COMPUTING ALL THE REAL AND COMPLEX ROOTS OF A POLYNOMIAL EQUATION OF DEGREE 'N' WITH REAL COEFFICIENTS BASED UPON BAIRSTOW'S METHOD USING C PROGRAMMING.

Dipak Majhi
Bachelor of Technology (B.tech) in Electronics and Communication Engineering, Kalyani Government Engineering College, Nadia-741235,
West Bengal, India.

Abstract — This program uses Bairstow's method to find the real and complex roots of a polyomial with real coefficients. There are several reasons for developing a routine based Bairstow's method. It is sometimes the case that all of the roots of a polynomial with real coefficients are desired. Bairstow's method provides an iterative process for finding both the real and complex roots using only real arithmetic.

Further, since it is based on Newton's Method for a systemof two nonlinear equations in two unknowns, it has the rapid convergence property of Newton's method for systems of equations. The major drawback of this method is that it sometimes fail to converge. This is because it is difficult to find an initial starting guess which satisfies the strict conditions necessary to assure convergence. When these conditions are not satisfied, the sequence of approximations may jump away from the desired roots or may iterate away from the roots indefinitely.

The program obtains an initial approximation to two roots of the polynomial and uses these two roots to estimate initial starting values of the Bairstow interaction. These values are the coefficients r and s of an approximate quadratic factor x^2 + rx + s, of the given polynomial. The given polynomial will go through synthetic division by the quadratic function which will eventually give all the real or complex or both in pairs depending upon the degree of the polynomial.

Keywords— Bairstow, c, complex, computing, coefficients, convergence, degree, division, equation, iterative, initial, method, newton, polyomial, programming, roots, real, synthetic.

INTRODUCTION

Bairstow Method is an iterative method used to find both the real and complex roots of a polynomial. It is based on the idea of synthetic division of the given polynomial by a quadratic function and can be used to find all the roots of a polynomial. Given a polynomial say,

$$f_n(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n \quad (1)$$

Bairstow's method divides the polynomial by a quadratic function.

$$(\mathbf{x^2 + rx + s})$$

Now the quotient will be a polynomial,

$$f_{n-2}(x) = b_2 + b_3 x + b_4 x^2 + \dots + b_{n-1} x^{n-3} + b_n x^{n-2} \quad \dots(2)$$

and the remainder is a linear function,

$$R(x)=b_1(x+r)+b_0$$

We use a form of Horner's method which avoids the certain introduction of error into the calculation of b[0] due to the loss of significant digits. For n$\geq$3, the iterative scheme to calculate the coefficients is,

$$b_n=a_n$$

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

$$b_i=a_i+rb_{i+1}+sb_{i+2} \quad \text{for } i= n\text{-}2 \text{ to } 1 \quad ...(3)$$

$$b_0=a_0+sb_2$$

If we calculate the partial derivatives of each of the equations (3) with respect to r , we have the iterative scheme,

$$c_n=b_n$$

$$c_{n-1}=b_{n-1}+r*c_n$$

$$c_i=b_i+r*c_{i+1}+s*c_{i+2} \quad \text{for } i= n\text{-}2 \text{ to } 2$$

$$c_1=s*c_3$$

If we calculate the partial derivatives of each of the equations (3) with respect to s , we have the iterative scheme,

$$d_n=b_n$$

$$d_{n-1}=b_{n-1}+r*d_n$$

$$d_i=b_i+r*d_{i+1}+s*d_{i+2} \quad \text{for } i= n\text{-}2 \text{ to } 3$$

$$d_2=b_2+s*d_4$$

Δr and Δs can be written as :

$$\Delta r == \frac{\mathrm{b}[0]*\mathrm{d}[3]-\mathrm{b}[1]*\mathrm{d}[2]}{(\mathrm{d}[2]+\mathrm{r}*\mathrm{d}[3])*\mathrm{d}[2]-\mathrm{s}*\mathrm{d}[3]*\mathrm{d}[3]}$$

$$\Delta s = \frac{\mathrm{b}[1]*\mathrm{s}*\mathrm{d}[3]-\mathrm{b}[0]*(\mathrm{d}[2]+\mathrm{r}*\mathrm{d}[3])}{(\mathrm{d}[2]+\mathrm{r}*\mathrm{d}[3])*\mathrm{d}[2]-\mathrm{s}*\mathrm{d}[3]*\mathrm{d}[3]}$$

These values can be used to improve initial approximation by,

$$p=r+\Delta r \quad \text{and} \quad q=s+\Delta s$$

At each step, an approximate error in r and s can also be found as:

$$|e_r|=|\frac{\Delta r}{r}|*100\%$$

$$|e_s|=|\frac{\Delta s}{s}|*100\%$$

When both estimates are less than the precision value, the roots are then found to be:

$$x=\frac{-r\pm\sqrt{r^2-4st}}{2t}$$

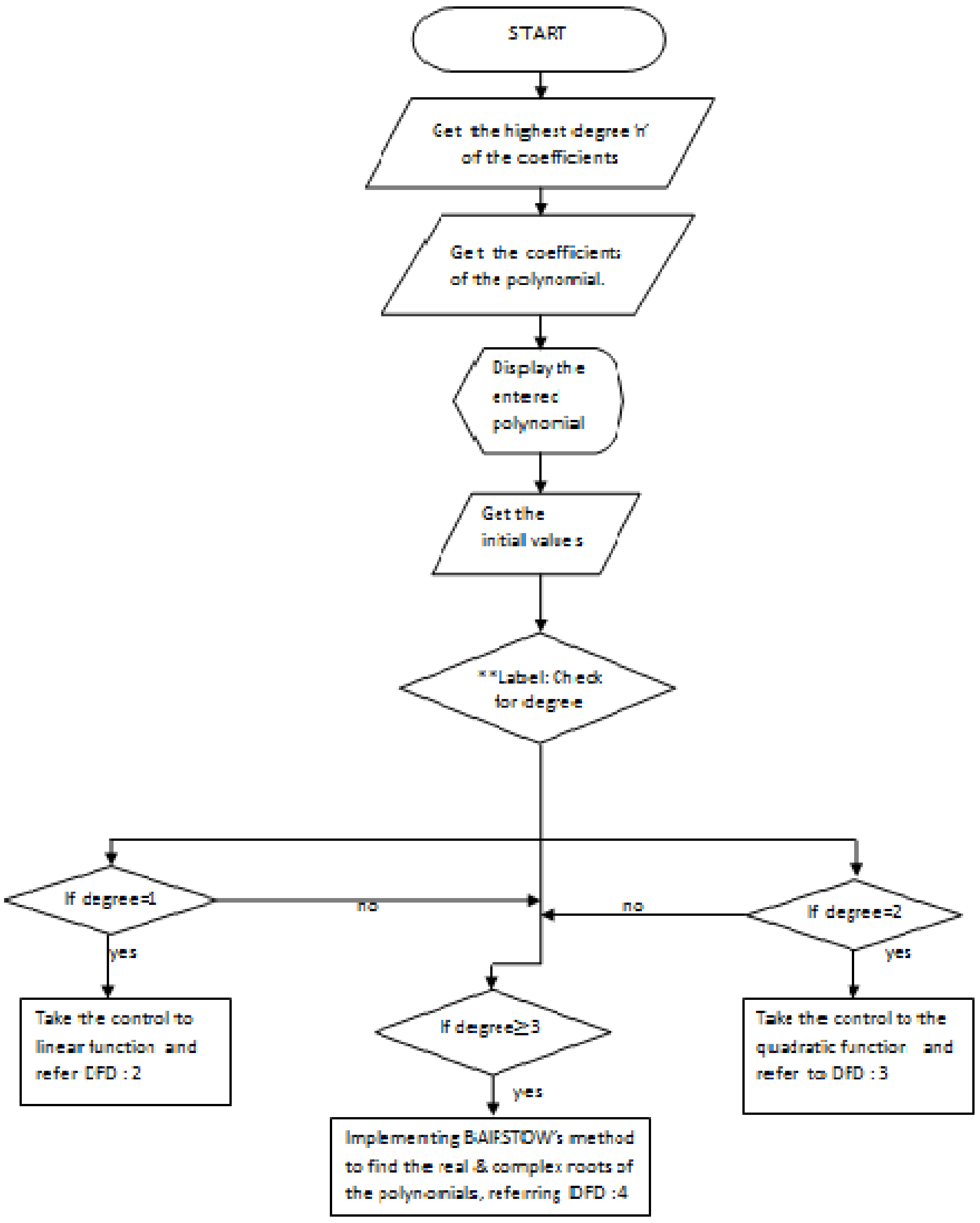
and the procedure continues for the polynomial $f_{n-2}(x)$. If f(x) is a first-order polynomial, the single real root can be evaluated simply as x=±s/r.

If we want to find all the roots of $f_n(x)$ then at this point we have the following three possibilities:

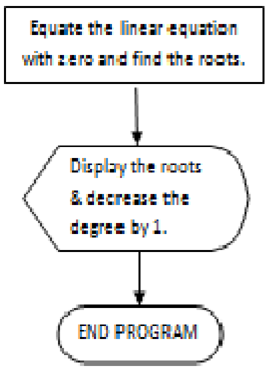
1. If the quotient polynomial $f_{n-2}(x)$ is a third (or higher) order polynomial then we can again apply the Bairstow's method to the quotient polynomial. The previous values of (r,s) can serve as the starting guesses for this application.
2. If the quotient polynomial $f_{n-2}(x)$ is a quadratic function then enter the values into the quadratic function by calling it to obtain the remaining two roots of $f_n(x)$.
3. If the quotient polynomial $f_{n-2}(x)$is a linear function say ax+b=0 then the remaining single root is given by x= - s/r.

Data flow diagram to compute all the real and complex roots of a polynomial equation of degree 'n' with real coefficients based upon Bairstow's method using C programming :

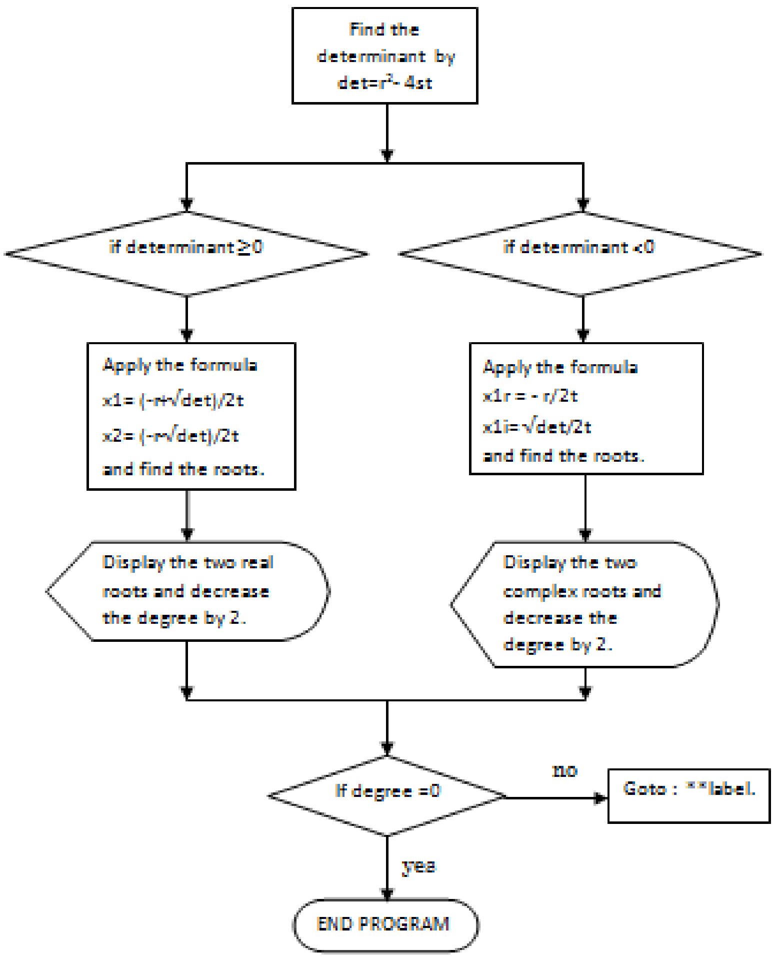
Data flow diagram : 1


START
Get the highest degree 'n' of the coefficients
Get the coefficients of the polynomial.
Display the entered polynomial
Get the initial values
**Label: Check for degree
If degree=1
no
yes
Take the control to linear function and refer DFD : 2
no
If degree=2
yes
Take the control to the quadratic function and refer to DFD : 3
If degree≥3
yes
Implementing BAIRSTOW's method to find the real & complex roots of the polynomials, referring DFD :4

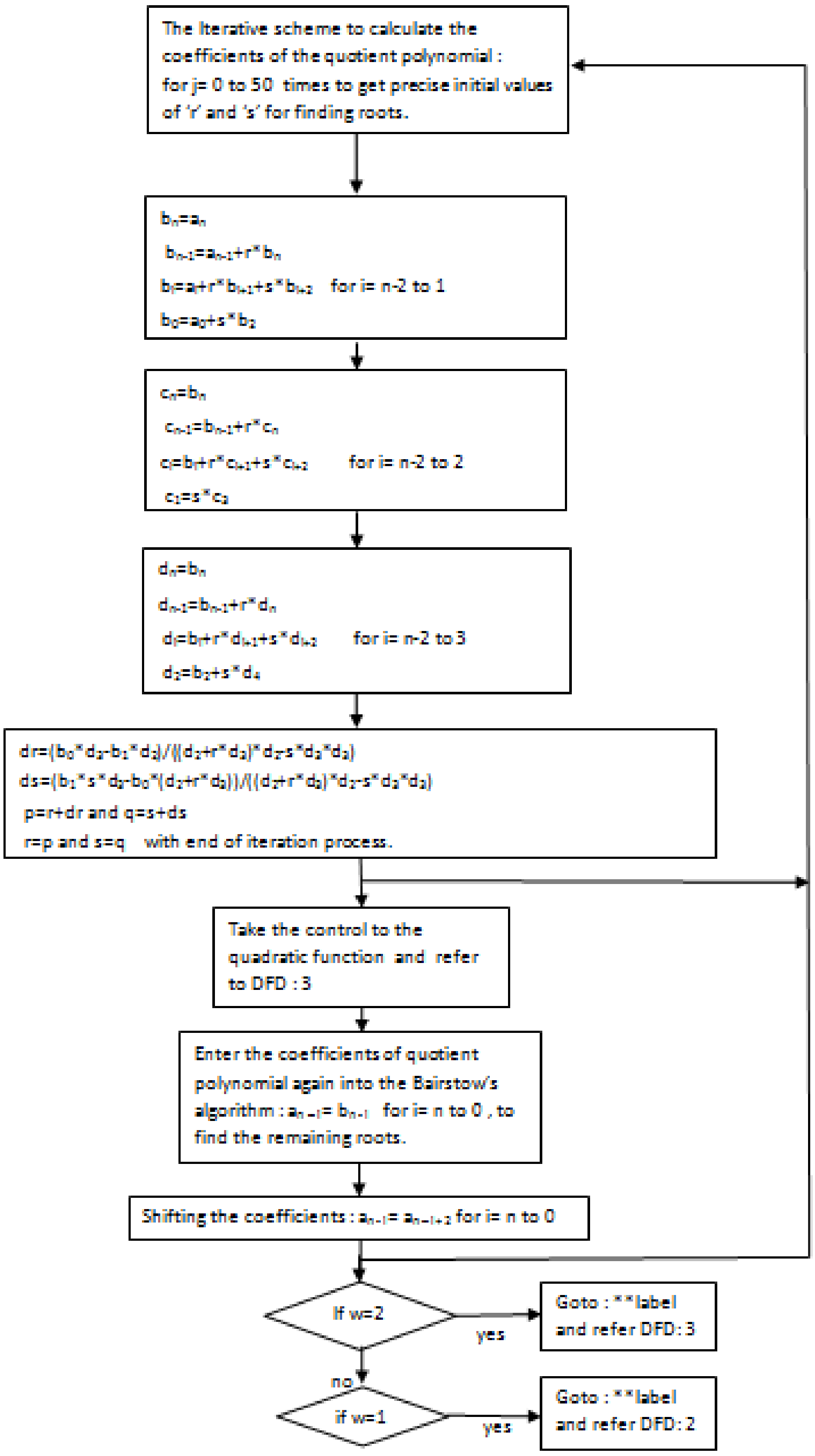
Data flow diagram : 2


Equate the linear equation with zero and find the roots.
Display the roots & decrease the degree by 1.
END PROGRAM

Data flow diagram : 3


Find the determinant by det=r²- 4st
if determinant ≥0
if determinant <0
Apply the formula x1= (-r+√det)/2t x2= (-r-√det)/2t and find the roots.
Apply the formula x1r = - r/2t x1i= √det/2t and find the roots.
Display the two real roots and decrease the degree by 2.
Display the two complex roots and decrease the degree by 2.
If degree =0
no
Goto : **label.
yes
END PROGRAM

Data flow diagram : 4


The Iterative scheme to calculate the coefficients of the quotient polynomial : for j= 0 to 50 times to get precise initial values of 'r' and 's' for finding roots.
$b_n=a_n$
$b_{n-1}=a_{n-1}+r*b_n$
$b_i=a_i+r*b_{i+1}+s*b_{i+2}$ for i= n-2 to 1
$b_0=a_0+s*b_2$
$c_n=b_n$
$c_{n-1}=b_{n-1}+r*c_n$
$c_i=b_i+r*c_{i+1}+s*c_{i+2}$ for i= n-2 to 2
$c_1=s*c_3$
$d_n=b_n$
$d_{n-1}=b_{n-1}+r*d_n$
$d_i=b_i+r*d_{i+1}+s*d_{i+2}$ for i= n-2 to 3
$d_2=b_2+s*d_4$
$dr=(b_0*d_3-b_1*d_2)/((d_2+r*d_3)*d_2-s*d_3*d_3)$
$ds=(b_1*s*d_3-b_0*(d_2+r*d_3))/((d_2+r*d_3)*d_2-s*d_3*d_3)$
p=r+dr and q=s+ds
r=p and s=q with end of iteration process.
Take the control to the quadratic function and refer to DFD : 3
Enter the coefficients of quotient polynomial again into the Bairstow's algorithm : $a_{n-i}= b_{n-i}$ for i= n to 0 , to find the remaining roots.
Shifting the coefficients : $a_{n-i}= a_{n-i+2}$ for i= n to 0
If w=2
yes
Goto : **label and refer DFD: 3
no
if w=1
yes
Goto : **label and refer DFD: 2

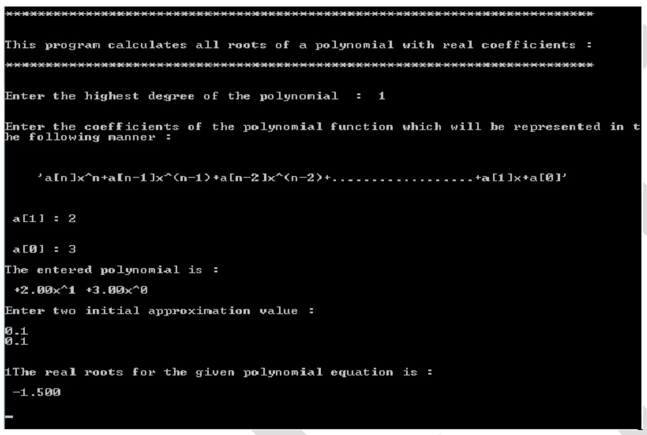
'C' coding to compute all the real and complex roots of a polynomial equation of degree 'n' with real coefficients based upon Bairstow's method using C programming:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
void linear(double a0,double a1)
{
    double e=-a0/a1;
    printf("\n\n1The real roots for the given polynomial equation is :\n\n %.3lf  \n\n",e);
}
void quadratic(double t,double r,double s)
{
       double det,x1,x2,x1r,x1i;
       det=(r*r)-(4*s*t);

if(det>=0)
       {
       x1= (-r+sqrt(det))/2*t;
       x2= (-r-sqrt(det))/2*t;
       printf("\n\n2The real roots for the given polynomial equation are :\n\n %.3lf , %.3lf \n\n",x2,x1);
       }
       else
       {
       x1r=-r/2;
       x1i=sqrt(fabs(det))/2;
       printf("\n\n2The complex roots for the given polynomial equation are :\n\n %.3lf + %.3lf j and %.3lf -
%.3lf j \n\n",x1r,x1i,x1r,x1i);
       }
}
int main()
{
   int w,j,i,n;
   double t,det,p,q,x1=0.00,x2=0.00,x1r,x1i,r,s,ds,dr,
            a[100],b[100],c[100],d[100];
            b[4]=0,b[3]=0;
   printf("\n****************************************
*********************************\n");
   printf("\n\nThis program calculates all roots of a   polynomial with real coefficients : \n");
   printf("\n****************************************   **********************************\n\n");
   printf("\nEnter the highest degree of the polynomial  :  ");
   scanf("%d",&n);
   printf("\n\nEnter the coefficients of the polynomial function which will be represented in the following
manner :\n\n");
   printf("\n\n    'a[n]x^n+a[n-1]x^(n-1)+a[n-2]x^(n-2)+.................+a[1]x+a[0]' \n\n");
   for(i=n;i>=0;i--)
   {
        printf("\n\n a[%d] : ",i);
        scanf("%lf",&a[i]);
   }
   printf("\nThe entered polynomial is : \n\n");
   for(i=0;i<=n;i++)
   if(a[n-i]>=0)
   printf(" +%.2lfx^%d",a[n-i],n-i);
   else
   printf(" %.2lfx^%d",a[n-i],n-i);
   printf("\n\nEnter two initial approximation value : \n\n");
   scanf("%lf  %lf",&r,&s);
   w=n;
   if(w==1)
     {
       linear(a[0],a[1]);
       w--;
     }
   if(w==2)
     {
       quadratic(a[2],a[1],a[0]);
       w=w-2;
     }
   while(w>=3)
   {
    for(j=1;j<=50;j++)
       {
           b[n]=a[n];
           b[n-1]=a[n-1]-r*b[n];
           for(i=n-2;i>=1;i--)
           {
            b[i]=a[i]-r*b[i+1]-s*b[i+2];
           }
           b[0]=a[0]-s*b[2];

           c[n]=b[n];
           c[n-1]=b[n-1]-r*c[n];
           for(i=n-2;i>=2;i--)
           {
             c[i]=b[i]-r*c[i+1]-s*c[i+2];
           }
           c[1]=-s*c[3];

           d[n]=b[n];
           d[n-1]=b[n-1]-r*d[n];
           for(i=n-2;i>=3;i--)
           {
             d[i]=b[i]-r*d[i+1]-s*d[i+2];
           }
           d[2]=b[2]-s*d[4];

           dr=(b[0]*d[3]-b[1]*d[2])/((d[2]-r*d[3])*d[2]+s*d[3]*d[3]);
           ds=(-b[1]*s*d[3]-b[0]*(d[2]-r*d[3]))/((d[2]-r*d[3])*d[2]+s*d[3]*d[3]);
           p=r-dr;
           q=s-ds;
           r=p;
           s=q;
        }

        t=1;
        quadratic(t,r,s);
        w=w-2;
        for(i=n;i>=0;i--)
        {
           a[n-i]=b[n-i];
        }
        for(i=n;i>=0;i--)
        {
           a[n-i]=a[n-i+2];
        }
       }

   if(w==2)
    {
       quadratic(b[4],b[3],b[2]);
       w=w-2;
    }

 if(w==1)
    {
      linear(b[2],b[3]);
      w--;
    }
  getch();
}
```

RESULTS :

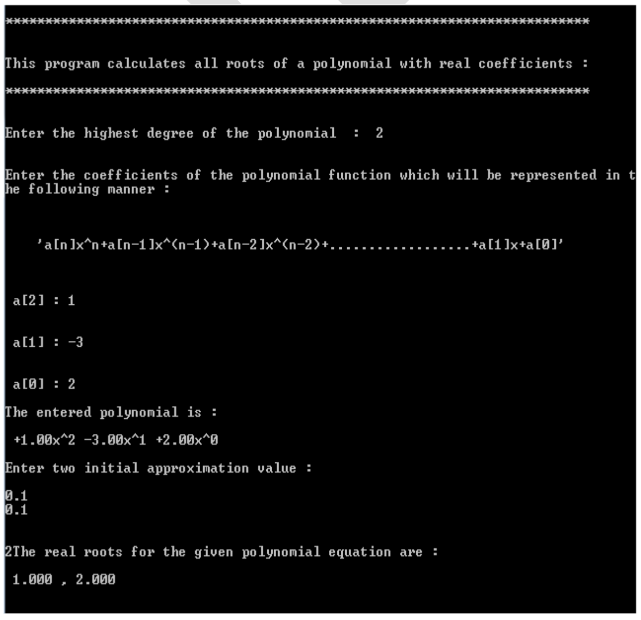
This code will be able to give the roots for almost 100th degree polynomial , though its checked upto 9th power only,

Output for degree 1:



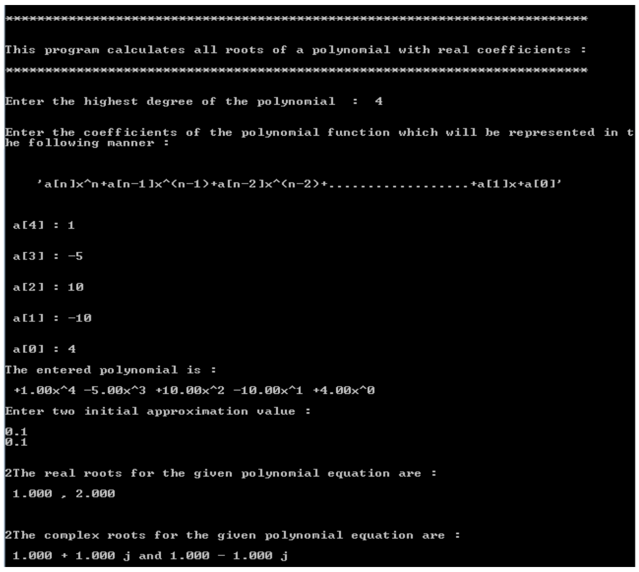
```
***************************************************************************

This program calculates all roots of a polynomial with real coefficients :

***************************************************************************

Enter the highest degree of the polynomial  :  1

Enter the coefficients of the polynomial function which will be represented in t
he following manner :


    'a[n]x^n+a[n-1]x^(n-1)+a[n-2]x^(n-2)+..................+a[1]x+a[0]'


 a[1] : 2

 a[0] : 3
The entered polynomial is :
 +2.00x^1 +3.00x^0
Enter two initial approximation value :
0.1
0.1

1The real roots for the given polynomial equation is :
 -1.500

_
```

Output for degree 2:



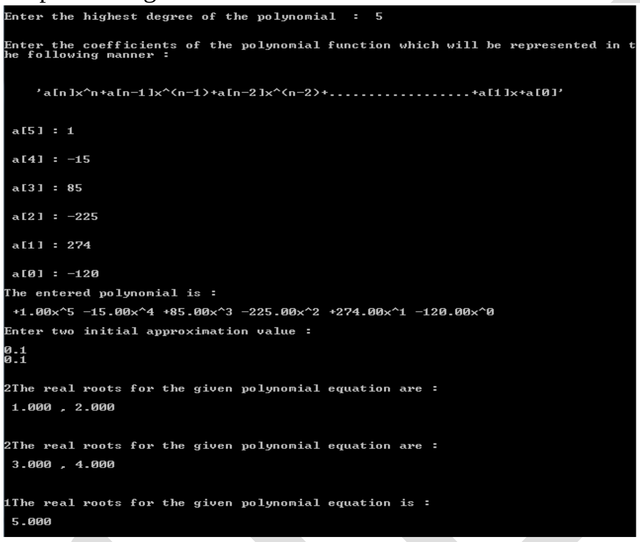
```
***************************************************************************

This program calculates all roots of a polynomial with real coefficients :

***************************************************************************

Enter the highest degree of the polynomial  :  2

Enter the coefficients of the polynomial function which will be represented in t
he following manner :


    'a[n]x^n+a[n-1]x^(n-1)+a[n-2]x^(n-2)+..................+a[1]x+a[0]'


 a[2] : 1

 a[1] : -3

 a[0] : 2
The entered polynomial is :
 +1.00x^2 -3.00x^1 +2.00x^0
Enter two initial approximation value :
0.1
0.1

2The real roots for the given polynomial equation are :
 1.000 , 2.000
```

Output for degree 4:



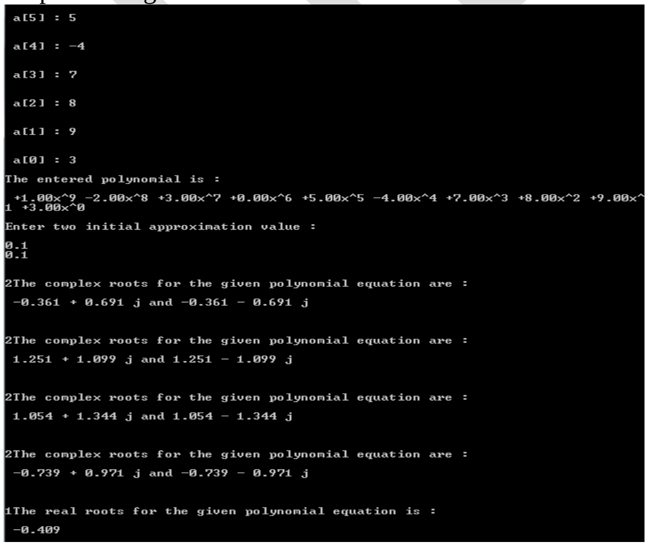
```
*************************************************************************

This program calculates all roots of a polynomial with real coefficients :
*************************************************************************

Enter the highest degree of the polynomial  :  4

Enter the coefficients of the polynomial function which will be represented in t
he following manner :


    'a[n]x^n+a[n-1]x^(n-1)+a[n-2]x^(n-2)+..................+a[1]x+a[0]'


 a[4] : 1

 a[3] : -5

 a[2] : 10

 a[1] : -10

 a[0] : 4
The entered polynomial is :
 +1.00x^4 -5.00x^3 +10.00x^2 -10.00x^1 +4.00x^0
Enter two initial approximation value :
0.1
0.1

2The real roots for the given polynomial equation are :
 1.000 , 2.000


2The complex roots for the given polynomial equation are :
 1.000 + 1.000 j and 1.000 - 1.000 j
```

Output for degree 5 :



```
Enter the highest degree of the polynomial  :  5

Enter the coefficients of the polynomial function which will be represented in t
he following manner :


    'a[n]x^n+a[n-1]x^(n-1)+a[n-2]x^(n-2)+..................+a[1]x+a[0]'


 a[5] : 1

 a[4] : -15

 a[3] : 85

 a[2] : -225

 a[1] : 274

 a[0] : -120
The entered polynomial is :
 +1.00x^5 -15.00x^4 +85.00x^3 -225.00x^2 +274.00x^1 -120.00x^0
Enter two initial approximation value :
0.1
0.1

2The real roots for the given polynomial equation are :
 1.000 , 2.000


2The real roots for the given polynomial equation are :
 3.000 , 4.000


1The real roots for the given polynomial equation is :
 5.000
```

Output for degree 9:



```
 a[5] : 5

 a[4] : -4

 a[3] : 7

 a[2] : 8

 a[1] : 9

 a[0] : 3
The entered polynomial is :
 +1.00x^9 -2.00x^8 +3.00x^7 +0.00x^6 +5.00x^5 -4.00x^4 +7.00x^3 +8.00x^2 +9.00x^
1 +3.00x^0
Enter two initial approximation value :
0.1
0.1

2The complex roots for the given polynomial equation are :
 -0.361 + 0.691 j and -0.361 - 0.691 j


2The complex roots for the given polynomial equation are :
 1.251 + 1.099 j and 1.251 - 1.099 j


2The complex roots for the given polynomial equation are :
 1.054 + 1.344 j and 1.054 - 1.344 j


2The complex roots for the given polynomial equation are :
 -0.739 + 0.971 j and -0.739 - 0.971 j


1The real roots for the given polynomial equation is :
 -0.409
```

Thus, it will give results for all powers of ‘n’.

CONCLUSION :

The above C code is capable of determining the roots of a polynomial of any degree. They have been tried for a polynomial with degree 1,2,3….8,9….. with high success and the results were very much comparable with already existing tools. The user will simply enter the coefficients of the polynomial in the indicated in the terminal, and the two initial values. The program, with few iterations, will then display the roots of the polynomial with the least possible error.
The procedure is easy and straight forward and can be used for other applications.

Acknowledgement:

This work was supported by Dr.Avijit Kar,Professor in the Department of Computer Science Engineering, Jadavpur University during my winter internship 2014-2015.

References

[1] Numerical Methods by Hilbert Schmidt.

[2] Karim Y. Kabalan, 2002, Technical Report on Root Computations of Real-coefficients Polynomials using Spreadsheets .

[3] NPTEL course on Numerical Analysis by Rathish Kumar.