasılığı ile sisteme girecek yakıt miktarı ve buhar miktarının tüm sistem içinde

optimizasyonunu yapmak mümkündür zira sisteme girecek buhar miktarı ancak bir yere kadar tüm sistem verimini olumlu etkileyecektir belirli bir değerden sonra sisteme yüksek miktarda buhar sağlamak verimi olumsuz yönde etkileyecektir. Bu program aracılığı ile buhar miktarına karşılık yakıt dönüştürücünün H_2 çıkışları rahatlıkla optimize edilebilir.

6.6 Ototermal Yakıt Dönüştürücü Programı

Ototermal yakıt dönüştürücü yakıt pillerinde sıkça rastlanılan bir başka hidrojen üretim tekniğidir ve buharlı yakıt dönüştürücülere oranla daha yaygın olarak tercih edilmektedirler. Bu yakıt dönüştürücülerde şu kimyasal reaksiyonlar gerçekleşir.

$$YAKIT + Su\ Buharı + Hava \longleftrightarrow H_2 + CO_2 + H_2O + Diğer\ yanma\ ürünleri$$

Bu yakıt dönüştürücü modeli ise dönüşüm için gereken ısıyı yakıtın bir kısmını hava ile yakarak elde etmektedir. Dolayısı ile bir ön ısıtma işlemine gerek kalmamaktadır. Bu model ile de şu hesaplamalar yapılabilmektedir.

Belirli bir yakıt hava su buharı karışımı için belirli sıcaklık da ki çıkış molleri hesaplanabilmektedir.

Belirli bir çıkış sıcaklığı yakalayabilmek içi gereken yakıt/hava oranının tespit edilmesi mümkündür.

Reaksiyon sırasında oluşabilecek + veya – herhangi bir ısı geçişi urumda çıkış değerlerini tespit etmek mümkündür.

Şimdi bir ototermal yakıt dönüştürtücü çıkışını hesaplayalım.

Giriş CH_4:0.92, N_2:0.929, C_2H_6:0.32, C_3H_8:0.1, C_4H_{10}:0.04, H_2O:3.0, O_2:0.05, CO:0.0, CO_2:0.0, H_2:0.0

Yakıt ve hava giriş sıcaklıkları 723 K dir.

Çıkış CH_4:1.1866E-6, N_2:0.9292, C_2H_6:2.5147E-77, C_3H_8:1.8757E-163, C_4H_{10}:1.0827E-225, H_2O:0.9812, O_2:4.238E-10, CO:1.9213, CO_2:0.09834, H_2:5.418

Çıkış sıcaklığımız ise 1768 K yükselmektedir ve çıkış için herhangi bir ısı gerekmemektedir.

Bunun yanında belirli bir sıcaklık çıkışı yani denge sıcaklığı için sisteme beslenmesi gereken O_2 miktarını da belirlemek mümkündür.

Örneğin aynı giriş değerlerini kullanalım ve çıkışta 750 ^{0}C ulaşmak isteyelim bu durumda ihtiyacımız olan O_2 miktarını kolayca programdan hesaplayabiliriz. Çıkış şu şekilde olmaktadır.

CH4:0.00714, N2:3.7877378, C2H6:2.924757E-9, C_3H_8:9.9655E-93, C_4H_{10} 4.473784E-171, H_2O:2.1937, O_2:1.40047E-189, H_2:4.19193, CO:1.20592, CO_2:0.8069

Bu durumda gereken O_2 miktarı da 1.006 kmol olacaktır.

Bu tip hesaplamaların bu program aracılığı ile yapılması belli bir sistem için en uygun optimum çıkış değerlerinin elde edilmesini sağlayacaktır.

6.7 Pompaların Modellenmesi

Sistemdeki pompalar modellenirken santrifuj bir pompaya ait eğriler kullanıldı. Ancak istendiği takdirde dışarıdan herhangi bir pompaya ait karakteristik eğriler de hesaplamalar için kullanılabilmektedir.

Bu şekilde pompa karakteristiğini veren eğriler kullanılarak sistemde istenen herhangi bir debi miktarı için pompanın çekeceği güç, pompanın verimi ve pompanın sağlayacağı basınç hesaplanabilmektedir.

Örneğin bu şekilde yapılmış bir hesaplama şu şekilde olmaktadır. Giriş değerleri olarak pompalanacak sıvının cinsi ve gereken debi miktarı girildiğinde sonuçta pompanın sağlayacağı basınç ve pompa verimini hesaplamak mümkün olmaktadır. Su akışkan olarak seçilir ve debi miktarı 480 m^3/h alınırsa çıkış basıncı 3.63 bar ve pompanın çalışma verimi de 0.60 olmaktadır. Eğer su yerine metanol kullanılırsa bu kez basılacak yükseklik 3.02 bar ve pompa verimi de 0.56'ya düşmektedir.

6.8 Isı Değiştirici Modellemesi

Isı değiştiricileri modellenirken plakalı ısı değiştirici modeli örnek alınmıştır. Çünkü sistemde bulunan ısı değiştiricilerinin çoğunluğu plakalı gaz-gaz ısı değiştiricilerdir. Bu ısı değiştiricilerin adım adım hesaplanması şu şekilde gerçekleştirilmiştir.

Isı değiştirici modeli belirli sıcak ve soğuk giriş değerleri ve çıkışta istenen sıcak soğuk arasındaki sıcaklık farkı için gereken ısı değiştirici alanını veya uygun bir ısı değişim yüzeyi için çıkışta ulaşılacak sıcak ve soğuk akışkan sıcaklıklarını elde etmek mümkündür.

Örnek olarak giriş sıcak akışkan 200^0C soğuk 150 ^{0}C de olsun çıkışta da 10 ^{0}C lik bir sıcaklık farkına kadar ısıtma

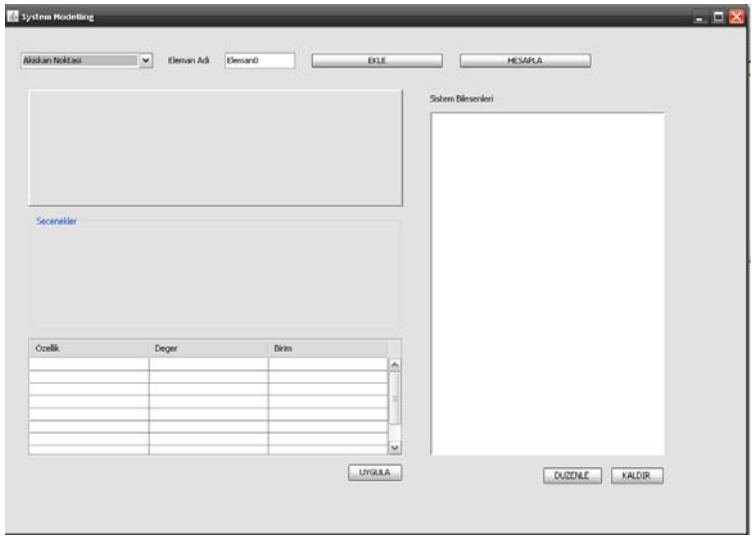
gerçekleştirdiğimizi varsayılsın bu durum da paralel akış da bize gereken ısı değiştirici alanı 10 m^2 olmaktadır.

6.9 Tüm Sistemin Birlikte Modellenmesi

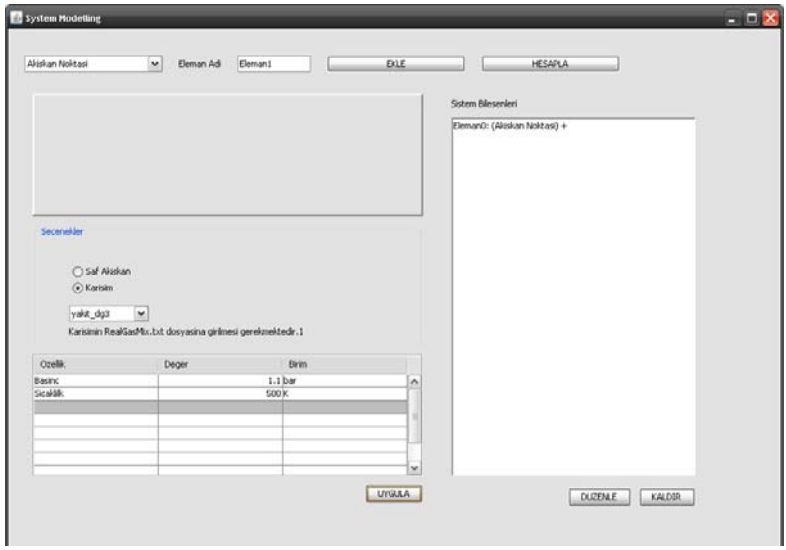
Sistem bileşenleri ayrı ayrı modellendikten sonra bu bileşenleri birlikte birbirine bağlayarak hesaplamayı gerçekleştirebilme imkânı doğar. Bu çalışmada da bu tip bir program geliştirilmiştir. Bu program sistem içinde bulunan bileşenlere tek bir ara yüz kullanarak ulaşmaya imkân sağlamıştır. Bu program aracılığı ile dizayn aşamasında arda arda benzer türden karmaşık termo-kimyasal hesaplamalar gerçekleştirilebilmektedir.

Aşağıda Şekil 6-11'de bu programa ait ara yüz görülmektedir. Bu program System Modeling olarak adlandırılmıştır. Şekilde görüldüğü gibi bu programda çeşitli elemanlar sisteme elenebilmekte ve bu bileşenlerin çıktı ve girdi noktaları Akışkan Noktası adı verilen ve bir akışkanın kimyasal bileşimini, sıcaklığını ve basıncını barındıran bir başka eleman tarafından birleştirilmektedir.

Akışkan noktasına ait özellikler ve bu özelliklerin nasıl programa girildiği Şekil 6-12 de görülmektedir.



Şekil 6-11: Sistemin tamamını modelleyen program ara yüzü



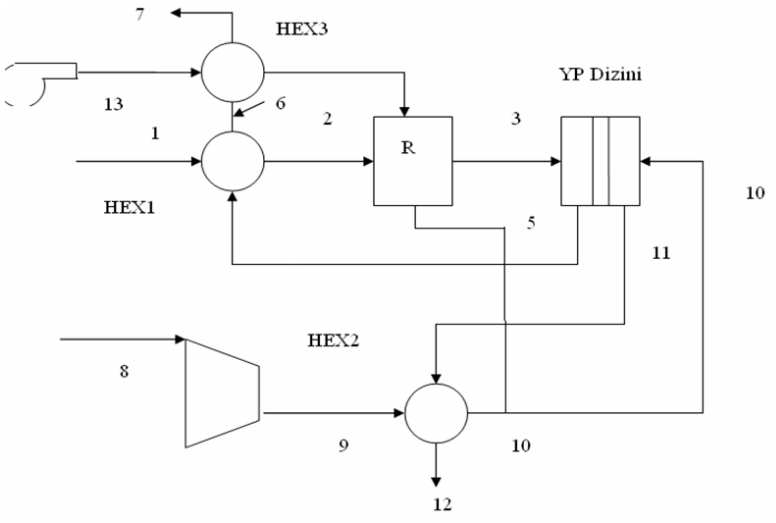
Şekil 6-12: Akışkan Noktası elemanın özellikleri

Bu şekilde farklı bileşenler sisteme eklenebilmekte ve sistem bir kere de çözülebilmektedir.

6.10 Örnek Bir Sistemin Benzeşimi

Şekil 6-13 de görüldüğü gibi bir sistemin termo-kimyasal analizini burada geliştirilen programlar ile yapmak mümkündür.

Şekilde görüldüğü gibi sistemde üç adet ısı değiştirici, bir adet ototermal yakıt dönüştürücü ve yakıt pili modülü içermektedir ve bu bileşenler programa arayüz aracılığı ile girilerek sistemdeki akışkanın sahip olacağı bileşimi ve çeşitli noktalardaki sıcaklığı tespit edilebilir.



Şekil 6-13:Sistem şeması

Ancak burada çözüme uygun bir noktadan başlamak gerekmektedir. Burada sistemin çalışma sıcaklığını esas olarak belirleyen kısım yakıt pilinin kendisidir. Yakıt pilinin çıkışlarının hesaplanması ile tüm sistemde kullanılan ve ısı geri kazanımını sağlayan ısı değiştiricilerin gerekli büyüklükleri elde edilebilir.

Yakıt pilinin çıkış değerlerinin bilinebilmesi içine öncelikle yakıt dönüştürücünün çıkışının tespit edilmesi gerekmektedir.

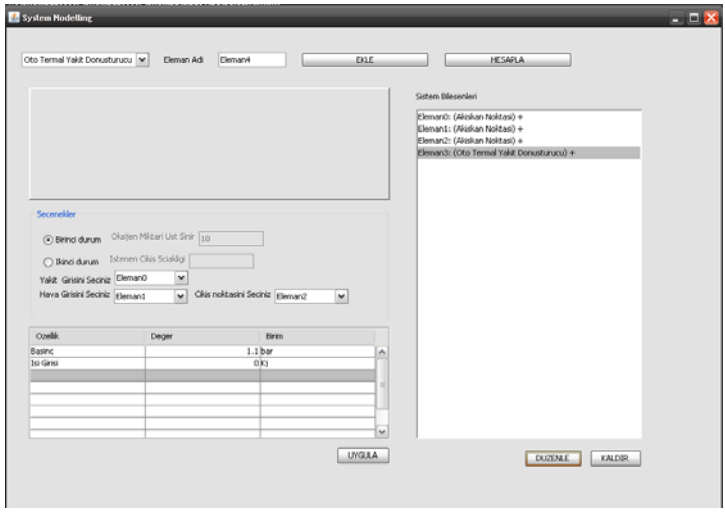
Yakıt dönüştürücüye ulaşan yakıt (2. nokta) doğal gazı temsil etmektedir. Bu doğalgazın bileşimi CH_4 :0.92, C_2H_6

:0.32, C_3H_8 :0.1, C_4H_{10} :0.04, N_2 :0.75 olarak alınabilir. Bu bileşimin program tarafından kullanılabilmesi için RealGasMix.txt dosyasına program çalıştırılmadan önce girilmesi gerekmektedir. Bu durumda dosya şu şekilde görülecektir.

```
1  yakit_dogalgaz
2  6
3  CH4 0.92
4  C2H6 0.32
5  C3H8 0.1
6  C4H10 0.04
7  N2 0.75
8  H2O 3.528
9
10 hava_dg
11 2
12 N2 0.359
13 O2 0.1
```

Şekil 6-14: Yakıt ve Hava bileşenlerinin dosyaya girilmesi.

Burada görüldüğü gibi doğalgaz yakıtı ve hava dosyaya program çalıştırılmadan önce girilmelidir. Daha sonra program çalıştırılarak bu noktalar yakıt dönüştürücü modeline girilir.



Şekil 6-15: Ototermal yakıt dönüştürücü modelinin girilmesi.

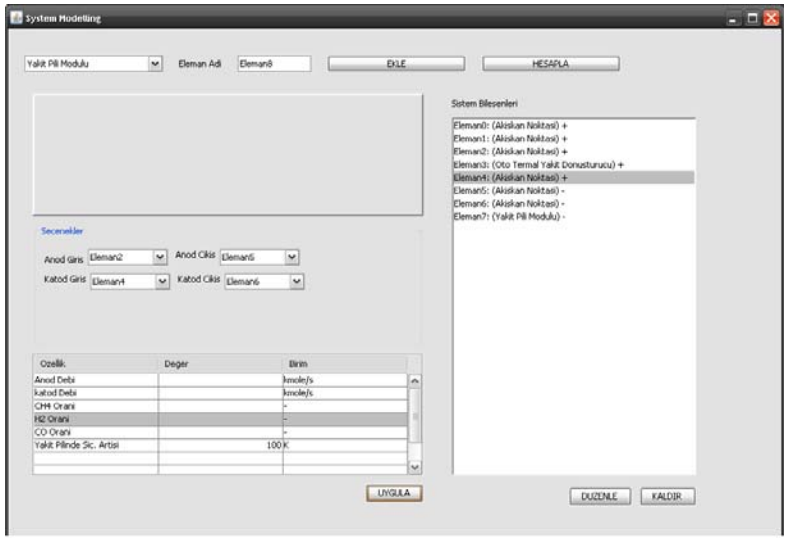
```
================================================================================
Ototermal Reformer

Reaksiyona Girenler
N2:    2.5454545454545454 kmole   T= 700.0 K
CH4:    0.92 kmole   T= 700.0 K
C2H6:    0.32 kmole   T= 700.0 K
C3H8:    0.1 kmole   T= 700.0 K
C4H10:    0.04 kmole   T= 700.0 K
H2O:    3.0 kmole   T= 700.0 K
O2:    0.5 kmole   T= 700.0 K
H2:    0.0 kmole   T= 298.15 K
CO:    0.0 kmole   T= 298.15 K
CO2:    0.0 kmole   T= 298.15 K


Ototermal Reformer Cikisi
N2:  2.545454545454547 kmole
CH4:  0.7629401057238645 kmole
C2H6:  6.385397800617426E-6 kmole
C3H8:  1.7258018411195108E-10 kmole
C4H10:  4.683492473620439E-135 kmole
H2O:  1.8869158823122343 kmole
O2:  1.8966069020535353E-292 kmole
H2:  2.987184749356269 kmole
CO:  0.4010101282378195 kmole
CO2:  0.8560369947249811 kmole
Urunlerin Cikis Sicakligi:   830.2896679288094 K
Ototermal Yakit Donusumu Reaksiyon Isi Girdisi: -0.004825037496630102  kj
```

Şekil 6-16:Ototermal yakıt dönüştürücünün çıkışı.

Şekil 6-16 da görüldüğü gibi yakıt dönüştürücünün çıkışında 2.9 kmol H_2 ve diğer bileşenler elde edilmektedir. Çıkıştaki sıcaklığımız ise 830 K olmaktadır. Giriş sıcaklığımız 700 K olduğuna göre şimdi bu yakıt suyun bu sıcaklığa çıkarılması gerekmektedir.



Şekil 6-17:Yakıt pili ile yakıt dönüştürücünün modellenmesi

Yakıt dönüştürücünün çıkışı ise şu şekilde olacaktır.

```
 1
 2
 3 ============================================
 4 Anod Giris
 5 H2    2.9871
 6 CO    0.40101
 7 CH4    0.0339
 8 CO2    1.1206
 9 H2O    5.1321
10 Katod Giris
11 N2    10.9
12 O2    2.79
13 CO2    0.0042
14 Yakit Pili Katod Cikisi:
15 N2   10.9
16 O2   2.4853475
17 CO2   0.0042
18 Anod cikisi
19 H2   0.08815500000000004
20 CO   0.015780000000000002
21 CH4   0.028815
22 CO2   1.215105
23 H2O   5.641815
```

Şekil 6-18: Yakıt pili çıkışı

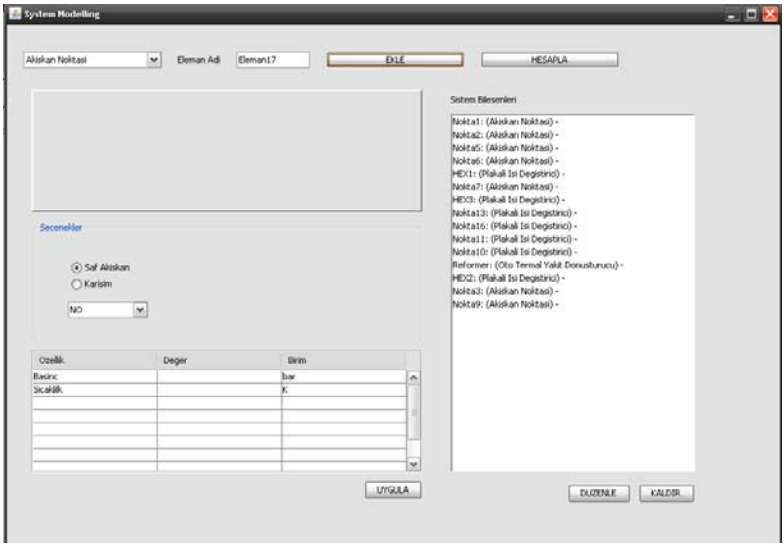
Benzer şekilde bu kez ısı değiştiriciler sırası ile sisteme girilebilir. Birinci ısı değiştirici önek olarak hesaplanır ise

```
839 ++++++++++++++++++++++++++++++++++59 . adim++++++++++++++++++++++++++++++++++++++++++++++++++++
840 fcoldx.T= 644.7894866866061 fcoldx.h= 18051.12413055174 fhotx.T= 941.1529723375845 fhotx.h= 20036.81730510584
841 Isi transferi miktari= 1332245.943944207 J/kg
842 fhotx.h= 20036.81730510584  q*a*delx/fhot.m*fhotx.Mass= 436.4437712361222
843 fcoldx.Mass*q*a*delx/fcold.m/1000= 436.4437712361223 fcoldx.h= 18051.12413055174 T= 941.1529723375845
844 fcoldx.T= 652.2670609580745 fcoldx.h= 18487.567901752675 fhotx.T= 954.4196032667528 fhotx.h= 20473.261076340958
845 Toplam uzunluk x= 0.5900000000000003
846 q= 1332245.943944207
847 (fhot.T-fhotx.T)/fhot.T>0.01= 0.019247183613263295
848 (fhotx.T-fcoldx.T)/fhotx.T>=0.09= 0.31650249017426376
849 ++++++++++++++++++++++++++++++++++60 . adim++++++++++++++++++++++++++++++++++++++++++++++++++++
850 fcoldx.T= 685.2670609580745 fcoldx.h= 18487.567901752675 fhotx.T= 954.4196032667528 fhotx.h= 20473.261076340958
851 Isi transferi miktari= 1368048.0041838407 J/kg
852 fhotx.h= 20473.261076340958  q*a*delx/fhot.m*fhotx.Mass= 448.17252617062616
853 fcoldx.Mass*q*a*delx/fcold.m/1000= 448.17252617062627 fcoldx.h= 18487.567901752675 T= 954.4196032667528
854 fcoldx.T= 697.8961480909026 fcoldx.h= 18935.74042790142 fhotx.T= 968.0129787850022 fhotx.h= 20921.433602510886
855 Toplam uzunluk x= 0.6000000000000003
856 q= 1368048.0041838407
```

Şekil 6-19: Isı değiştirici çıkışı

Şekil 6-19 de görüldüğü gibi 900 K de plakalı ısı değiştiriciye giren sıcak akışkan yakıtımız olan doğalgazı 659 K'e kadar ısıtabilmektedir.

Buy şekilde diğer ısı değiştiriciler içinde gereken değerler aynı şekilde elde edilebilir.



Şekil 6-20:Örnek sistem modelinin tamamı

```
Nokta1 T: 298 K           Nokta10 T: 680 K
       P: 1.1 bar                 P: 1.1 bar

Nokta2 T: 689 K           Nokta11 T: 900 K
       P: 1.1 bar                 P: 1.1 bar

Nokta3 T: 800 K           Nokta12 T: 500 K
       P: 1.1 bar                 P: 1.1 bar

Nokta4 T: 298 K
       P: 1.1 bar

Nokta5 T: 900 K
       P: 1.1 bar

Nokta6 T: 850 K
       P: 1.1 bar

Nokta7 T: 500 K
       P: 1.1 bar

Nokta8 T: 298 K
       P: 1.1 bar

Nokta9 T: 298 K
       P: 1.1 bar
```

Şekil 6-21:Programdan elde edilen noktalar

```
++++++++++++++++++++++++++++++++++56 . adim++++++++++++++++++++++++++++++++++++++++++++++++++
fcoldx.T= 600.22271077469744 fcoldx.h= 16000.149055370700 fhotx.T= 903.20203708026720 fhotx.h= 18793.842229689075
Isi transferi miktari= 1232145.029586429 J/kg
fhotx.h= 18793.842229689075  q*a*delx/fhot.m*fhotx.Mass= 403.6507116925141
fcoldx.Mass*q*a*delx/fcold.m/1000= 403.65071169251416 fcoldx.h= 16808.149055378788 T= 903.20203708026720
fcoldx.T= 630.27147304193400 fcoldx.h= 17211.799766954235 fhotx.T= 915.55478270065270 fhotx.h= 19197.492941381500
Toplam uzunluk x= 0.5600000000000003
q= 1232145.029586429
(fhot.T-fhotx.T)/fhot.T>0.01= 0.05918431618902253
(fhotx.T-fcoldx.T)/fhotx.T>=0.09= 0.31159611095821604
++++++++++++++++++++++++++++++++++57 . adim++++++++++++++++++++++++++++++++++++++++++++++++++
fcoldx.T= 700.27147304193400 fcoldx.h= 17211.799766954235 fhotx.T= 915.55478270065270 fhotx.h= 19197.492941381500
Isi transferi miktari= 1264344.7977453256 J/kg
fhotx.h= 19197.492941381558  q*a*delx/fhot.m*fhotx.Mass= 414.19935574136866
fcoldx.Mass*q*a*delx/fcold.m/1000= 414.19935574136866 fcoldx.h= 17211.799766954235 T= 915.55478270065270
fcoldx.T= 637.45903582498850 fcoldx.h= 17625.999122614485 fhotx.T= 928.20165200260860 fhotx.h= 19611.692297115227
Toplam uzunluk x= 0.5700000000000003
q= 1264344.7977453256
```

Şekil 6-22: HEX2 nin sonucu ısı değiştirici boyu 0.5 m çıkmıştır.

7 SONUÇ

Bu çalışmada geliştirilen modeller gerçek akışkanların özelliklerini kullandıkları için ideal gaz varsayımı ile elde edilecek sonuçlara göre daha doğru yaklaşımlar elde edilmiştir. Özellikler Gibbs minimizasyonu yönteminin gerçek gaz özellikleri kullanılarak uygulanması yanma ve yakıt dönüştürücülerin başarılı bir şekilde modellenmesine imkân vermiştir.

Bu programın geliştirilmesinde temel amaç yakıt pili sistemlerinin termodinamik analizlerini hızlı bir biçimde ve gerçeğe en yakın şekilde gerçekleştirerek gerekli optimizasyonların yapılabilmesi için imkân sağlamaktır. Bu tip sistemler içinde en kritik elemanları oluşturan yakıt dönüşüm kısımlarının modellemesi yukarıda anlatıldığı gibi uygulanmıştır. Bunun yanında sistemde bulunması muhtemel ısı değiştirici gibi diğer elemanların modellemeleri de halihazır da kullanılan yöntemler ile programa eklenmiş ve bu sistemler de çalışmanın genel mantığına uygun şekilde programa eklenmiştir.

Sonuç olarak burada parça parça modellenen sistem bileşenleri kullanıcı tarafından yine bir program kullanarak birleştirilip bir sistemin tamamının modellenmesinde kullanılabilir hale getirilmiştir. Bu modelleme sonucu bir optimizasyon sağlamak mümkündür.

EKLER

Ek 1 Geliştirilen temel gibbs, programı[2]

Ek 2 LU, Ridder ve Bracket yöntemi

[2] Bu çalışma sırasında geliştirilen diğer programlar tezle birlikte sunulan CD içinde yer almaktadır. Bunların hacmi fazla olduğundan örnek olması amacı ile yalnızca temel iki sınıf burada verilmiştir.

gibbs.java (Geçek gaz özelliklerini kullanarak Gibbs minimizasyonu gerçekeleştiren program kodu)

```
public class gibbs  {
        //gibbsSpecies.java yı kullanır

        double[][] A;
        //LeeKesler[] LK;
        gibbsSpecies[] gs;
        double[][] n0;
        double[][] n00;
        double[][] n;
        double[][] del_lnnm;
        double[][] del_nm;
        double[][] lnnm;
        double nt;
        double lnnt;
        double del_lnnt;
        int NG;//Gas species number
        int l;//element number
        double[][] d_coeff;
        double[][] d_const;
        double[][] b0;
        double[][] b;
        double[][] mu;
        double[] mutemp;
        double P;
        double T;
        double T0;
        double[] Te;
        double lambda;
        double lambda1;
        double lambda2;
        double etemp=0;
        double Tref=298.2;
        double G=0;
        double Gnew=0;
        double diff=0;
        double SIZE=25.328436;//18.420681;
        double ntahminilk;
        int it_max=100;
        int it_max0=100;
        int o2=-1;
        int n2=-1;
        double MolWt=1;
        //double[] error=new double[];
        public gibbs(gibbsSpecies[] igs,double[][] in0,double iT,double iP){
                int nn=igs.length;
                int nat=110;
                double B[][]=new double[nat][nn];
```

```
                        int nk=0;
                        for(int i=0;i<nn;i++)
                        {
for(nk=0;nk<igs[i].atomList.length;nk++){B[igs[i].atomList[nk].number-
1][i]=igs[i].atomList[nk].N;}}
                        //System.out.println("B = "+Matrix.toString(B));
                        int natcount=0;
                        boolean zeroflag[]=new boolean[110];
                        for(int j=0;j<nat;j++)
                        {zeroflag[j]=false;for(int
i=0;i<nn;i++){if(B[j][i]!=0){zeroflag[j]=true;}};if(zeroflag[j]) natcount++;}
                        A=new double[natcount][nn];
                        int k=0;
                        for(int j=0;j<nat;j++)
                        {if(zeroflag[j]){A[k]=B[j];k++;}}
                        n0=in0;
                        n00=new double[n0.length][1];
                        double tempntn0=getnt(in0);
                        for(int i=0;i<igs.length;i++){
                                        MolWt+=gs[i].MolWt*in0[i][0];
                        }
                        for(int i=0;i<n0.length;i++) n00[i][0]=in0[i][0];
                        gs=igs;
                        P=iP;
                        T=iT;
                        for(int i=0;i<in0.length;i++) Te[i]=iT;
                        mu=new double[gs.length][1];
                        mutemp=new double[gs.length];
                        l=A.length;
                        NG=A[0].length;
                        //n=n0;
                        //n=in;
                        lambda=1;
                        del_lnnm=new double[NG][1];
                        del_nm=new double[NG][1];
                        lnnm=new double[NG][1];

                        d_coeff=new double[l+1][l+1];
                        d_const=new double[l+1][1];
                        b=new double[l][1];
                        nt=getnt(n0);
                        ntahminilk=nt/n0.length;
                        n=new double[n0.length][1];
                        for(int i=0;i<n0.length;i++)
                        {n[i][0]=ntahminilk;}
                        nt=getnt(n);
                        b0=new double[l][1];
                        for(int ii=0;ii<b0.length;ii++){
                                for(int ij=0;ij<NG;ij++){
                                        b0[ii][0]+=A[ii][ij]*n0[ij][0];

                                }
                        }
```

```
                    calMu(n0);
                    }
                    public gibbs(String igs[],double[][] in0,double iT,double iP,int
iit_max){
                                it_max=iit_max;
                                it_max0=it_max;
                                Te=new double[in0.length];
                                for(int i=0;i<in0.length;i++) Te[i]=iT;
                                setgibbs(igs,in0,iT,iP);
                    }

                    public gibbs(String igs[],double[][] in0,double iT,double iP,int
iit_max,double[][] in){

                                it_max=iit_max;
                                it_max0=it_max;
                                n=in;
                                Te=new double[in0.length];
                                for(int i=0;i<in0.length;i++) Te[i]=iT;
                                setgibbs(igs,in0,iT,iP);
                    }

                    public gibbs(String igs[],double[][] in0,double iTe[],double
iT,double iP,int iit_max){
                                it_max=iit_max;
                                it_max0=it_max;
                                Te=new double[iTe.length];
                                for(int i=0;i<iTe.length;i++)
                                Te[i]=iTe[i];
                                setgibbs(igs,in0,iT,iP);
                    }
                    public gibbs(String igs[],double[][] in0,double iTe[],double iP,int
iit_max){
                                it_max=iit_max;
                                it_max0=it_max;
                                Te=new double[iTe.length];
                                for(int i=0;i<iTe.length;i++)
                                Te[i]=iTe[i];
                                //T=Te[0];
                                setgibbs(igs,in0,iP);
                    }

                    public void setgibbs(String igs[],double[][] in0,double iT,double
iP){
                                gs=new gibbsSpecies[igs.length];
                                MolWt=1;
                                double tempntn0=getnt(in0);
                                for(int i=0;i<igs.length;i++){
                                            gs[i]=new gibbsSpecies(igs[i],"gas");
                                            MolWt+=gs[i].MolWt*in0[i][0];
                                }
                                for(int i=0;i<igs.length;i++){
                                            gs[i]=new gibbsSpecies(igs[i],"gas");
```

```
                }
            int nn=igs.length;
                int nat=110;
                double B[][]=new double[nat][nn];
                int nk=0;
                for(int i=0;i<nn;i++){
                        for(nk=0;nk<gs[i].atomList.length;nk++){
                                B[gs[i].atomList[nk].number-
1][i]=gs[i].atomList[nk].N;
                                }
                        }
                int natcount=0;
                boolean zeroflag[]=new boolean[110];
                for(int j=0;j<nat;j++)
                {zeroflag[j]=false;for(int
i=0;i<nn;i++){if(B[j][i]!=0){zeroflag[j]=true;}};if(zeroflag[j]) natcount++;}
                A=new double[natcount][nn];
                int k=0;
                for(int j=0;j<nat;j++)
                {if(zeroflag[j]){A[k]=B[j];k++;}}
                n0=in0;
                n00=new double[n0.length][1];
                for(int i=0;i<n0.length;i++) n00[i][0]=in0[i][0];
                P=iP;
                T=iT;
                T0=iT;
                mu=new double[gs.length][1];
                mutemp=new double[gs.length];
                l=A.length;
                NG=A[0].length;
                lambda=1;
                del_lnnm=new double[NG][1];
                del_nm=new double[NG][1];
                lnnm=new double[NG][1];

                d_coeff=new double[l+1][l+1];
                d_const=new double[l+1][1];
                b=new double[l][1];
                nt=getnt(n0);
                ntahminilk=nt/n0.length;
                n=new double[n0.length][1];
                for(int i=0;i<n0.length;i++){
                        n[i][0]=ntahminilk;
                }
                nt=getnt(n);

                b0=new double[l][1];
                for(int ii=0;ii<b0.length;ii++){
                        for(int ij=0;ij<NG;ij++){
                                b0[ii][0]+=A[ii][ij]*n0[ij][0];

                        }
                }
```

```
                calMu(n0);
        }

        public void setgibbs(String igs[],double[][] in0,double iP){
                gs=new gibbsSpecies[igs.length];
                MolWt=1;
                double tempntn0=getnt(in0);
                for(int i=0;i<igs.length;i++){
                        gs[i]=new gibbsSpecies(igs[i],"gas");
                        MolWt+=gs[i].MolWt*in0[i][0];
                }
                for(int i=0;i<igs.length;i++){
                        gs[i]=new gibbsSpecies(igs[i],"gas");
                }
        int nn=igs.length;
                int nat=110;
                double B[][]=new double[nat][nn];
                int nk=0;
                for(int i=0;i<nn;i++){
                        for(nk=0;nk<gs[i].atomList.length;nk++){
                                B[gs[i].atomList[nk].number-
1][i]=gs[i].atomList[nk].N;
                                }
                        }
                int natcount=0;
                boolean zeroflag[]=new boolean[110];
                for(int j=0;j<nat;j++)
                {zeroflag[j]=false;for(int
i=0;i<nn;i++){if(B[j][i]!=0){zeroflag[j]=true;}};if(zeroflag[j]) natcount++;}
                A=new double[natcount][nn];
                int k=0;
                for(int j=0;j<nat;j++)
                {if(zeroflag[j]){A[k]=B[j];k++;}}
                n0=in0;
                n00=new double[n0.length][1];
                for(int i=0;i<n0.length;i++) n00[i][0]=in0[i][0];
                P=iP;
                mu=new double[gs.length][1];
                mutemp=new double[gs.length];
                l=A.length;
                NG=A[0].length;
                lambda=1;
                del_lnnm=new double[NG][1];
                del_nm=new double[NG][1];
                lnnm=new double[NG][1];

                d_coeff=new double[l+1][l+1];
                d_const=new double[l+1][1];
                b=new double[l][1];
                nt=getnt(n0);
                ntahminilk=nt/n0.length;
                n=new double[n0.length][1];
                for(int i=0;i<n0.length;i++){
```

```
                                        n[i][0]=ntahminilk;
                                }
                                nt=getnt(n);

                                b0=new double[l][1];
                                for(int ii=0;ii<b0.length;ii++){
                                        for(int ij=0;ij<NG;ij++){
                                                b0[ii][0]+=A[ii][ij]*n0[ij][0];

                                        }
                                }
                                //calMu(n0);
                        }
                        public void setN0(double[][] in0){
                                //n00=new double[n0.length][1];
                           for(int i=0;i<n0.length;i++) n0[i][0]=in0[i][0];
                        }
                        public void setN0Default(){
                                //n00=new double[n0.length][1];
                           for(int i=0;i<n0.length;i++) n0[i][0]=n00[i][0];
                           b0=getb(n00);
                        }
                        public void setT(double iTeq){
                                T=iTeq;
                                calMu(n0);
                        }
                        public void setTDefault(){
                                T=T0;
                        }
                        public void setNmax(int iit_max){
                                it_max=iit_max;
                        }
                        public void setNmaxDefault(){
                                it_max=it_max0;
                        }
                        public void getEq(){
                                //System.out.println(l+"  "+NG);
                                double[][] innern0=new double[n0.length][1];
                                double[][] innern=new double[n0.length][1];
                                for(int i=0;i<n0.length;i++){
                                        innern0[i][0]=n0[i][0]/MolWt;
                                        //innern[i][0]=n[i][0]/MolWt;
                                }
                                nt=getnt(innern0);
                                ntahminilk=nt/innern0.length;
                                for(int i=0;i<innern0.length;i++){
                                        innern[i][0]=ntahminilk;
                                }
                                nt=getnt(innern);
                                //System.out.println("iterasyon ba??
n0=\n"+Matrix.toString(n0));
                                /*for(int ii=0;ii<b0.length;ii++){
                                        b0[ii][0]=0;
```

```
                                for(int ij=0;ij<NG;ij++){

        b0[ii][0]+=A[ii][ij]*n0[ij][0];
                                }
                        }
                        b0=getb(innern0);
                                double emax=1e-13;
                                double etest=1;
                                int it_counter=0;
                                double temp1=0;
                                double temp2=0;
                                lnnt=0;
                                /*System.out.println("n0=\n"+Matrix.toString(n0));
                                System.out.println("ba?lang?ç Gn0= "+getG(n0));*/
                                //System.out.println("iterasyona ba?larken
n=\n"+Matrix.toString(n));
                                //System.out.println("ba?lang?ç Gn= "+getG(n));
                                //getmu(T);
                                //System.out.println("T= "+T);
                                for(int i=0;i<gs.length;i++) gs[i].getmu0(T,P);

                        while(it_counter<it_max && etest>emax){
                                temp1=0;
                                for(int k=0;k<l;k++){
                                for(int i=0;i<l;i++){
                                for(int j=0;j<NG;j++){
                                        temp1+=A[k][j]*A[i][j]*innern[j][0];
                                                                }
                                                        d_coeff[k][i]=temp1;
                                                                temp1=0;
                                                }
                                        }

        //System.out.println("d_coeff=\n"+Matrix.toString(d_coeff));
        for(int k=0;k<l;k++){
        for(int j=0;j<NG;j++){
        temp2+=A[k][j]*innern[j][0];
                                        }
                                        d_coeff[k][l]=temp2;
                                        temp2=0;
                                        }
                                        temp1=0;
                                        temp2=0;

        //System.out.println("d_coeff=\n"+Matrix.toString(d_coeff));
                                for(int i=0;i<l;i++){
                                for(int j=0;j<NG;j++){
                                temp1+=A[i][j]*innern[j][0];
                                                }
                                                d_coeff[l][i]=temp1;
                                                temp1=0;
                                        }
```

```
                                        d_coeff[l][l]=getnt(innern)-nt;

                                        b=getb(innern);
                                        calMu(innern);
                                        for(int k=0;k<l;k++) {
                                                for(int j=0;j<NG;j++){
                                temp1+=A[k][j]*innern[j][0]*mu[j][0]/(8.3145*T);
                                                }
                                        d_const[k][0]=b0[k][0]-b[k][0]+temp1;
                                                temp1=0;
                                        }
                                        temp1=0;
                                        for(int j=0;j<NG;j++){

        temp1+=innern[j][0]*mu[j][0]/(8.3145*T);
                                                }
                                        d_const[l][0]=temp1+nt-getnt(innern);
                                        double[][] x=Matrix.AXB(d_coeff,d_const);
                                        del_lnnt=x[l][0];
                                        temp2=0;
                                        for(int j=0;j<NG;j++){

                for(int i=0;i<l;i++){temp2+=A[i][j]*x[i][0]; }
                del_lnnm[j][0]=temp2+x[l][0]-mu[j][0]/(8.3145*T);
                                                temp2=0;
                                        }
                                for(int i=0;i<del_lnnm.length;i++){
                                lnnm[i][0]=Math.log(innern[i][0]);
                                del_nm[i][0]=Math.exp(del_lnnm[i][0]);      }
        ////////////////////////////////////////////////////////////////////////////////////////////////
                                        double q=0;
                                        double max=0;
                                        double min=1e10;
                                        double s=0;
                                        for(int i=0;i<NG;i++){
        if((lnnm[i][0]-Math.log(getnt(innern)))>-SIZE){
max= Math.abs(del_lnnm[i][0])>Math.abs(max) ?  del_lnnm[i][0] :max ;
                                                }
                                        else {
s=del_lnnm[i][0]>0 ? (Math.abs((-1*(lnnm[i][0]-Math.log(getnt(innern)))-
9.2103404)/(del_lnnm[i][0]-del_lnnt))) : 1000;
min= s>min ? min : s;

        //System.out.println("s= "+s);
                                                }

                                        }
                                        q=(5*Math.abs(del_lnnt)>Math.abs(max)) ?
(5*Math.abs(del_lnnt)):Math.abs(max);
                                        //System.out.println("q= "+q);

                                        lambda1=2/q;
                                        lambda2=min;
```

```
		lambda= (lambda1<lambda2) ? (1<lambda1 ? 1 : lambda1) : (1<lambda2 ? 1 :
lambda2);
                                        G=getG(innern);
for(int i=0;i<NG;i++){
		lnnm[i][0]+=lambda*del_lnnm[i][0];
		innern[i][0]=Math.exp(lnnm[i][0]);
		lnnt+=lambda*del_lnnt;
		Gnew=getG(innern);
		etest=etest(innern);
		it_counter++;}
		if(it_counter==it_max)System.out.println("gibbs.java:442 Maximum number
of Iteration has been exceeded!");
                                        for(int i=0;i<n.length;i++){
                                                n[i][0]=MolWt*innern[i][0];
                                         }
                        }
                        public double delQ(){
                                        double Q=0;
                                        double href=0;
                                        for(int i=0;i<n.length;i++){
                                        Q+=gs[i].ht(T0)*n[i][0]-gs[i].ht(Te[i])*n0[i][0];
                        }
                                        return Q;//Kj
                        }
                        public double delQ_T(double innerT){
                                        //System.out.println("delQ_T baslad?");
                                        double Q=0;
                                        double href=0;
                                        setT(innerT);
                                        setNmax(100);
                                        getEq();
                                        //System.out.println("2");
                                        for(int i=0;i<n.length;i++){
                                        Q+=gs[i].ht(innerT)*n[i][0]-gs[i].ht(Te[i])*n0[i][0];
                                        }
                                        setTDefault();
                                        setNmaxDefault();
                                        //System.out.println("delQ_T bitti");
                                        return Q;//Kj
                        }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
                        public double Airad(double iQ,double ix1,double ix2){
                                        for(int i=0;i<gs.length;i++){
                                                if(gs[i].Formula.equals("O2")){
                                                        o2 = i;
                                                        break;
                                                }
                                        }
                        for(int i=0;i<gs.length;i++){
                                                if(gs[i].Formula.equals("N2")){
                                                        n2 = i;
                                                        break;
```

```
                                        }
                                        //System.out.println("n2= "+n2);
                                }
                                if(o2!=-1 && n2!=-1){
                                        //Ridder Metodu ile lineer olmayan
denklemlerin köklerinin bulunmas?
                                //referans : Numerical Recipes in C, second edition, William
H. Press,
                                //Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery
                                //cambridge university press
                        double[][] x_bracket=new double[2][1];
                        x_bracket=delQAirBracket(iQ,ix1,ix2);
                        double x1=x_bracket[0][0];
                        double x2=x_bracket[1][0];
                        //System.out.println("x1= "+x1+" x2= "+x2);
                        int MAXIT=50;
                          double xacc=1.0e-4;int j;
                                double fl,fh,xl,xh,swap,dx,del,ff;
                                double rtf=x1;
                                fl=delQAir(x1)-iQ;
                                fh=delQAir(x2)-iQ;
                                //System.out.println("fl= "+fl+" fh= "+fh);
                           if (fl*fh > 0.0)System.out.println("Kök s?n?rlar? do?ru olarak
seçilmemi? \n sonuç hatal? olabilir");
                                if (fl < 0.0) {
                                        xl=x1;
                                        xh=x2;
                                } else {
                                        xl=x2;
                                        xh=x1;
                                        swap=fl;
                                        fl=fh;
                                        fh=swap;
                                }
                                //System.out.println("xl= "+xl+" xh= "+xh);
                                //System.out.println("fl= "+fl+" fh= "+fh);
                                dx=xh-xl;
                                for (j=1;j<=MAXIT;j++) {
                                        rtf=xl+dx*fl/(fl-fh);
                                        ff=delQAir(rtf)-iQ;
                                        if (ff < 0.0) {
                                                del=xl-rtf;
                                                xl=rtf;
                                                fl=ff;
                                        } else {
                                                del=xh-rtf;
                                                xh=rtf;
                                                fh=ff;
                                        }
                                        //System.out.println("rtf= "+rtf);
                                        dx=xh-xl;
                                        //System.out.println("xl= "+xl+" xh= "+xh);
                                //System.out.println("fl= "+fl+" fh= "+fh);
```

```
            if (Math.abs(del) < xacc || ff == 0.0) return rtf;
                        }
System.out.println("Uyar? maximum iterasyon say?s? a??ld? \n"+
                    " çözüm geçerli olm?yabilir");
                     return rtf;
                        }else {
                    System.exit(1);
                        }
                        return 0;
                    }
                    public double delQAir(double x){
                        for(int i=0;i<n0.length;i++) n0[i][0]=n00[i][0];
                        n0[o2][0]=x;
                        n0[n2][0]=x*79/21;
                        System.out.println(LK[i].Formula+"  "+n0[i][0]);
                        return delQ_T(T);
                    }

                    public double delQAir(double x,int ii,int ij){
                        for(int i=0;i<n0.length;i++){
                                n0[i][0]=n00[i][0];
                        }
                        n0[ii][0]=x;
                        n0[ij][0]=x*79/21;
                        return delQ_T(T);
                    }

                    public  double[][] delQAirBracket(double iQ,double x1,double x2){
                        // koklerin yer ald??? alt bölgeleri saptar
                        // n : verilen bölgeyi böldü?ümüz alt bölge say?s?
                        // x1,x2 : s?n?r de?erleri
                        // nbb = aranan bölgedeki köksay?s?
                        int n=4;
                        int nbb=1;
                        int nb;
                        int i;
                        double x,fp,fc,dx;
                        double xb[][]=new double[2][nbb];
                        nb=0;
                        dx=(x2-x1)/n;
                        x=x1;
                        //System.out.println("x= "+x);
                        fp=delQAir(x1)-iQ;
                        //System.out.println("x1= "+x1+" fp= "+fp);
                        for (i=1;i<=n;i++)
                        {
                                x+=dx;
                                fc=delQAir(x)-iQ;
                                //System.out.println("x= "+x+" fc= "+fc);
                                // e?er 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 tamamland? kök olan bölge bulunamad?");
                        else if(nb<nbb){
System.out.println("arama tamamland? sadece "+nb+" adet kök bulundu \n"+
                          "siz "+nbb+" adet kök için arama yapt?rd?n?z");
                         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  double[][] TadBracket(double iQ){
                                // koklerin yer ald??? alt bölgeleri saptar
                                // n : verilen bölgeyi böldü?ümüz alt bölge say?s?
                                // x1,x2 : s?n?r de?erleri
                                // nbb = aranan bölgedeki köksay?s?
                                int n=4;
                                int nbb=1;
                                int nb;
                                double x1=298;
                                double x2=5000;
                                int i;
                                double x,fp,fc,dx;
                                double xb[][]=new double[2][nbb];
                                nb=0;
                                dx=(x2-x1)/n;
                                x=x1;
                                fp=delQ_T(x1)-iQ;
                                //System.out.println("x1= "+x1+" fp= "+fp);
                                for (i=1;i<=n;i++)
                                {
                                        x+=dx;
                                        fc=delQ_T(x)-iQ;
                                        //System.out.println("x= "+x+" fc= "+fc);
                                        // eger 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 tamamland? kök olan bölge bulunamad?")
else if(nb<nbb){
System.out.println("arama tamamland? sadece "+nb+" et kök bulundu \n"+
"siz "+nbb+" adet kök için arama yapt?rd?n?z");
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  double Tad(double iQ)
{
// Ridder Metodu ile lineer olmayan denklemlerin köklerinin bulunmas?
//referans : Numerical Recipes in C, second edition, William H. Press,
//Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery
//cambridge university press
double[][] x_bracket=new double[2][1];
x_bracket=TadBracket(iQ);
double x1=x_bracket[0][0];
  double x2=x_bracket[1][0];
 //System.out.println("x1= "+x1+" x2= "+x2);
 int MAXIT=50;
 double xacc=1.0e-4;int j;
double fl,fh,xl,xh,swap,dx,del,ff;
double rtf=x1;
fl=delQ_T(x1)-iQ;
fh=delQ_T(x2)-iQ;
if (fl*fh > 0.0)System.out.println("Kök s?n?rlar? do?ru olarak lmemi? \n sonuç hatal?
olabilir");
if (fl < 0.0) {
xl=x1;
xh=x2;
} else {
xl=x2;
xh=x1;
swap=fl;
fl=fh;
fh=swap;
}
dx=xh-xl;
for (j=1;j<=MAXIT;j++) {
rtf=xl+dx*fl/(fl-fh);
ff=delQ_T(rtf)-iQ;
if (ff < 0.0) {
del=xl-rtf;
xl=rtf;
fl=ff;
} else {
del=xh-rtf;
xh=rtf;
fh=ff;
}
dx=xh-xl;
if (Math.abs(del) < xacc || ff == 0.0) return rtf;                                    }
System.out.println("Uyar? maximum iterasyon say?s? a??ld? \n"+
  " çözüm geçerli olm?yabilir");
return rtf;
}
```

```
public void calMu(double[][] in){
for(int i=0;i<mu.length;i++){
mu[i][0]=gs[i].mu(T,P,in[i][0],getnt(in));
}
}
double lnP=Math.log(P/P0);
//for(int i=0;i<n.length;i++) nt+=in[i][0];
for(int i=0;i<mu.length;i++){
//System.out.println(""+LK[i].better_Cp);
value=Math.abs(in[i][0])<1e-15? 0 :Math.log(in[i][0]/getnt(in));
//mu[i][0]=(LK[i].m0_T_LK_V(T)+8.3145*T*Math.log(in[i][0]/getnt(in))+8.3145*T*lnP
);
mu[i][0]=(mutemp[i]+8.3145*T*value+8.3145*T*lnP);
//mu[i][0]*=in[i][0];
//System.out.println(LK[i].Formula+"  g(t)= "+LK[i].m0_T_LK_V(T)+"  RTlnxi=
"+(8.3145*T*value)+"  RTlnP= "+(8.3145*T*lnP)+"  "+mu[i][0]);}
//System.out.println("mu=\n"+Matrix.toString(mu));
//System.out.println("mu/RT=\n"+Matrix.toString(Matrix.multiply((1/8.3145/T),mu)));}
/*public void getmu(double T){
//System.out.println("T= "+T);
for(int i=0;i<mutemp.length;i++) mutemp[i]=LK[i].m0_T_LK_V(T);
//System.out.println("mu2");
}*/
public double[][] getb(double[][] in){
//System.out.println("A= \n"+Matrix.toString(A));
//System.out.println("in= \n"+Matrix.toString(in));
double inner_b[][]=new double[A.length][1];
for(int ii=0;ii<inner_b.length;ii++){
inner_b[ii][0]=0;
for(int ij=0;ij<NG;ij++){
//System.out.println("A[ii][ij]= "+A[ii][ij]+" n0[ij][0]="+n0[ij][0] );
inner_b[ii][0]+=A[ii][ij]*in[ij][0];
}
return inner_b;
}
public double etest(double[][] iinnern){
double[][] e=new double[NG+2][1];
double inner_etest=0;
//System.out.println("n=\n "+Matrix.toString(n));
//System.out.println("deln=\n "+Matrix.toString(del_lnnm));
for(int j=0;j<NG;j++){
//System.out.println(""+del_lnnm[j][0]+"  "+n[j][0]+" "+getnt(n));
e[j][0]=iinnern[j][0]*Math.abs(del_lnnm[j][0])/getnt(iinnern);
//System.out.println(j+". eleman  "+e[j][0]);  }
e[NG][0]=nt*Math.abs(del_lnnt)/getnt(iinnern);
//System.out.println("toplam n. eleman  "+e[NG][0]);
e[NG+1][0]=Math.abs(G-Gnew)/Math.abs(G);
//System.out.println("g hatas?. eleman  "+e[NG+1][0]);
//double etest=Matrix.norminf(Matrix.vek(e,0));
/*System.out.println("b0=\n"+Matrix.toString(b0));
System.out.println("b0=\n"+Matrix.toString(b0));
System.out.println("Matrix.substract(b0,getb(n))=\n"+Matrix.toString(Matrix.substract(get
b(n),b0)));*/
```

```
./System.out.println("Matrix.vek(Matrix.substract(b0,getb(n)),0))=\n"+Matrix.toString(Ma
trix.vek(Matrix.substract(b0,getb(n)),0)));
//System.out.println("Substract=\n"+Matrix.toString((Matrix.substract(Matrix.vek(b0,0),M
atrix.vek(getb(n),0)))));
etest=Matrix.norminf(Matrix.vek(Matrix.substract(b0,getb(n)),0));
double etest1=Matrix.normEns(Matrix.substract(b0,getb(iinnern)));
double etest0=Matrix.normEns(e);
//System.out.println("etest1= "+etest1+"  etest0= "+etest0);
if(etest0>1e-5){
ıf(etest1>1e-6){
inner_etest= etest1<etest0 ? etest0 : etest1;
return inner_etest;
}
else return etest0;
}
else{
if(etest1>1e-6)            return etest1;
else{
inner_etest= etest1<etest0 ? etest0 : etest1;
return inner_etest;
}
}
//return etest1;
}
public double getnt(double[][] jn){
double nt_inner=0;
for(int i=0;i<jn.length;i++) nt_inner+=jn[i][0];
return nt_inner;
}
public double getG(double[][] in){
calMu(in);
double munt=0;
for(int j=0;j<mu.length;j++) mu[j][0]*=in[j][0];
for(int i=0;i<mu.length;i++) munt+=mu[i][0];
return munt;
}
public double[][] getSolution(){
return n;
public int geto2Index(){return o2;
}
public int getn2Index(){
return n2;
}
}
```

AutoThermalRef.java (gibbs.java'yı kuulanarak ototermal yakıt dönüştürücü hesabı yapan program kodu)

```
import java.util.*;

public class AutoThermalRef {

	Vector[] vStore=new Vector[3];//vStore[0] contains name of all gases ,
vStore[1] contains kmoles of all gases , vStore[2] contains temperature of all gases
	Vector   vrefproName=new Vector<String>();
	String[] Fuel;
	String[] compeq;
	String[] refproName;
	double[] refproN;
	double[][] neq;
	double Teq;
	double[] Tfuelfinal;
	double[][] NFuel;
	double Nair;
	double NairMax;
	double[] TFuel;
	double Tout;
	double Tair;
	double Q;
	double P;
	gibbs gb;
	int scenario=0;
	final int nmax=100;
	String[] resultName;
	double[] resultN;
	fluidPoint fpfuelin;
	fluidPoint fpairin;
	fluidPoint fpout;
	String s="";

	/*
*for scenario 1 : known -fuel composition and amount  , fuel temperature , air amount and
temperature ,
*heat leak amount-  unknown -product temperature , product composition-
*/
	public AutoThermalRef(fluidPoint fpfuel,fluidPoint fpair,double iQ,double
iP){
		try{
		fpfuelin=fpfuel;
		fpairin=fpair;
		String[] fpairGasList=fpair.getGasList();
		double[] fpairGasMoles=fpair.getGasMoles();
		double iNair=0;
		for(int i=0;i<fpairGasList.length;i++){
			System.out.println("fpairGasList= "+fpairGasList[i]);
			if(fpairGasList[i].equals("O2")){
```

```
                                        iNair=fpairGasMoles[i];


                        }
                        }
                        setScenario(1);
                        refproName=new String[4];
                        refproName[0]="H2";
                        refproName[1]="CO";
                        refproName[2]="CO2";
                        refproName[3]="H2O";
                        setQ(iQ);
                        setP(iP);
                        setNair(iNair);
                        //System.out.println("Tair=  "+iTair);
                        setTair(fpair.T);
                        refproN=new double[refproName.length];

                        for(int i=0;i<refproN.length;i++) refproN[i]=0.0;

        s="=============================================================
========================\n";

        setAutoThermalRef(fpfuel.getGasList(),fpfuel.getGasMoles(),fpfuel.T);

        }catch(GException e){System.out.println(" "+e.getLocalizedMessage());}
        }
        public AutoThermalRef(String[] iFuel,double[][] iNFuel,double iNair,double[]
iTFuel,double iTair,double iQ,double iP){
                        setScenario(1);
                        refproName=new String[4];
                        refproName[0]="H2";
                        refproName[1]="CO";
                        refproName[2]="CO2";
                        refproName[3]="H2O";
                        setQ(iQ);
                        setP(iP);
                        setNair(iNair);
                        //System.out.println("Tair=  "+iTair);
                        setTair(iTair);
                        refproN=new double[refproName.length];

                        for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                        setAutoThermalRef(iFuel,iNFuel,iTFuel);
        }
        public AutoThermalRef(String[] iFuel,double[][] iNFuel,double iNair,double[]
iTFuel,
        double iTair,double iQ,double iP,String[] irefproName){
                        setScenario(1);
                        setQ(iQ);
                        setP(iP);
                        setNair(iNair);
                        setTair(iTair);
```

```
                        setrefproName(irefproName);
                        refproN=new double[irefproName.length];

                        for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                        setAutoThermalRef(iFuel,iNFuel,iTFuel);
                }
                public AutoThermalRef(Vector[] iv,double iNair,double iTair,double iQ,double
iP) throws InappropriateArrayException{
                        if(iv[0].size()!=iv[1].size() || iv[1].size()!=iv[2].size() ) throw new
InappropriateArrayException();
                        setScenario(1);
                        refproName=new String[4];
                        refproName[0]="H2";
                        refproName[1]="CO";
                        refproName[2]="CO2";
                        refproName[3]="H2O";
                        setQ(iQ);
                        setP(iP);
                        setNair(iNair);
                        setTair(iTair);
                        refproN=new double[refproName.length];

                        for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                        setrefproName(refproName);
                        setAutoThermalRef(iv);
                }
                public AutoThermalRef(Vector[] iv,double iNair,double iTair,double iQ,double
iP,
                Vector irefproName) throws InappropriateArrayException{
                        if(iv.length!=3 || iv[0].size()!=iv[1].size() || iv[1].size()!=iv[2].size()
)
                        throw new InappropriateArrayException();
                        setScenario(1);
                        setQ(iQ);
                        setP(iP);
                        setNair(iNair);
                        setTair(iTair);
                        setrefproName(irefproName);
                        setAutoThermalRef(iv);
                }
/*
*Scenario 1 ends
*/



/*
*for scenario 2 : known -fuel composition and amount  , fuel temperature , air  temperature
,heat leak
*Outlet temperature-  unknown -needed heat , product composition-
*/
                public AutoThermalRef(double iTout,fluidPoint fpfuel,fluidPoint fpair,double
iQ,double iP,double iNairMax){
```

```
		try{
		String[] fpairGasList=fpair.getGasList();
		double[] fpairGasMoles=fpair.getGasMoles();
		double iNair=0;
		for(int i=0;i<fpairGasList.length;i++){
			System.out.println("fpairGasList= "+fpairGasList[i]);
			if(fpairGasList[i].equals("O2")){
				//System.out.println("sfdbaSd.vjSD.k");
				iNair=fpairGasMoles[i];

		}
		}
		setScenario(2);
		refproName=new String[4];
		refproName[0]="H2";
		refproName[1]="CO";
		refproName[2]="CO2";
		refproName[3]="H2O";
		setTout(iTout);
		//setTeq(iTeq);
		setP(iP);
		setQ(iQ);
		setNair(0.0);
		setNairMax(iNairMax);
		setTair(fpair.T);
		refproN=new double[refproName.length];

		for(int i=0;i<refproN.length;i++) refproN[i]=0.0;

	setAutoThermalRef(fpfuel.getGasList(),fpfuel.getGasMoles(),fpfuel.T);

	}catch(GException e){System.out.println(" "+e.getLocalizedMessage());}
	}

	public AutoThermalRef(double iTout,String[] iFuel,double[][] iNFuel,double
iQ,
	double[] iTFuel,double iNairMax,double iTair,double iP){
		setScenario(2);
		refproName=new String[4];
		refproName[0]="H2";
		refproName[1]="CO";
		refproName[2]="CO2";
		refproName[3]="H2O";
		setTout(iTout);
		//setTeq(iTeq);
		setP(iP);
		setQ(iQ);
		setNair(0.0);
		setNairMax(iNairMax);
		setTair(iTair);
		refproN=new double[refproName.length];

		for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
```

```
                setAutoThermalRef(iFuel,iNFuel,iTFuel);
        }
        public AutoThermalRef(double iTout,String[] iFuel,double[][] iNFuel,double[]
iTFuel,
        double iNairMax,double iTair,double iQ,double iP,String[] irefproName ){
                setScenario(2);
                setTout(iTout);
                setQ(iQ);
                setP(iP);
                setNair(0.0);
                setNairMax(iNairMax);
                setTair(iTair);
                setrefproName(irefproName);
                refproN=new double[irefproName.length];

                for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                setAutoThermalRef(iFuel,iNFuel,iTFuel);
        }
        public AutoThermalRef(double iTout,Vector[] iv,double iNairMax,double
iTair,double iQ,double iP)
         throws InappropriateArrayException{
                if(iv[0].size()!=iv[1].size() || iv[1].size()!=iv[2].size() ) throw new
InappropriateArrayException();
                setScenario(2);
                refproName=new String[4];
                refproName[0]="H2";
                refproName[1]="CO";
                refproName[2]="CO2";
                refproName[3]="H2O";
                setTout(iTout);
                setQ(iQ);
                setP(iP);
                setNair(0.0);
                setNairMax(iNairMax);
                setTair(iTair);
                refproN=new double[refproName.length];

                for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                setrefproName(refproName);
                setAutoThermalRef(iv);
        }
        public AutoThermalRef(double iTout,Vector[] iv,double iNairMax
        ,double iTair,double iQ,double iP,Vector irefproName)
         throws InappropriateArrayException{
                if(iv[0].size()!=iv[1].size() || iv[1].size()!=iv[2].size() ) throw new
InappropriateArrayException();
                setScenario(2);
                setrefproName(irefproName);
                setTout(iTout);
                setQ(iQ);
                setP(iP);
                setNair(0.0);
                setNairMax(iNairMax);
```

```
                setTair(iTair);
                setAutoThermalRef(iv);
        }


/*
*Scenario 2 ends
*/

        //public void steTeq(double iTeq){Teq=iTeq}
        public void setTout(double iTout){Tout=iTout;}
        public void setQ(double iQ){ Q=iQ;}
        public void setP(double iP){P=iP;}
        public void setNair(double iNair){Nair=iNair;}
        public void setNairMax(double iNairMax){NairMax=iNairMax;}
        public void setTair(double iTair){Tair=iTair;}
        public void setrefproName(Vector irefproName){
        vrefproName.clear();
        vrefproName.addAll(irefproName);

        }
        public void setrefproName(String[] irefproName){
                //for(int i=0;i<irefproName.length;i++)
System.out.println("iref"+irefproName[i]);
                        refproName=irefproName;
                        //for(int i=0;i<refproName.length;i++)
System.out.println("ref"+refproName[i]);
                        vrefproName.clear();
                        for(int i=0;i<refproName.length;i++)
                        vrefproName.add(refproName[i]);

        }
        public void setScenario(int ii){
                scenario=ii;
        }
        public void setAutoThermalRef(String[] iFuel,double[] iNFuel,double iTFuel){

                vStore[0]=new Vector<String>();
                vStore[1]=new Vector<Double>();
                vStore[2]=new Vector<Double>();

                for(int i=0;i<iFuel.length;i++){
                        vStore[0].add(iFuel[i]);
                        vStore[1].add(iNFuel[i]);
                        vStore[2].add(iTFuel);
                }
                int index=0;
                double NairTot=0;
                double TairAvg=0;
                if(!vStore[0].contains("O2")){
                        //System.out.println("1");
                        vStore[0].add("O2");
                        vStore[1].add(Nair);
```

```
                        //System.out.println("Tair=  "+Tair);
                        vStore[2].add(Tair);
                }
                else{
                        index=vStore[0].indexOf("O2");

        NairTot=(((Double)vStore[1].elementAt(index)).doubleValue()+Nair);

        TairAvg=(((Double)vStore[1].elementAt(index)).doubleValue()*
                        ((Double)vStore[2].elementAt(index)).doubleValue()+
                        Nair*Tair)/NairTot;
                        vStore[1].setElementAt(NairTot,index);
                        vStore[2].setElementAt(TairAvg,index);
                }
                index=0;
                double NnitTot=0;
                double TnitAvg=0;
                if(!vStore[0].contains("N2")){
                        vStore[0].add("N2");
                        vStore[1].add(Nair*79/22);
                        vStore[2].add(Tair);
                }
                else{
                        index=vStore[0].indexOf("N2");

        NnitTot=(((Double)vStore[1].elementAt(index)).doubleValue()+Nair*79/22);


        TnitAvg=(((Double)vStore[1].elementAt(index)).doubleValue()*
                        ((Double)vStore[2].elementAt(index)).doubleValue()+
                        Nair*Tair*79/22)/NnitTot;

                        vStore[1].setElementAt(NnitTot,index);
                        vStore[2].setElementAt(TnitAvg,index);
                }
                //System.out.println(""+vStore[1].elementAt(index));
                //System.out.println(""+vStore[2].elementAt(index));
                for(int i=0;i<refproName.length;i++){
                        if(!vStore[0].contains(refproName[i])){
                                vStore[0].add(refproName[i]);
                                vStore[1].add(refproN[i]);
                                vStore[2].add(298.15);
                        }
                }
        }
        public void setAutoThermalRef(String[] iFuel,double[][] iNFuel,double[]
iTFuel){
                vStore[0]=new Vector<String>();
                vStore[1]=new Vector<Double>();
                vStore[2]=new Vector<Double>();

                for(int i=0;i<iFuel.length;i++){
                        vStore[0].add(iFuel[i]);
```

```
                    vStore[1].add(iNFuel[i][0]);
                    vStore[2].add(iTFuel[i]);
            }
            int index=0;
            double NairTot=0;
            double TairAvg=0;
            if(!vStore[0].contains("O2")){
                    //System.out.println("1");
                    vStore[0].add("O2");
                    vStore[1].add(Nair);
                    //System.out.println("Tair=  "+Tair);
                    vStore[2].add(Tair);
            }
            else{
                    index=vStore[0].indexOf("O2");

NairTot=(((Double)vStore[1].elementAt(index)).doubleValue()+Nair);

TairAvg=(((Double)vStore[1].elementAt(index)).doubleValue()*
                    ((Double)vStore[2].elementAt(index)).doubleValue()+
                    Nair*Tair)/NairTot;
                    vStore[1].setElementAt(NairTot,index);
                    vStore[2].setElementAt(TairAvg,index);
            }
            index=0;
            double NnitTot=0;
            double TnitAvg=0;
            if(!vStore[0].contains("N2")){
                    vStore[0].add("N2");
                    vStore[1].add(Nair*79/22);
                    vStore[2].add(Tair);
            }
            else{
                    index=vStore[0].indexOf("N2");

NnitTot=(((Double)vStore[1].elementAt(index)).doubleValue()+Nair*79/22);


TnitAvg=(((Double)vStore[1].elementAt(index)).doubleValue()*
                    ((Double)vStore[2].elementAt(index)).doubleValue()+
                    Nair*Tair*79/22)/NnitTot;

                    vStore[1].setElementAt(NnitTot,index);
                    vStore[2].setElementAt(TnitAvg,index);
            }
            //System.out.println(""+vStore[1].elementAt(index));
            //System.out.println(""+vStore[2].elementAt(index));
            for(int i=0;i<refproName.length;i++){
                    if(!vStore[0].contains(refproName[i])){
                            vStore[0].add(refproName[i]);
                            vStore[1].add(refproN[i]);
                            vStore[2].add(298.15);
                    }
```

```
                }
                //1 element is fuel reactants , 2-refproN H2O , refproN-Fuel
reformer products

                /*for(int i=0;i<compeq.length;i++){
                        System.out.println("compeq["+i+"]= "+compeq[i]);
                        System.out.println("neq["+i+"][0]= "+neq[i][0]);
                        System.out.println("Teq["+i+"]= "+Teq[i]);
                }       */
        }
        public void setAutoThermalRef(Vector[] iv)throws
InappropriateArrayException{
                //for(int i=0;i<3;i++) vStore[i]=new Vector();
                vStore[0]=new Vector(iv[0]);
                vStore[1]=new Vector(iv[1]);
                vStore[2]=new Vector(iv[2]);
                LeeKesler g;
                int index=0;
                double NairTot=0;
                double TairAvg=0;
                if(!vStore[0].contains("O2")){
                        vStore[0].add("O2");
                        vStore[1].add(Nair);
                        vStore[2].add(Tair);
                }
                else{
                        index=vStore[0].indexOf("O2");

        NairTot=(((Double)vStore[1].elementAt(index)).doubleValue()+Nair);

        TairAvg=(((Double)vStore[1].elementAt(index)).doubleValue()*
                        ((Double)vStore[2].elementAt(index)).doubleValue()+
                        Nair*Tair)/NairTot;
                        vStore[1].setElementAt(NairTot,index);
                        vStore[2].setElementAt(TairAvg,index);
                }
                index=0;
                double NnitTot=0;
                double TnitAvg=0;
                if(!vStore[0].contains("N2")){
                        vStore[0].add("N2");
                        vStore[1].add(Nair*79/22);
                        vStore[2].add(Tair);
                }
                else{
                        index=vStore[0].indexOf("N2");

        NnitTot=(((Double)vStore[1].elementAt(index)).doubleValue()+Nair*79/22);


        TnitAvg=(((Double)vStore[1].elementAt(index)).doubleValue()*
                        ((Double)vStore[2].elementAt(index)).doubleValue()+
```

```
                                Nair*Tair*79/22)/NnitTot;

                                vStore[1].setElementAt(NnitTot,index);
                                vStore[2].setElementAt(TnitAvg,index);
                        }
                        //System.out.println(""+vStore[1].elementAt(index));
                        //System.out.println(""+vStore[2].elementAt(index));
                        //System.out.println("vrefproName.size()= "+vrefproName.size());
                        for(int i=0;i<vrefproName.size();i++){
                                if(!vStore[0].contains(vrefproName.get(i))){
                                        vStore[0].add(vrefproName.get(i));
                                        vStore[1].add(0.0);
                                        vStore[2].add(298.15);
                                        //System.out.println(""+vrefproName.get(i));
                                }

                        }

                        if(vStore[0].size()!=vStore[1].size()) throw new
InappropriateArrayException();
                        //1 element is fuel reactants , 2-refproN H2O , refproN-Fuel
reformer products
                        /*for(int i=0;i<compeq.length;i++){
                                System.out.println("compeq["+i+"]= "+compeq[i]);
                                System.out.println("neq["+i+"][0]= "+neq[i][0]);
                                System.out.println("Tfuelfinal["+i+"]= "+Tfuelfinal[i]);
                        }*/
                }
/*
*set air reformer methods end
*/
/*
*add methods
*/
                public void addpro(String igas)throws GasAlreadyDefinedExcception{
                        vrefproName.add(igas);
                        if(!vStore[0].contains(igas)){
                                        vStore[0].add(igas);
                                        vStore[1].add(0.0);
                                        vStore[2].add(298.15);

                        }
                        else throw new GasAlreadyDefinedException();
                }

/*
*add methods end
*/

                public void calcEq(){
                        compeq=new String[vStore[0].size()];
                        neq=new double[vStore[1].size()][1];
                        Tfuelfinal=new double[vStore[2].size()];
```

```
                int ei=0;
                toString("Ototermal Reformer");
                toString("\nReaksiyona Girenler");
                for(int i=0;i<compeq.length;i++){
                        compeq[i]=(String)vStore[0].elementAt(i);

        neq[i][0]=((Double)vStore[1].elementAt(i)).doubleValue();

        Tfuelfinal[i]=((Double)vStore[2].elementAt(i)).doubleValue();

                }

                for(int i=0;i<compeq.length;i++){
                        //System.out.println("compeq["+i+"]= "+compeq[i]);

                }
                for(int i=0;i<compeq.length;i++){
                        System.out.println("compeq["+i+"]= "+compeq[i]);
                        System.out.println("neq["+i+"][0]= "+neq[i][0]);
                        toString(compeq[i]+":   "+neq[i][0]+" kmole  T=
"+Tfuelfinal[i]+" K");

                }
                for(int i=0;i<compeq.length;i++){

                        System.out.println("Tfuelfinal["+i+"]= "+Tfuelfinal[i]);
                }
                if(scenario==1){
                System.out.println("scenario 1");
                        gb=new gibbs(compeq,neq,Tfuelfinal,P,nmax);
                        resultName=new String[gb.gs.length];
                        resultN=new double[gb.n.length];
                        Tout=gb.Tad(Q);
                        toString("\n\nOtotermal Reformer Cikisi");
                        for(int i=0;i<gb.n.length;i++)
                        {
                                resultName[i]=gb.gs[i].Formula;
                                resultN[i]=gb.n[i][0];
                                System.out.println(gb.gs[i].Formula+"
"+gb.n[i][0]);
                                toString(gb.gs[i].Formula+":  "+gb.n[i][0]+"
kmole");

                        }
                        double Q=gb.delQ_T(Tout);
                        toString("Urunlerin Cikis Sicakligi:   "+Tout+" K");
                        toString("Ototermal Yakit Donusumu Reaksiyon Isi
Girdisi: "+Q+"  kj");
                        System.out.println("Tout= "+Tout);
                        System.out.println("Q= "+Q);
                }
                if(scenario==2){
                //System.out.println("scenario 2");
```

```
                                gb=new gibbs(compeq,neq,Tfuelfinal,Tout,P,nmax);
                                resultName=new String[gb.gs.length];
                                resultN=new double[gb.n.length];
                                double nAir=gb.Airad(0.0,0.0001,10.0);
                                toString("Istenilen Sicaklik Cikisini Saglamak Icin
Gereken O2 Miktari: "+nAir+"  kmole");
                                System.out.println("Required Air Amount kmole=
"+nAir);
                                for(int i=0;i<gb.n.length;i++){
                                        resultName[i]=gb.gs[i].Formula;
                                        resultN[i]=gb.n[i][0];
                                         System.out.println(gb.gs[i].Formula+"
"+gb.n[i][0]);
                                        toString(gb.gs[i].Formula+":  "+gb.n[i][0]+"
kmole");
                                 }
                                 double Q=gb.delQ_T(Tout);
                                 toString("Ototermal Yakit Donusumu Reaksiyon Isi
Girdisi: "+Q+"  kj");
                                System.out.println("Q= "+Q);

                        }

        }
        public fluidPoint getOutlet(){
                try{
                fpout=new fluidPoint(resultName,resultN,Tout,P);
                        return fpout;
                }catch(java.io.IOException e){System.out.println("IOException @
AutoThermalRef.java ");}
                catch(GException ge){System.out.println("
"+ge.getLocalizedMessage());}
                return null;
        }
        public String getResult(){
                return s;
        }
        public void toString(String a){
                s+=a+"\n";
        }
}
```

steamref.java (Buhralı yakıt dönüştürücü hesabı yapana program kodu)

```
import java.util.*;

public class steamRef{
        Vector[] vStore=new Vector[3];//vStore[0] contains name of all gases ,
vStore[1] contains kmoles of all gases , vStore[2] contains temperature of all gases
        Vector   vrefproName=new Vector<String>();
        String[] Fuel;
        String[] compeq0;
        String[] refproName;
        double[] refproN;
        double[][] neq0;
        double Teq;
        double[] TFuelfinal;
        double[][] NFuel;
        double Nsteam;
        double[] TFuel;
        double Tout;
        double Tsteam;
        double Q;
        double Qneeded=-1;
        double P;
        String[] resultName;
        double[] resultN;
       // gibbs gb;
        int scenario=0;
        final int nmax=100;
        fluidPoint fpfuelin;
       fluidPoint fpsteamin;
       fluidPoint fpout;

/*
*for scenario 1 : known -fuel composition and amount  , fuel temperature , steam amount
and temperature ,
*added heat amount-  unknown -product temperature , product composition-
*/
        public steamRef(fluidPoint fpfuel,fluidPoint fpsteam,double iNsteam,double
iQ,double iP){
                try{
                setScenario(1);
                refproName=new String[4];
                refproName[0]="H2";
                refproName[1]="CO";
                refproName[2]="CO2";
                refproName[3]="O2";
                setQ(iQ);
                setP(iP);
                setNsteam(iNsteam);
                setTsteam(fpsteam.T);
                refproN=new double[refproName.length];

                for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
```

```
                    setSteamRef(fpfuel.getGasList(),fpfuel.getGasMoles(),fpfuel.T);

            }catch(GException e){System.out.println(" "+e.getLocalizedMessage());}
            }
            public steamRef(String[] iFuel,double[][] iNFuel,double iNsteam,double[]
iTFuel,double iTsteam,double iQ,double iP){
                    setScenario(1);
                    refproName=new String[4];
                    refproName[0]="H2";
                    refproName[1]="CO";
                    refproName[2]="CO2";
                    refproName[3]="O2";
                    setQ(iQ);
                    setP(iP);
                    setNsteam(iNsteam);
                    setTsteam(iTsteam);
                    refproN=new double[refproName.length];

                    for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                    setSteamRef(iFuel,iNFuel,iTFuel);
            }
            public steamRef(String[] iFuel,double[][] iNFuel,double iNsteam,double[]
iTFuel,
            double iTsteam,double iQ,double iP,String[] irefproName){
                    setScenario(1);
                    setQ(iQ);
                    setP(iP);
                    setNsteam(iNsteam);
                    setTsteam(iTsteam);
                    setrefproName(irefproName);
                    refproN=new double[irefproName.length];

                    for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                    setSteamRef(iFuel,iNFuel,iTFuel);
            }
            public steamRef(Vector[] iv,double iNsteam,double iTsteam,double iQ,double
iP) throws InappropriateArrayException{
                    if(iv[0].size()!=iv[1].size() || iv[1].size()!=iv[2].size() ) throw new
InappropriateArrayException();
                    setScenario(1);
                    refproName=new String[4];
                    refproName[0]="H2";
                    refproName[1]="CO";
                    refproName[2]="CO2";
                    refproName[3]="O2";
                    setQ(iQ);
                    setP(iP);
                    setNsteam(iNsteam);
                    setTsteam(iTsteam);
                    refproN=new double[refproName.length];

                    for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                    setrefproName(refproName);
```

```
                        setSteamRef(iv);
                }
                public steamRef(Vector[] iv,double iNsteam,double iTsteam,double iQ,double
iP,Vector irefproName) throws InappropriateArrayException{
                        if(iv.length!=3 || iv[0].size()!=iv[1].size() || iv[1].size()!=iv[2].size()
)
                        throw new InappropriateArrayException();
                        setScenario(1);
                        setQ(iQ);
                        setP(iP);
                        setNsteam(iNsteam);
                        setTsteam(iTsteam);
                        setrefproName(irefproName);
                        setSteamRef(iv);
                }


/*
*Scenario 1 ends
*/

/*
*for scenario 2 : known -fuel composition and amount  , fuel temperature , steam amount
and temperature ,
*Outlet temperature-  unknown -needed heat , product composition-
*/
                public steamRef(double iTout,fluidPoint fpfuel,fluidPoint fpsteam,double
iNsteam,double iP){
                        try{
                        setScenario(2);
                        refproName=new String[4];
                        refproName[0]="H2";
                        refproName[1]="CO";
                        refproName[2]="CO2";
                        refproName[3]="O2";
                        setTout(iTout);
                        setP(iP);
                        setNsteam(iNsteam);
                        setTsteam(fpsteam.T);
                        refproN=new double[refproName.length];

                        for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                        setSteamRef(fpfuel.getGasList(),fpfuel.getGasMoles(),fpfuel.T);

                        }catch(GException e){System.out.println("
"+e.getLocalizedMessage());}
                }

                        public steamRef(double iTout,String[] iFuel,double[][]
iNFuel,double iNsteam,
                double[] iTFuel,double iTeq,double iTsteam,double iP){
                        setScenario(2);
                        refproName=new String[4];
```

```
                refproName[0]="H2";
                refproName[1]="CO";
                refproName[2]="CO2";
                refproName[3]="O2";
                setTout(iTout);
                setP(iP);
                setNsteam(iNsteam);
                setTsteam(iTsteam);
                refproN=new double[refproName.length];

                for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                setSteamRef(iFuel,iNFuel,iTFuel);
        }
        public steamRef(double iTout,String[] iFuel,double[][] iNFuel,double
iNsteam,double[] iTFuel,double iTeq,
        double iTsteam,double iP,String[] irefproName ){
                setScenario(2);
                setTout(iTout);
                setP(iP);
                setNsteam(iNsteam);
                setTsteam(iTsteam);
                setrefproName(irefproName);
                refproN=new double[irefproName.length];

                for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
                setSteamRef(iFuel,iNFuel,iTFuel);
        }
        public steamRef(double iTout,Vector[] iv,double iNsteam,double iTeq,double
iTsteam,double iP)
         throws InappropriateArrayException{
                if(iv[0].size()!=iv[1].size() || iv[1].size()!=iv[2].size() ) throw new
InappropriateArrayException();
                setScenario(2);
                refproName=new String[4];
                refproName[0]="H2";
                refproName[1]="CO";
                refproName[2]="CO2";
                refproName[3]="O2";
                setTout(iTout);
                setP(iP);
                setNsteam(iNsteam);
                setTsteam(iTsteam);
                refproN=new double[refproName.length];

                for(int i=0;i<refproN.length;i++) refproN[i]=0.0;
           setrefproName(refproName);
                setSteamRef(iv);
        }
        public steamRef(double iTout,Vector[] iv,double iNsteam,double iTeq,double
iTsteam,double iP,Vector irefproName)
         throws InappropriateArrayException{
                if(iv[0].size()!=iv[1].size() || iv[1].size()!=iv[2].size() ) throw new
InappropriateArrayException();
```

```
                        setScenario(2);
                        setrefproName(irefproName);
                        setTout(iTout);
                        setP(iP);
                        setNsteam(iNsteam);
                        setTsteam(iTsteam);
                        setSteamRef(iv);
                }


/*
*Scenario 2 ends
*/

/*
*set steam reformer methods start
*/
                public void steTeq(double iTeq){Teq=iTeq;}
                public void setTout(double iTout){Tout=iTout;
                        Teq=Tout;
                }
                public void setQ(double iQ){ Q=iQ;}
                public void setP(double iP){P=iP;}
                public void setNsteam(double iNsteam){Nsteam=iNsteam;}
                public void setTsteam(double iTsteam){Tsteam=iTsteam;}
                public void setrefproName(Vector irefproName){
                                        vrefproName.clear();
                                        vrefproName.addAll(irefproName);

                }
                public void setrefproName(String[] irefproName){
                        //for(int i=0;i<irefproName.length;i++)
System.out.println("iref"+irefproName[i]);
                                        refproName=irefproName;
                                        //for(int i=0;i<refproName.length;i++)
System.out.println("ref"+refproName[i]);
                                        vrefproName.clear();
                                        for(int i=0;i<refproName.length;i++)
                                        vrefproName.add(refproName[i]);

                }
                public void setScenario(int ii){
                        scenario=ii;
                }
                public void setSteamRef(double[][] iNFuel){
                        setSteamRef(this.Fuel,iNFuel,this.TFuel);
                }
                public void setSteamRef(String[] iFuel,double[] iNFuel,double iTFuel){
                        //Fuel=iFuel;
                        //NFuel=iNFuel;
                        //TFuel=iTFuel;
                        vStore[0]=new Vector<String>();
                        vStore[1]=new Vector<Double>();
```

```
vStore[2]=new Vector<Double>();
steam st;
for(int i=0;i<iFuel.length;i++){
vStore[0].add(iFuel[i]);
vStore[1].add(iNFuel[i]);
vStore[2].add(iTFuel);
}
int index=0;
double NsteamTot=0;
double TsteamAvg=0;
if(!vStore[0].contains("H2O")){
vStore[0].add("H2O");
vStore[1].add(Nsteam);
vStore[2].add(Tsteam);
}
else{
index=vStore[0].indexOf("H2O");
st=new steam();
st.setUnit(true);
st.setUnit("SI");
double
n1=((Double)vStore[1].elementAt(index)).doubleValue();
double
h1=st.h(((Double)vStore[2].elementAt(index)).doubleValue(),

st.v_tp(((Double)vStore[2].elementAt(index)).doubleValue(),P));

double h2=st.h(Tsteam,st.v_tp(Tsteam,P));
//System.out.println("h1= "+h1+" n1= "+n1+" h2=
"+h2+" n2= "+Nsteam);
NsteamTot=(n1+Nsteam);
TsteamAvg=st.t_ph(P,(n1*h1+Nsteam*h2)/NsteamTot);

vStore[1].setElementAt(NsteamTot,index);
vStore[2].setElementAt(TsteamAvg,index);
}
//System.out.println(""+vStore[1].elementAt(index));
//System.out.println(""+vStore[2].elementAt(index));
for(int i=0;i<refproN.length;i++){
if(!vStore[0].contains(refproName[i])){
vStore[0].add(refproName[i]);
vStore[1].add(refproN[i]);
vStore[2].add(298.15);
}
}
//1 element is fuel reactants , 2-refproN H2O , refproN-Fuel
reformer products

/*for(int i=0;i<compeq.length;i++){
System.out.println("compeq["+i+"]= "+compeq[i]);
System.out.println("neq["+i+"][0]= "+neq[i][0]);
System.out.println("Teq["+i+"]= "+Teq[i]);
}	*/
```

```
	}
	public void setSteamRef(String[] iFuel,double[][] iNFuel,double[] iTFuel){
		Fuel=iFuel;
		NFuel=iNFuel;
		TFuel=iTFuel;
		vStore[0]=new Vector<String>();
		vStore[1]=new Vector<Double>();
		vStore[2]=new Vector<Double>();
		steam st;
		for(int i=0;i<iFuel.length;i++){
			vStore[0].add(iFuel[i]);
			vStore[1].add(iNFuel[i][0]);
			vStore[2].add(iTFuel[i]);
		}
		int index=0;
		double NsteamTot=0;
		double TsteamAvg=0;
		if(!vStore[0].contains("H2O")){
			vStore[0].add("H2O");
			vStore[1].add(Nsteam);
			vStore[2].add(Tsteam);
		}
		else{
			index=vStore[0].indexOf("H2O");
			st=new steam();
			st.setUnit(true);
			st.setUnit("SI");
			double
n1=((Double)vStore[1].elementAt(index)).doubleValue();
			double
h1=st.h(((Double)vStore[2].elementAt(index)).doubleValue(),

	st.v_tp(((Double)vStore[2].elementAt(index)).doubleValue(),P));

			double h2=st.h(Tsteam,st.v_tp(Tsteam,P));
			//System.out.println("h1= "+h1+" n1= "+n1+" h2=
"+h2+" n2= "+Nsteam);
			NsteamTot=(n1+Nsteam);
			TsteamAvg=st.t_ph(P,(n1*h1+Nsteam*h2)/NsteamTot);

			vStore[1].setElementAt(NsteamTot,index);
			vStore[2].setElementAt(TsteamAvg,index);
		}
		//System.out.println(""+vStore[1].elementAt(index));
		//System.out.println(""+vStore[2].elementAt(index));
		for(int i=0;i<refproN.length;i++){
			if(!vStore[0].contains(refproName[i])){
			vStore[0].add(refproName[i]);
			vStore[1].add(refproN[i]);
			vStore[2].add(298.15);
			}
		}
```

```
                        //1 element is fuel reactants , 2-refproN H2O , refproN-Fuel
reformer products

                        /*for(int i=0;i<compeq.length;i++){
                                System.out.println("compeq["+i+"]= "+compeq[i]);
                                System.out.println("neq["+i+"][0]= "+neq[i][0]);
                                System.out.println("Teq["+i+"]= "+Teq[i]);
                        }               */
                }
                public void setSteamRef(Vector[] iv)throws InappropriateArrayException{

                        //for(int i=0;i<3;i++) vStore[i]=new Vector();
                        vStore[0]=new Vector(iv[0]);
                        vStore[1]=new Vector(iv[1]);
                        vStore[2]=new Vector(iv[2]);
                        steam st;
                        int index=0;
                        double NsteamTot=0;
                        double TsteamAvg=0;
                        if(!vStore[0].contains("H2O")){
                                vStore[0].add("H2O");
                                vStore[1].add(Nsteam);
                                vStore[2].add(Tsteam);
                        }
                        else{
                                index=vStore[0].indexOf("H2O");
                                st=new steam();
                                st.setUnit(true);
                                st.setUnit("SI");
                                double
n1=((Double)vStore[1].elementAt(index)).doubleValue();
                                double
h1=st.h(((Double)vStore[2].elementAt(index)).doubleValue(),

                st.v_tp(((Double)vStore[2].elementAt(index)).doubleValue(),P));

                                double h2=st.h(Tsteam,st.v_tp(Tsteam,P));
                                //System.out.println("h1= "+h1+" n1= "+n1+" h2=
"+h2+" n2= "+Nsteam);
                                NsteamTot=(n1+Nsteam);
                                TsteamAvg=st.t_ph(P,(n1*h1+Nsteam*h2)/NsteamTot);

                                vStore[1].setElementAt(NsteamTot,index);
                                vStore[2].setElementAt(TsteamAvg,index);
                        }
                        //System.out.println(""+vStore[1].elementAt(index));
                        //System.out.println(""+vStore[2].elementAt(index));
                        //System.out.println(""+vrefproName.size());
                        for(int i=0;i<vrefproName.size();i++){
                                if(!vStore[0].contains(vrefproName.get(i))){
                                        vStore[0].add(vrefproName.get(i));
                                        vStore[1].add(0.0);
```

```
                                        vStore[2].add(298.15);

                                }

                        }

                        if(vStore[0].size()!=vStore[1].size()) throw new
InappropriateArrayException();
                        //1 element is fuel reactants , 2-refproN H2O , refproN-Fuel
reformer products
                        /*for(int i=0;i<compeq.length;i++){
                                System.out.println("compeq["+i+"]= "+compeq[i]);
                                System.out.println("neq["+i+"][0]= "+neq[i][0]);
                                System.out.println("Teq["+i+"]= "+Teq[i]);
                        }*/
                }
/*
*set steam reformer methods end
*/
/*
*add methods
*/
                public void addpro(String igas)throws GasAlreadyDefinedException{
                        vrefproName.add(igas);
                        if(!vStore[0].contains(igas)){
                                        vStore[0].add(igas);
                                        vStore[1].add(0.0);
                                        vStore[2].add(298.15);

                        }
                        else throw new GasAlreadyDefinedException();
                }

/*
*add methods end
*/



                public void calcEq(){
                        gibbs gb;
                        String[] compeq=new String[vStore[0].size()];
                        double[][] neq=new double[vStore[1].size()][1];
                        TFuelfinal=new double[vStore[2].size()];
                        int ei=0;
                        for(int i=0;i<compeq.length;i++){
                                compeq[i]=(String)vStore[0].elementAt(i);

                neq[i][0]=((Double)vStore[1].elementAt(i)).doubleValue();

                TFuelfinal[i]=((Double)vStore[2].elementAt(i)).doubleValue();
                        }
                /*      for(int i=0;i<compeq.length;i++){
```

```
			System.out.println("compeq["+i+"]= "+compeq[i]);

		}
		for(int i=0;i<compeq.length;i++){
			System.out.println("neq["+i+"][0]= "+neq[i][0]);

		}
		for(int i=0;i<compeq.length;i++){

			System.out.println("TFuelfinal["+i+"]= "+TFuelfinal[i]);
		}		*/
		if(scenario==1){

			gb=new gibbs(compeq,neq,TFuelfinal,Teq,P,nmax);
			resultName=new String[gb.gs.length];
			resultN=new double[gb.n.length];
			Tout=gb.Tad(Q);
			for(int i=0;i<gb.n.length;i++){
				resultName[i]=gb.gs[i].Formula;
				resultN[i]=gb.n[i][0];
				System.out.println(gb.gs[i].Formula+"
"+gb.n[i][0]);

			 }
			System.out.println("Tout= "+Tout);
			//System.out.println("Q= "+gb.delQ_T(Tout));
			neq0=gb.n;
		}
		else if(scenario==2){
		//System.out.println("Tout= "+Tout);

	//System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
			gb=new gibbs(compeq,neq,TFuelfinal,Tout,P,nmax);
			resultName=new String[gb.gs.length];
			resultN=new double[gb.n.length];
			gb.getEq();
			for(int i=0;i<gb.n.length;i++){
				resultName[i]=gb.gs[i].Formula;
				resultN[i]=gb.n[i][0];
				System.out.println(gb.gs[i].Formula+"
"+gb.n[i][0]);
			 }
			Qneeded=gb.delQ();
			System.out.println("Q= "+gb.delQ());
			//
System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
			 //System.out.println("Tout= "+Tout);
			 neq0=gb.n;
		}
		else{
			System.out.println("Scenario error in steamRef.java");
```

```
                        neq0=neq;
                }
                compeq0=compeq;
        }
        public fluidPoint getOutlet(){
                try{
                fpout=new fluidPoint(resultName,resultN,Tout,P);
                        return fpout;
                }catch(java.io.IOException e){System.out.println("IOException @
AutoThermalRef.java ");}
                catch(GException ge){System.out.println("
"+ge.getLocalizedMessage());}
                return null;
        }



}
```

Ek-2

LU-LT-ÜST ÜÇGEN MATRİS- GAUSS (DOLİTTLE) 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 (Çoban, 2001-2008)

$$[L]=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 & 0 \\ l_{41} & l_{42} & l_{43} & 1 & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & 1 \end{bmatrix}$$

$$[U]=\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} \\ 0 & 0 & u_{33} & u_{34} & u_{35} \\ 0 & 0 & 0 & u_{44} & u_{45} \\ 0 & 0 & 0 & 0 & u_{55} \end{bmatrix}$$

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. (Çoban, 2001-2008)

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. x3=(x1+x2)/2 aha sonra fonksiyonu lineer fonksiyona dönüştüren özel bir faktörü eQ faktörünü çözer. Daha açık yazacak olur isekf(x_1)-2f(x_3) e^Q + f(x_2) e^{2Q} = 0 fonksiyonundan e^Q faktörünü çözer. (Çoban, 2001-2008)

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. (Çoban, 2001-2008)

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

Denklem ikinci dereceden olduğundan ikinci derece yaklaşım sağlar. Aynı zamanda bulunan kök her zaman x_1 ve x_2 arasında yer alacaktır. (Çoban, 2001-2008)

KÖK SINIRLARININ BELİRLENMESİ

Bir değişkenli fonksiyonlar için (a,b) bölgesinde en az bir kök vardır diyebilmemiz için üç 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 geliştirilmiştir.(Çoban, 2001-2008)

REFERANSLAR

Cengel.Y. Boles. M. (2003). *Thermodynamics: An Engineering Approach* (6th ed.).

Brock, J. R. (1955). *American Instıtute of Chemıcal Engıneers Journal* v 21,(102).

Gomez-Nieto, M. T. (1978). *Industrial Engineering. Chemical. Fundamentals, 17*(45).

Gordon, S. M., B. J. (1994). *Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications* (No. NASA Reference Publication-1311): NASA.(1-30)

Hakim, D. I. (1971). *Ind. Eng. Chem. Fundam., 10*(174).

Keenan, H. (1969). *Steam Tables* (1 ed.): Wiley Interscience.

Pitzer, K. S. (1957). *J. AM. Chem. Soc., 79*(2369), 12-17.

Prausnitz, R. C. e. a. (1987). The Properties of Gases and Liquids (4 ed.): McGraw-Hill.

Çoban, M. Turhan, Ege Üniversitesi Makine Mühendisliği Bölümü Yüksek Lisans Optimizasyon ve Genetik Algoritmalar Ders Notları 2004

Çoban, M. Turhan, Ege Üniversitesi Makine Mühendisliği Bölümü Yüksek Lisans İleri Termodinamik Ders Notları 2005

Çoban, M. Turhan, Ege Üniversitesi Makine Mühendisliği Bölümü Yüksek Lisans Yakıt Pilleri Ders Notları 2005

Çoban, M. Turhan, Java Programlama Dili Örnekleri İle Sayısal Çözümleme Notları, 2005,

ÖZGEÇMİŞ

Abdullah Sermest Tazebay, 17 Mayıs 1980 yılında Gaziantep'de doğmuştur. İlk, orta ve lise öğrenimini Gaziantep'de tamamladı. Ege Üniversitesi Makine Mühendisliği bölümünde 2005 yıllında mezun oldu. Aynı yıl Ege Üniversitesi Fen Bilimleri Enstitüsünde yüksek lisansa başladı.

2005-2007 yılları arsında Ege Üniversitesi Makine Mühendisliği bölümünde Yrd. Doç. Dr. M. Turhan Çoban tarafından yürütülmekte olan "Doğalgaz Yakıtlı Katı Oksitli Yakıt Pili Sisteminin Geliştirilmesi" adlı DPT projesinde çalışmıştır.