Declaration

I do hereby declare that Ms. Shinjini Chattopadhyay has done her dissertation under my guidance and this project report entitled " Bairstow's Method" has been submitted for partial fulfilment of the dissertation in BSc. 6th semester for the year 2016 at the department of mathematics, St. Xavier's College ,Kolkata(Autonomous)

Professor Diptiman Saha
Department of Mathematics
St. Xavier's College, Kolkata

Acknowledgement

I take immense pleasure in thanking my supervisor Prof. Diptiman Saha, Department of Mathematics, St. Xavier's College, Kolkata for giving me a wonderful opportunity to carry out this project work.

Also I want to express my deep sense of gratitude to all the faculty members, Department of Mathematics, St. Xavier's College, Kolkata.

Finally, yet importantly I want to thank my parents for their support and love.

Last but not the least I am grateful to my friends for their help and co-operation that led to the successful completion of the project.

Shinjini Chattopadhyay,

MTMA- Roll 80,

Department of Mathematics,

St. Xavier's College, Kolkata

Abstract

We describe iterative methods for polynomial zero -finding and, specifically, the Bairstow's method and how it is used to find the roots of a polynomial equation for higher degrees.

We have also tried to describe it as an algorithm process and proceeded to implement it in a C-program

Also, we compare our results with Newton- Raphson's method.

Contents

Introduction

The problem of solving the polynomial equation

$$p(x) = p_0 + p_1 x + p_2 x^2 + \ldots + p_n x^n = 0 \qquad (0.1)$$

was known to the Sumerians (third millennium BC) and has greatly influenced the development of mathematics throughout the centuries. Starting with the Sumerian and Babylonian times, the study of polynomial zero finding focused on smaller degree equations for specific coefficients [Pan97]. The solution of specific quadratic equations by the Babylonians (about 200 BC) and the Egyptians (found in the Rhind or Ahmes papyrus of the second millennium BC) corresponds to the use of the quadratic formula:

$$x_{1,2} = \frac{-p_1 \pm \sqrt{p_1^2 - 4p_0p_2}}{2p_2}.$$

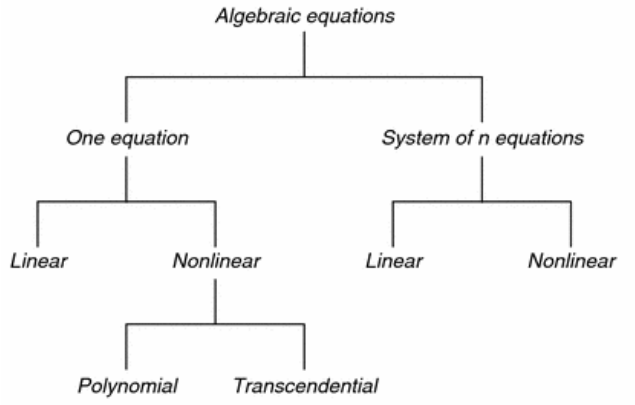
A full understanding of this solution formula, however, required the introduction of negative, irrational and complex numbers. Attempts to find solution formulae which would involve only arithmetic operations and radicals succeeded in the sixteenth century for polynomials of degree 3 and 4. However, for polynomials of degree greater than 4, it was not possible1 to find such formulae as was proved by Abel in 1827. The Galois theory was motivated by the same problem of solving (0.1) and included the proof of the nonexistence of the solution in the form of formulae. In spite of the absence of solution formulas in radicals, the fundamental theorem of algebra states that equation (0.1) always has a complex solution for any polynomial p(x) of any positive degree n. With no hope left for the exact solution formulae, the motivation came for designing iterative algorithms for the approximate solution and, consequently, for introducing several major techniques. Actually the list of iterative algorithms proposed for approximating the solution z1, z2, z3, . . . , zn of (0.1) includes hundreds of items and encompasses about four millennia.

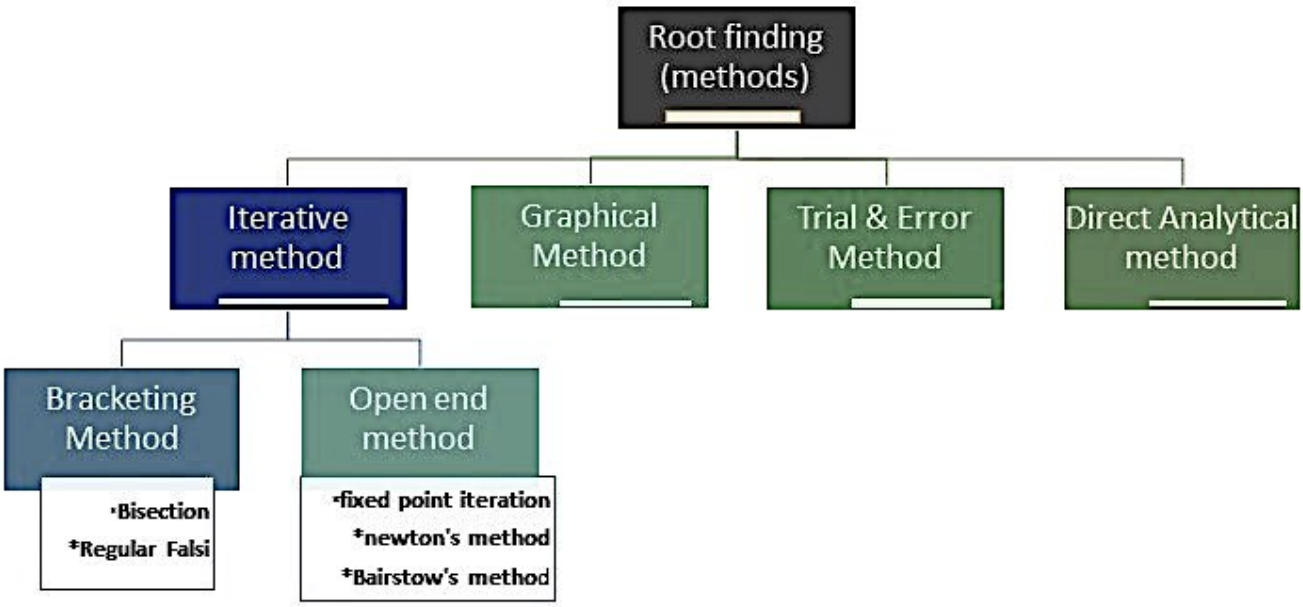
Root - Finding

Numerous scientific and technical problems can be described by means of single equations with one variable or systems of equations with n variables. According to the character of functions appearing in these equations , they can be linear or non linear. The corresponding classification of algebraic equations is given in the image below.

Definition Of Nonlinear Equations

Equation whose graph does not form a straight line (linear) is called a Nonlinear Equation. The variables are of degree greater than 1 or less than 1 , but never 1.





We have at our disposal various methods to find the roots of any non linear equation. Here, we will deal with polynomials in particular (which are

continuous functions). We are going to study Bairstow's method and also Newton's method

Certain Important Concepts

A numerical method for determining a zero of a polynomial generally takes the form of a prescription to construct one or several sequences zn of complex numbers supposed to converge to a zero of the polynomial. As one would expect, each algorithm has its advantages and disadvantages and therefore the choice of the 'best' algorithm for a given problem is never easy.

The desirable properties that an algorithm may or may not have include the following:

1. Global Convergence: Many algorithms can be guaranteed to converge only if the start- ing value z0 is sufficiently close to a zero of the polynomial. These are said to be locally convergent. Algorithms that do not require a sufficiently close starting value are globally convergent.

2. Unconditional Convergence: Some algorithms will only converge if the given polynomial has some special properties, e.g., all zeros simple or no equi-modular zeros. These algorithms are conditionally convergent. If an algorithm is convergent (locally or globally) for all polynomials, it is unconditionally convergent.

3. A posteriori Estimates: In practice, any algorithm must be artificially terminated after a finite number of steps. The approximation at this stage, zn, say, will not in general be identical to a zero ξ of the polynomial. Under such circumstances, it is desirable that we be able to calculate, from the data provided by the algorithm, a bound βn for the error $|zn - \xi|$ of the last approximation.

4. Speed of Convergence: The concept of order is frequently used as a measure of the ultimate speed of convergence of an algorithm. The order ν is defined as the supremum of all real numbers α such that

$$\limsup_{n\to\infty} \frac{|z_{n+1} - \xi|}{|z_n - \xi|^\alpha} < \infty.$$

Newton's method, for example, has the order of convergence $\nu = 2$, which means asymptotically that the number of correct decimal places is doubled at each step. Thus, the higher the order of convergence, the faster $|z - \xi|$ converges ultimately to zero.

5. Simultaneous Determination of All Zeros: Most algorithms determine one zero at a time. If the zero has been determined with sufficient accuracy, the polynomial is deflated and the algorithm is applied again on the deflated polynomial. It may be desirable for practical as well as theoretical reasons [Hen74] to determine all zeros simultaneously.

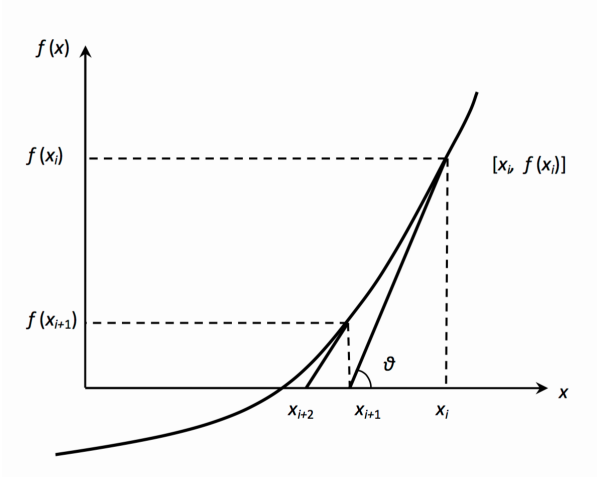
6. Cluster Insensitivity: A major problem in the numerical determination of zeros is presented by the occasional occurrence of the 'clusters' of zeros, i.e., sets of several zeros that either coincide or are close. The performance of many otherwise excellent methods is worsened in the presence of a cluster. We therefore want methods that are insensitive to clusters.

7. Numerical Stability: In real life, all computing is done in the finite system of discrete and bounded numbers of a machine, instead of the field of real or complex numbers. The set of numbers provided by floating point arithmetic is finite. Thus, algorithms originally devised to work in the continuum are adapted to 'machine numbers' by the devices of rounding and scaling. Not all algorithms are equally insensitive to this adaptation. By numerical stability, we mean the lack of sensitivity to rounding and scaling operations or more precisely, the sensitivity of the algorithm should be no greater than that inherent in the ill-conditioning of the zero finding problem.

Newton's Method

Before, we study Bairstow's method, we need to understand Newton's method.

Newton Raphson method, also called the Newton's method, is the fastest and simplest approach of all methods to find the real root of a nonlinear function. It is an open bracket approach, requiring only one initial guess. This method is quite often used to improve the results obtained from other iterative approaches.



Newton's method, or sometimes the Newton - Raphson method, is an iterative method which is intended to have second-order convergence to the solution of $f(x) = 0$ is,

The iteration formulae is,

$$x_{n+1} = x_n - f(x_n)/f'(x_n) \qquad \text{(1.1)}$$

Let us start by presenting one of the ways that we can actually obtain such a numerical root finding scheme.

Derivation of Newton-Raphson method from Taylor series:

Newton-Raphson method can also be derived from Taylor series. For a general function f (x), the Taylor series is:

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 + \cdots$$

As an approximation, taking only the first two terms of the right hand side,

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

and we are seeking a point where f (x)= 0, that is, if we assume

$$f(x_{i+1}) = 0,$$

$$0 \approx f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

which gives,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Newton's method convergence theorem:

Suppose f is twice differentiable on an interval [a, b] and that

(i) f(a) f(b) < 0,

(ii) f ' has no zeros in [a, b],

(iii) f'' does not change sign in [a,b]

(iv) | f(a)/f'(a) |, | f(b)/f'(b) | <b-a

then, for any $x_0 \in$ [a,b], Newton's iteration converges to the unique solution of (1.1) in [a,b].Condition (iv) in this theorem is used to guarantee that the iterates remain in the interval (a,b).

System Of Equations

For a system of equations

$$F(x) = 0 \text{ ——————-(3)}$$

where F: IRn —> IRn is a differentiable function Newton's method can be derived from the

first-order Taylor approximation:

$$\boldsymbol{F(\mathbf{x} + \mathbf{h}) \approx F(\mathbf{x}) + J\mathbf{h}}$$

where J is the Jacobian matrix of F at x: $J_{ij} = \frac{\partial F_i}{\partial x_j}$

evaluated at x. We therefore get the Newton iteration:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - J^{-1}F(\mathbf{x}_k)$$

In practice, the inverse matrix would not be generated (certainly for $n > 2$) but the correction would be computed by solving the appropriate system of equations. Thus the iteration becomes

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{s}_k$$

where s_k is the solution of $J_s = F(x_k)$

BAIRSTOW'S METHOD

We know that dividing a polynomial by its root result in the remainder of zero. If the divisor is not a root then the reminder will not be zero.

The basic approach of Bairstow's method is starting on guess value for root and dividing the polynomial by this value. Then the estimated root is changed until the reminder of the division get sufficiently small.

Complex Roots and Quadratic Factors:-

In many situations, the roots we seek for a polynomial equation may be complex or perhaps real repeated roots or real roots of the same magnitude but opposite signs. Such roots, and others whose separation is not great, may be best found by identifying the appropriate quadratic factor of the original polynomial and then solving the quadratic for the actual roots.

Fundamentals of Bairstow's Method

This method is only valid for polynomials with real coefficients. For such polynomials, we know that any complex roots occur as conjugate pairs.

Bairstow's **method** is an iterative technique for finding a quadratic factor of a polynomial. Thus we seek a quadratic $x^2 - ux - v$ —-2.1

which is a factor of the polynomial

$$p(x) = a_N x^N + a_{N-1} x^{N-1} + a_{N-2} x^{N-2} + \cdots + a_1 x + a_0 \quad \text{—-2.2}$$

Using the remainder theorem, we can write

$$p(x) = (x^2 - ux - v)\, q(x) + cx + d$$

where q is a polynomial of degree $N - 2$ and $(x^2 - ux - v)$ is a factor of p if $c = d = 0$. For convenience in the algorithm, we write the remainder in the form $b_1(x - u) + b_0$ and the polynomial q as $q(x) = b_N x^{N-2} + b_{N-1} x^{N-3} + \cdots + b_3 x + b_2$. We thus have

$$p(x) = a_N x^N + a_{N-1}x^{N-1} + a_{N-2}x^{N-2} + \cdots + a_1 x + a_0 \qquad \text{—2.3}$$
$$= (x^2 - ux - v)(b_N x^{N-2} + b_{N-1}x^{N-3} + \cdots + b_3 x + b_2) + b_1(x - u) + b_0$$

and seek the coefficients u, v such that $b_1 = b_0 = 0$.

Setting $b_{N+2} = b_{N+1} = 0$ and comparing coefficients in (5.2.3) we get the relations

$$b_k = a_k + ub_{k+1} + vb_{k+2} \qquad (k = N, N-1, \ldots, 1, 0) \qquad \text{——2.4}$$

from which we see immediately that the coefficients bk are all functions of u, v. The task is thus to solve the equations

$$b_1(u, v) = b_0(u, v) = 0 \qquad \text{———————-2.5}$$

Application Of Newton's Method

Bairstow's method is the application of Newton's method to the solution of (2.5). In order to apply Newton's method we require the Jacobian matrix for this system. The elements of this matrix are derived using a similar recursion to (2.4). With

After obtaining a quadratic factor, the polynomial p is deflated and the same procedure is applied to the deflated polynomial.
This method, when it converges, is quadratically convergent. However, such convergence requires a very good initial approximation though the method uses only real arithmetic to compute even complex roots.

ALGORITHM

Input Coefficients $a_0, a_1, \ldots, a_N$ of the polynomial;
Initial guess u_1, v_1 for coefficients of a quadratic factor $(x^2 - ux - v)$

Initialize $b_{N+1} = b_{N+2} = c_{N+1} = c_{N+2} = 0$

Compute repeat

$u_0 := u_1;\ v_0 := v_1$

for $k := N$ down to 0 $\quad b_k := a_k + u_0 b_{k+1} + v_0 b_{k+2}$

for $k := N$ down to 1 $\quad c_k := b_k + u_0 c_{k+1} + v_0 c_{k+2}$

$D := c_2^2 - c_1 c_3$

$u_1 := u_0 - (c_2 b_1 - c_3 b_0)/D$

$v_1 := v_0 - (c_2 b_0 - c_1 b_1)/D$

until convergence

Output Quadratic factor $(x^2 - u_1 x - v_1)$

Analysis

Polynomial Deflation

Thus far, we have techniques for locating the largest and smallest roots

and for finding quadratic factors of polynomials. In the event that we know the vicinity of other roots, we can of course use general equation solving routines such as Newton's method to locate them accurately.

In the absence of such additional information, other techniques must be adopted. One possibility is to locate the dominant root of the polynomial and then to factor this root out of the polynomial before locating the dominant root of the resulting quotient polynomial. This process is known as **polynomial deflation.**

Of course any root could be used to deflate the polynomial and so reduce the degree for the remaining problem.

The Problem of Deflation

Suppose that we have identified an approximate root u of the polynomial $p(x)$ given by

$$p(x) = p_0 + p_1 x + p_2 x^2 + \ldots + p_n x^n = 0$$

Following a similar procedure to that used in Bairstow's method, we first use the remainder theorem to write

$$p(x) = (x - u)\, q(x) + b_0$$

where q is a polynomial of degree $n-1$. Of course if u is an exact root then $b_0 = 0$ and further roots of p can be obtained by finding roots of q.

One difficulty with this approach is that because the factors are approximate there is a build-up of round-off error which results in subsequent roots being less accurate than may be desired. It is always good practice therefore to use roots of the deflated polynomial as initial points for Newton's method applied to the original polynomial to reduce this effect.

Note that deflation by a quadratic factor is achieved by taking the polynomial q obtained in Bairstow's method.

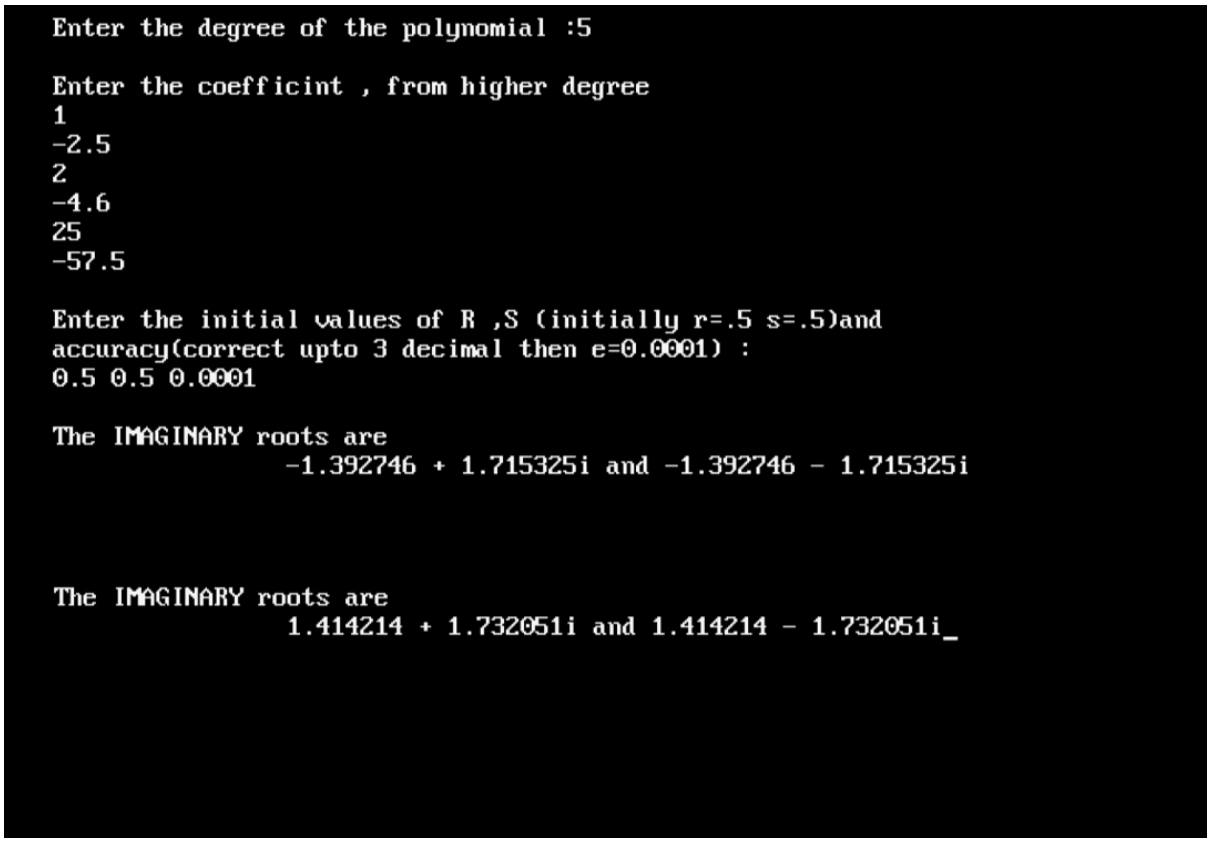
C-Programs

//Bairstow Method for solution of Polynomial

```
#include<math.h>
#include<stdio.h>
#include<conio.h>
void main()
{       int n,i,it=0;
        float a[10],b[10],c[10];
        float r,s,r1,s1,e,dis,rt;
        printf("\nEnter the degree of the polynomial :");
        scanf("%d",&n);
        printf("\nEnter the coefficint , from higher degree",i);
        for(i=n;i>=0;i--)
                scanf("%f",&a[i]);
        printf("\nEnter the initial values of R ,S (initially r=.5 s=.5)and
\naccuracy(correct upto 3 decimal then e=0.0001) : ");
        scanf("%f%f%f",&r,&s,&e);
        do{     it++;
                b[n]=a[n];
                b[n-1]=a[n-1]+r*b[n];
                c[n]=b[n];
                c[n-1]=b[n-1]+r*c[n];
                for(i=n-2;i>=0;i--)
                {       b[i]=a[i]+r*b[i+1]+s*b[i+2];
                        c[i]=b[i]+r*c[i+1]+s*c[i+2];
                }
                r1=(b[0]*c[3]-b[1]*c[2])/(c[2]*c[2]-c[1]*c[3]);
                s1=(b[1]*c[1]-b[0]*c[2])/(c[2]*c[2]-c[1]*c[3]);
                r=r+r1;
                s=s+s1;
        }while((fabs(r1)>e)||(fabs(s1)>e));
        dis=r*r+4*s;
        rt=(sqrt(fabs(dis)))/2.0;
        if(dis>=0)
                printf("\nThe REAL roots are \n\t\t%f  %f",r/2.0+rt,r/2.0-rt);
        else
                printf("\nThe IMAGINARY roots are \n\t\t%f + %fi and %f -
%fi",r/2.0,rt,r/2.0,rt);
        getch();
```

}

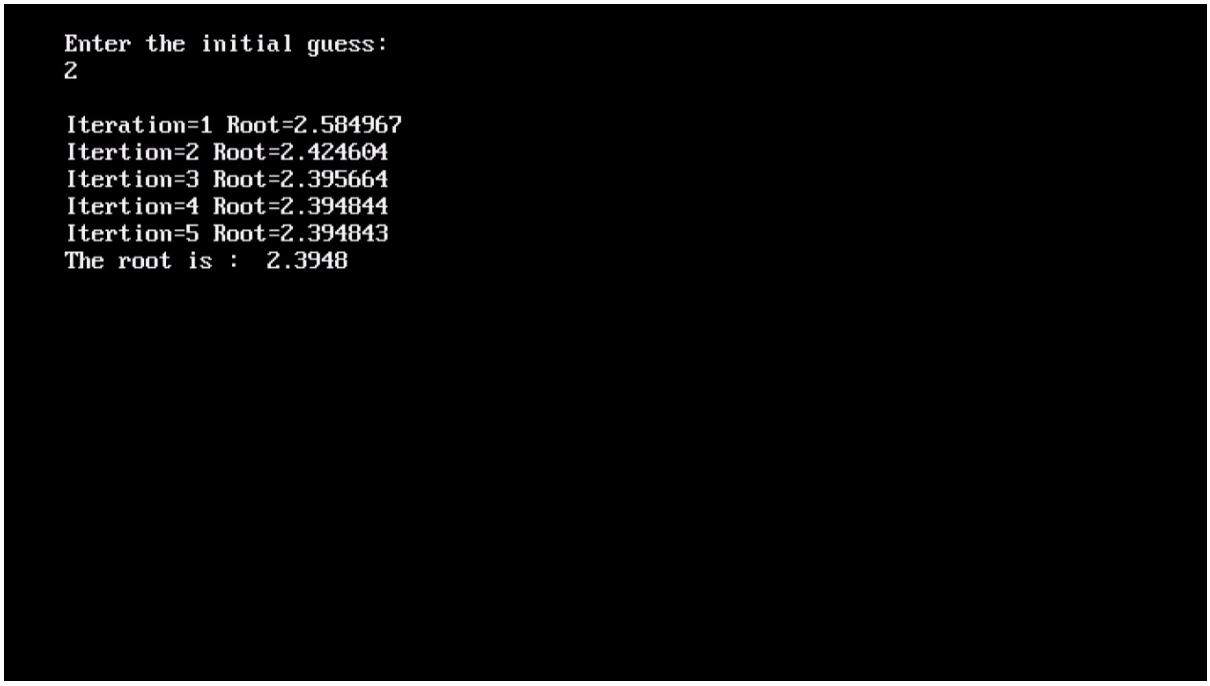
OUTPUT:


Enter the degree of the polynomial :5
Enter the coefficint , from higher degree
1
-2.5
2
-4.6
25
-57.5
Enter the initial values of R ,S (initially r=.5 s=.5)and
accuracy(correct upto 3 decimal then e=0.0001) :
0.5 0.5 0.0001
The IMAGINARY roots are
-1.392746 + 1.715325i and -1.392746 - 1.715325i
The IMAGINARY roots are
1.414214 + 1.732051i and 1.414214 - 1.732051i_

Newton Raphson Method

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float f(float);
float df(float);
void main()
{
	float x[100];
	int i;
	clrscr();
	printf("\nEnter the initial guess:\n");
	scanf("%f",&x[0]);
	x[1]=x[0]-(f(x[0])/df(x[0]));
	printf("\nIteration=1 Root=%f",x[1]);
	i=0;
	while(fabs(x[i+1]-x[i])>0.00001)
	{
		i++;
		x[i+1]=x[i]-(f(x[i])/df(x[i]));
		printf("\nItertion=%d Root=%f",i+1,x[i+1]);
	}
	printf("\nThe root is : %7.4f",x[i+1]);
	getch();
}
float f(float x)
{
	return((x*x*x*x*x)-(2.5*x*x*x*x)+(2*x*x*x)-(4.6*x*x)+(25*x)-57.5);
}
float df(float x)
{
	return((5*x*x*x*x)-(4*2.5*x*x*x)+(6*x*x)-(9.2*x)+25);
}
```

OUTPUT:


Enter the initial guess:
2
Iteration=1 Root=2.584967
Itertion=2 Root=2.424604
Itertion=3 Root=2.395664
Itertion=4 Root=2.394844
Itertion=5 Root=2.394843
The root is : 2.3948

Conclusion

Newton- Raphson

Advantages

*This method is the best fastest convergences to the root. This feature makes the newton raphson method to stand upfront from the other known methods.

*Apart from the fast convergences, it also converges quadraticly on the root. This advantages show that this method also deals with the higher degree of variable involved.

*Next third advantage that in this method, the number of significant digits doubles with each step approximately (near a root).

*This method is that it is flexible; it means that it is easier to convert this method to multiple dimensions.

Drawbacks

1. **Divergence at inflection points**

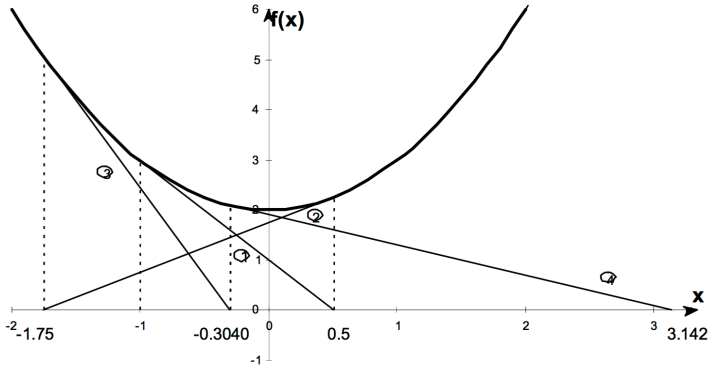
If the selection of the initial guess or an iterated value of the root turns out to be close to the inflection point (see the definition in the appendix of this chapter) of the function
f(x)in the equation f(x)=0,Newton-Raphson method may start diverging away from the root. It may then start converging back to the root. For example, to find the root of the equation

2. **Division by zero For the equation**

If f'(x) at any point becomes zero , then the method fails to converge and hence mayn't produce a root.

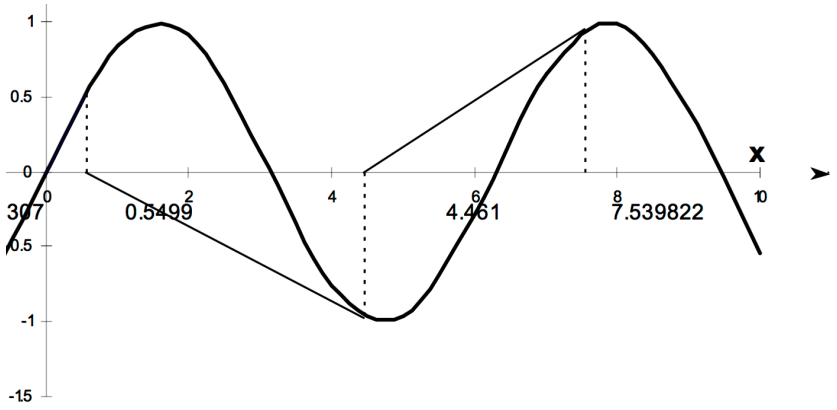
3. **Oscillations near local maximum and minimum**

Results obtained from the Newton-Raphson method may oscillate about the local maximum or minimum without converging on a root but converging on the local maximum or minimum. Eventually, it may lead to division by a number close to zero and may diverge.



4. **Root jumping**

In some case where the function f (x) is oscillating and has a number of roots, one may
choose an initial guess close to a root. However, the guesses may jump and converge to some other root.



Root jumping from intended location of root for $f(x) = \sin x = 0$.

Bairstow's Method

Advantages

Only real arithmetic is needed during computation; this can give a good increase in the performance and much easier handling of overflow and underflow conditions;

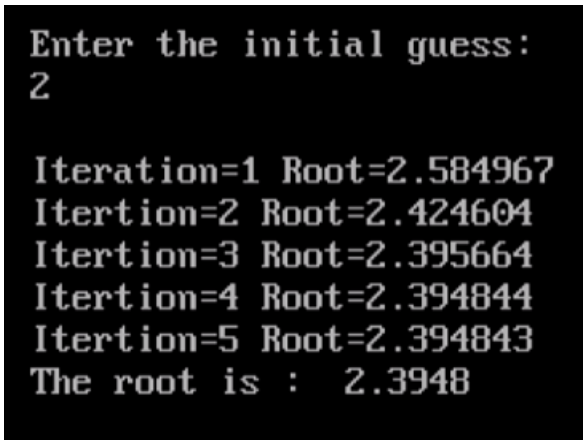
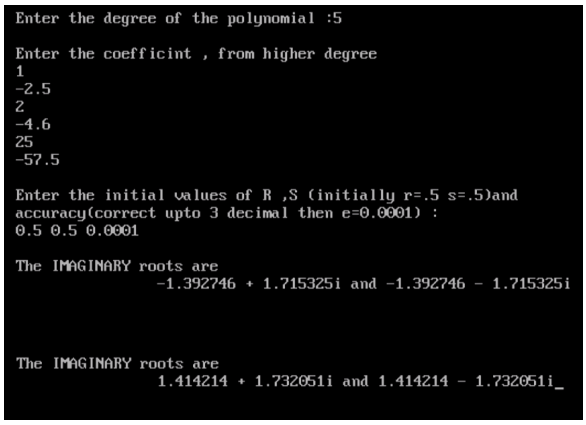
The structure of the problem is used and maintained, so the roots computed by the algorithm are guaranteed to be real or complex conjugated pairs;

Disadvantages

The main drawback is that it sometimes fail to converge.

This is because it is hard to guess an initial starting point which satisfies the necessary and sufficient condition necessary to assure convergence. When these conditions are not satisfied the sequence approximations may jump away from the desired root or may iterate away from the roots indefinitely

It is limited to real polynomials and therefore is not so convenient in practice as one is usually interested in polynomials with complex coefficients as well. Unless modified, the method is quite slow to converge to quadratic factors of multiplicity greater than 1.

Newton Raphson Method	Bairstow's Method
Newton's method is used to find the real roots of a polynomial function	**Bairstow's method is used to find both real and complex roots of the polynomial**
If the root being sought has multiplicity greater than one, the convergence rate is very slow	**This method, when it converges, is quadratically convergent. However, such convergence requires a very good initial approximation though the method uses only real arithmetic to compute even complex roots.**
The more closely a tangent line matches its parent function, the better a job it will do of approximating the roots and y-values of its function.	**Bairstow involves less work, but the result of two applications will leave a deflated polynomial whose error bounds are large**
OUTPUT	**OUTPUT**
	
Hence, it is not feasible for complex roots. we can only find the real roots of the polynomial with real coefficients	**The results appear in conjugates. (either complex or real)**

Reference

1. Numerical Analysis Peter R. Turner

2. Bairstow's Algorithm on Mathworld

3. Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis. Berlin, Heidelberg, New York:

Springer 1980

4. Schroder, J.: Factorization of polynomials by generalized Newton procedures. In: Constructive

aspects of the fundamental theorem of algebra (Dejon, B., Henrici, P., eds.), pp. 295-320. Proc.

Symp. Zurich-Ruschlikon, London: John Wiley 1967

5. http://ocw.mit.edu/courses/mathematics/18-330-introduction-to-numerical-analysis-spring-2004/