W3 NUMERICAL ANALYSIS 2019-2020 FALL

ROOT FINDING f(x)=0 x=?

CLASS EXERCISES

Class exercises will be completed and graded in class

EX 1 Horner's Method (uses synthetic division) Roots of degree n polynomial

$$f(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ..$$

Java version c.java class contains complex class and complex algebra

```
 import javax.swing.*;
import java.util.*;
public class horner
{
c CA[];//complex polynomial coefficient
public static c f_poly(c c1[],c x)
{
// this function calculates the value of
// c least square curve fitting function
int n=c1.length;
c ff;
if(n!=0.0)
  { ff=new c(c1[n-1]);
  for(int i=n-2;i>=0;i--)
   { ff=c.p(c.p(ff,"*",x),"+",c1[i]); }
  }
else
  ff=new c(0,0);
return ff;
}
public double[] toDouble()
{int n=CA.length;
 double d[]=new double[n];
 for(int i=0;i<n;i++)
    {   d[i]=c.toDouble(CA[i]);}
    return d;}
public c horner(double a[], c x)
{   int n=a.length;
                c c1[]=new c[n];
    for(int i=0;i<n;i++) c1[i]=new c(a[i]);
    return horner(c1,x);}
public c horner(c c3[], c x)
{   int n=c3.length;
    int iter=0;
    int nmax=200;
    boolean b=true;
    int j=0;
                c c1[]=new c[n];
                int n1=n-1;
                c cp[]=new c[n1];
                c1[n-1]=new c(c3[n-1]);
    for(int i=n-2;i>=0;i--)
    {   c c2=new c(c3[i]);
                   c1[i]=c.p(c2,"+",c.p(x,"*",c1[i+1]));
    }
                c P=c1[0];
                for(int i=n1-1;i>=0;i--)
                {cp[i]=c1[i+1];}
                CA=cp;
                c Q=f_poly(cp,x);
                //Newton's method
                x=c.p(x,"-",c.p(P,"/",Q));//x=x-P/Q
                iter++;
                if(c.abs(P)<1.0e-20 || c.abs(P)==0 ||iter>nmax)
                {return x;}
                else
                {
                while(b)
                {
                c1[n-1]=new c(c3[n-1]);
    for(int i=n-2;i>=0;i--)
```

```
    {   c c2=new c(c3[i]);
                    c1[i]=c.p(c2,"+",c.p(x,"*",c1[i+1]));
                }
                P=c1[0];
                for(int i=n1-1;i>=0;i--)
                {cp[i]=c1[i+1];}
                CA=cp;
    Q=f_poly(cp,x);
                if(c.abs(P)<1.0e-20||iter>nmax)
                {b=false;
                 return x;
                }
                //Newton's method
                x=c.p(x,"-",c.p(P,"/",Q));
                iter++;
   }
}
  return x;
}

public String toString()
{int n=CA.length;
 String s="";
 for(int i=n-1;i>=0;i--) {s+=CA[i]+" ";}
 return s;
}
public static String toString(c x[])
{int n1=x.length;
 String s="";
 for(int i=n1-1;i>=0;i--) {s+="x["+i+"] = "+x[i]+"\n";}
 return s;
}
public c[] root(double a[])
{   int n=a.length;
    c y[]=new c[n-1];
                c x=new c(0.9458647);
                c x1=horner(a, x);
                int i=n-2;
    y[i]=x1;
    i--;
    int nn=CA.length;
    while(i>=0)
    {  if(nn!=CA.length)
      {c D[]=new c[nn];
      for(int j=0;j<nn;j++) {D[j]=CA[j];}
      CA=D;
       }
     if(nn>3)
     {c z=horner(CA,x);

      y[i]=z;
     i--;
     nn--;
     }
     else if(nn==3)
     { c z[]=new c[2];
      c z1=new c(CA[0]);
     c ac4=c.p(z1,"*",CA[2]);
     ac4=c.p(ac4,"*",4.0);
     z1=new c(CA[1]);
     c b2=c.p(z1,"*",z1);
     c DD=c.sqrt(c.p(b2,"-",ac4));
     z1=new c(CA[1]);
     z[0]=c.p(DD,"-",z1);
     z[0]=c.p(z[0],"/",2.0);
     z[0]=c.p(z[0],"/",CA[2]);
     z[1]=c.p(DD,"+",CA[1]);
     z[1]=c.p(z[1],"/",2.0);
     z[1]=c.p(z[1],"/",CA[2]);
     z[1]=c.p(z[1],"*",-1.0);
     y[i]=z[0];
                  y[i-1]=z[1];
     i-=2;
     nn-=2;
     }
```

```
    else if(nn==2)
    {c z;
    c z1=new c(a[0]);
                z=c.p(z1,"*",-1.0);
                z=c.p(z,"/",a[1]);
                y[i]=z;
                i-=2;
    nn-=2;
                }
    }
   return y;
}
    public static double[] enterdata(String s)
    {
    String s1=JOptionPane.showInputDialog(s);
    StringTokenizer token=new StringTokenizer(s1);
    int n=token.countTokens()-1;
    int m=n+1;
    double a[]=new double[m];
    int j=0;
    while(token.hasMoreTokens())
    {
    Double ax=new Double(token.nextToken());
    a[j++]=ax.doubleValue();
    }
    return a;
    }
public static void calculate_Horner()
{   horner h=new horner();
            String s="enter coefficients of the n degree polynomial a[0] a[1]..a[n] y=a[0]+a[1]*x+a[2]*x^2+a[3]*x^3+..+a[n]*x^n ";
   double [] x0=enterdata(s);
   int n=x0.length;
   String s1[]=new String[n];
   String s2=" coefficients of the n degree polynomial y=a[0]+a[1]*x+a[2]*x^2+a[3]*x^3+..+a[n]*x^n ";
   for(int i=0;i<n;i++)
   {s1[i]="a["+i+"]";}

   Text.print(x0,s1,s2);
   JOptionPane.showMessageDialog(null,toString(h.root(x0)),
   "roots of the nth degree polynomial by Horner's synthetic division & Newton-Raphson method : ",JOptionPane.PLAIN_MESSAGE);

}
public static void main(String arg[])
{   calculate_Horner();}
}
```

Horner method python version

```
from math import *

def f_poly(c1,x):
     # this function calculates the value of
     # c least square curve fitting function
     n=len(c1);
     if n!=0.0:
        ff=c1[n-1];
        for i in range(n-2,-1,-1):
           ff=ff*x+c1[i];
     else:
        ff=0,0;
     return ff;

def horner(a):
  # roots of a polynomial by using synthetic division process
     n=len(a);
     x1=complex(1.56435,0.1);
     n1=n;
     nmax=150;
     tolerance=1.0e-10;
     n1=n+1;
     ref=3.1;
     y=[complex(0.0,0.0) for i in range(n1-2)];
     n2=n1-2;
     for k in range(0,n2):
           n=len(a);
```

```
            x=x1;
            xold=3.4878;
            ref=abs(xold-x);
            b=[complex(0.0,0) for i in range(n)];
            c=[complex(0.0,0.0) for i in range(n-1)];
            b[n-1]=a[n-1];
            ii=0;
            while ref>tolerance and ii<nmax:
              for i in range(n-2,-1,-1):
                 b[i]=a[i]+b[i+1]*x;
              for i in range(n-2,-1,-1):  c[i]=b[i+1];
              Q=f_poly(c,x);
              P=b[0];
              x=x-P/Q;
              ref=abs(xold-x);
              ii=ii+1;
              xold=x;
            a=c;
            n=len(a);
            y[k]=x;
     return y;

a=[1,3,3,1];
print(horner(a));
```

EX2 Halley root finding method

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

Java version

```
// ROOT FINDING Halley method
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import javax.swing.*;
public class Halley2
{public static double Halley(if_x f,double x)
{
int nmax=500;
double tolerance=1.0e-8;
double fx,dfx,fxm,dfxm,d2fx;
for(int i=0;i<nmax;i++)
{  fx=f.func(x);
   dfx=f.dfunc(x);
   d2fx=f.dfunc(x,2);
   x-=2.0*fx*dfx/(2.0*dfx*dfx-fx*d2fx);
  if(Math.abs(fx)<tolerance) { return x;}
}
JOptionPane.showMessageDialog(null,"Maximum number of iteration is exceeded in Halley method\n"+
   " results may not be valid","MAXIMUM ITERATION WARNING",JOptionPane.WARNING_MESSAGE);
return x;
}
public static void main(String arg[])
{    if_x f=x->x*x-2;
     double x=Double.parseDouble(JOptionPane.showInputDialog(" enter lower limit of the function a : "));
     double r= Halley(f,x);
     JOptionPane.showMessageDialog(null," root  : "+r+"\nFunction value : "+f.func(r));
     Plot p=new Plot(f,0,10);
     p.plot();
}
}
```

Halley method python version

```
#halley.py
from math import *
from graphics import *
from f_x import *;

class f1(f_x):
  def func(self,x):
     y=x*x+2;
```

```
    return y;

def halley(f,x):
  # halley root finding method
  nmax=100
  tolerance=1.0e-10
  for i in range(0,nmax):
     fx= f.func(x);
     dfx=f.dfunc(x,1);
     d2fx=f.dfunc(x,2);
     x=x-2.0*fx*dfx/(2.0*dfx*dfx-fx*d2fx)
     print("fx",fx,"dfx",dfx,"x=",x)
     if abs(fx)<tolerance: return x
  return x

def read(s1,s2):
    # reads x1 and x2 values as float then
    # converts to a float array and returned
    s="read data "+s1+" and "+s2+" as float push mouse to exit";
    win = GraphWin(s, 450, 100)
    win.setCoords(0.0, 0.0, 3.0, 4.0)
    # Draw the interface
    Text(Point(1,2), s1).draw(win)
    Text(Point(1,1),s2).draw(win)
    i1 = Entry(Point(2,2), 20)
    i1.setText("")
    i1.draw(win);
    i2 = Entry(Point(2,1), 20)
    i2.setText("")
    i2.draw(win);
    # wait for a mouse click
    win.getMouse()
    # convert input
    x1 = float(i1.getText());
    x2 = float(i2.getText());
    x=[x1,x2];
    return x;
def main():
    x1=read("x1:","x2:");
    x=complex(x1[0],x1[1]);
    f=f1();
    r=halley(f,x);
    print(" ROOT  : ",r,"\nFUNCTION VALUE : ",f.func(r));
main();
```

EX3 Olver's root finding method

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

Write the code in your selected language and find the root of $f(x) = x^3 + 3.6x - 36.4$

HOMEWORK EXERCISES

Homework exercises will be done at home and will bring to next Thursday class printed no late exercises will be excepted. Each code should include student name id#, code plus results should be given. Homeworks will be accepted in written format plus a computer copy in pdf format will be sent to numerical_analysis@turhancoban.com adress your file name should be "group"+"week#"+studentname+studentid#.pdf

A_W1_turhan_coban_0101333.pdf

B_W3_ali_veli_02335646.pdf

W3HW1

W1HW1 : function f(x) $= x * x - 5 * x + 1$ is given find the root by using Horner's method of synthetic division

a) by hand (finding just one root is enough)

b) by using java program
c) by using python program

W3HW2: function f(x) $= x * x - 5 * x + 1$ is given find the root by using Halley's method by hand (finding just one root is enough)

a) by hand (finding just one root is enough)
b) by using java program
c) by using python program

W3HW3 : function f(x) $= \sin(x)$ is given find the root in the range of $0.2 \leq x \leq 2.5$ by using

a) Olver's method by hand
b) Olver's method by computer program (you can select your language)