**North Western University, Khulna**

**Course Title :Numerical Analysis**

**Course Code :CSE-2303**

*Sajib Chatterjee*

*Lecturer*

*North Western University*

*Khulna*

*Name : Shimul Majumdar*

*ID : 20191119010*

*Section : D*

*Year : 2nd*

*Semester:3rd*

*Program : B.Sc in CSE*

***BISECTION METHOD:***

**Bisection Method background:**

**Bisection** is the division of something into two equal or [congruent](https://en.wikipedia.org/wiki/Congruence_(geometry)) parts, usually by a [line](https://en.wikipedia.org/wiki/Line_(mathematics)), which is then called a *bisector*. The most often considered types of bisectors are the *segment bisector* (a line that passes through the midpoint of a given [segment](https://en.wikipedia.org/wiki/Line_segment)) and the *angle bisector* (a line that passes through the apex of an [angle](https://en.wikipedia.org/wiki/Angle), that divides it into two equal angles).The bisection method is one of the simplest and most reliable of iterative methods for the solution of nonlinear equations. This method, also known as binary chopping or half-interval method, relies on the fact that if f(x) is real and continuous in the interval a < x < b, and f(a) and f(b) are of opposite signs, that is,

F(a) f(b) < 0

Then there is at least one real root in the interval between a and b. (There may be more than one root in the interval).

Let x = a and x2 = b. Let us also define another point x0 r, to be the midpoint between a and b. That is,

x0 =

Now, there exists the following three conditions:

1. f (x0) = 0, we have a root at x0
2. 2. If f(x0) f(x1)<0, there is a root between x0 and x1 .
3. 3. If f(X0) f(x2 )< 0, there is a root between x0, and x2

It follows that by testing the sign of the function at midpoint, we can deduce which part of the interval contains the root. This is illustrated in shows that, since f(x0) and f(x2) are of opposite sign, a root between x0, and x2 .We can further divide this subinterval into two ha to locate a new subinterval containing the root. This process can be repeated until the interval containing the root is as small as we desire.

**Bisection algorithm:**

1. Decide initial values for x1, and x2, and stopping criterion, E.
2. Compute f1 = f(x1) and f2= f(x2).
3. If f1 \* f2 >0, x1  and x2  do not bracket any root and go to step 7;

Otherwise continue.

4. Compute x0 = (x1 + X2)/2 and compute f0= f(x0)

5. If f1 x f2<0 then

set x2 = x0

else

set x1 = x0

set f1 = f0

6. If absolute value of (x2 – x1)/x2 is less than error E, then

root = (x1 + x2)/2

write the value of root

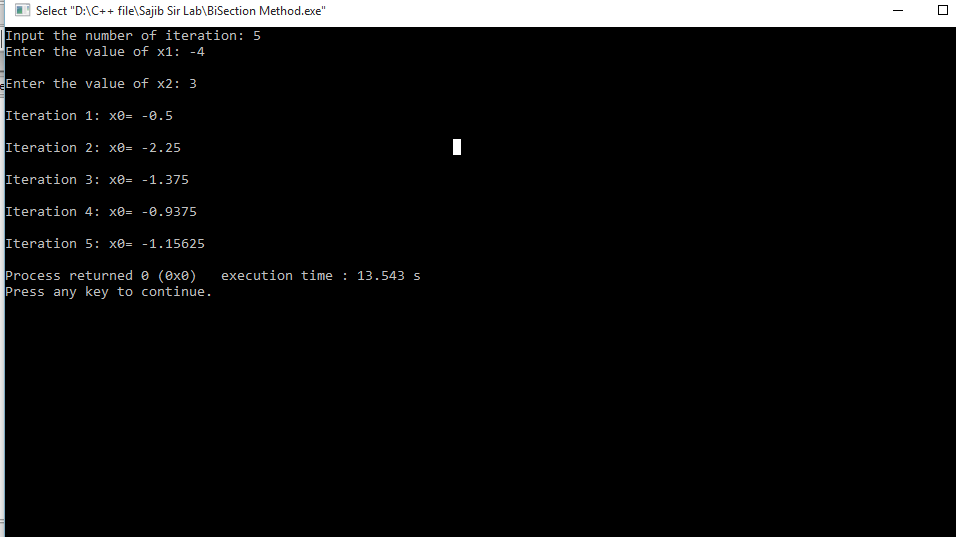
go to step 7

else

go to step 4

7. Stop

**Bisection output:**

****

***FALSE POSITION METHOD:***

**False Position Method Background:**

An algorithm for finding roots which retains that prior estimate for which the function value has opposite sign from the function value at the current best estimate of the root. In this way, the **method** of **false position** keeps the root bracketed.Bisection method, the interval between x1, and x2, is divided into two equal halves, irrespective of location of the root. It may be possible that the root is closer to one end. Note that the root is closer to x1. Let us join the points x1, and x2, by a straight line. The point of intersection of this line with the x0 axis (x0) gives an improved estimate of the root and is called the false position of the root. This point then replaces one of the initial guesses that has a function value of the same sign as f(x0). The process is repeated with the new values of x1, and x2. Since this method uses the false position of the root repeatedly, it is called the false position method. It is also called the linear interpolation method (because approximate root is determined by linear interpolation).

A graphical depiction of the false position method is show the picture .we know that equation of the line joining the points ( x1, f(x1) )and ( x2, f(x2) is given by

Since the line intersects the x-axis at x0 when x=x0 ,y=0,we have

Or,

Then we have,

This equation is known as the false position formula. Note that is obtained by applying a correction to .

**False Position Method Algorithm:**

Let \*

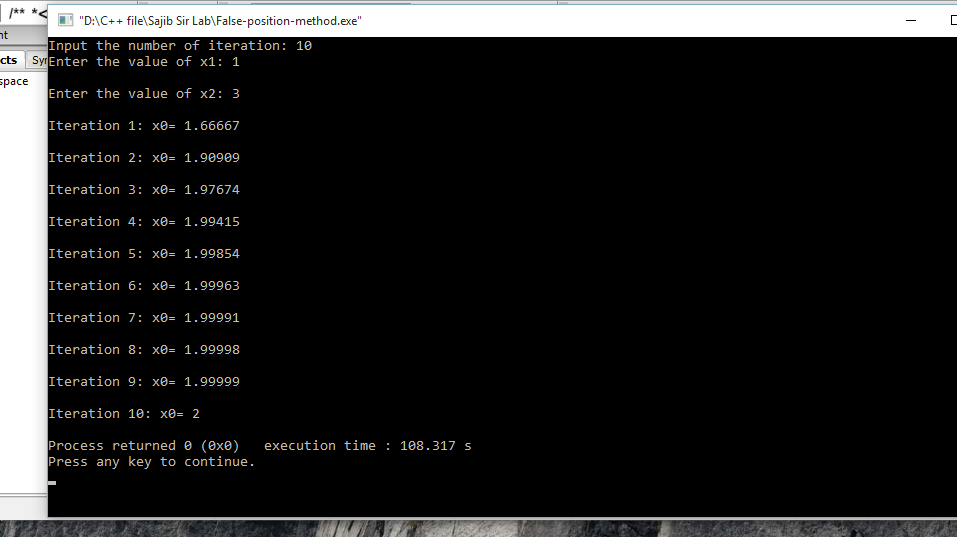
If <0

Set x2 =x0

Otherwise

Set x1=x0

**False position output :**



***SECANT METHOD:***

**Secant Method Background:**

In numerical analysis, the **secant method** is a root-finding **algorithm** that uses a succession of roots of **secant** lines to better approximate a root of a function f. The **secant method** can be thought of as a finite-difference approximation of Newton's **method**.Secant method, like the false position and bisection methods, uses two initial estimates but does not require that they must bracket the root. For example, the secant method can use the points x1 and x2 as starting values, although they do not bracket the root. Slope of the secant line passing through x1, and x2 is given by

Or,

Then,

By adding and subtracting to the numerator and rearranging the terms we get

This Equation is known as the secant formula. If the secant line represents the linear interpolation polynomial of the function f(x) (with the interpolating points x1, and x2) then x3, which intercepts the x-axis, represents the approximate root of f(x).

The approximate value of the root can be refined by repeating this procedure by replacing x, and x2 by x2 and x3, respectively

That is, next approximate value is given by

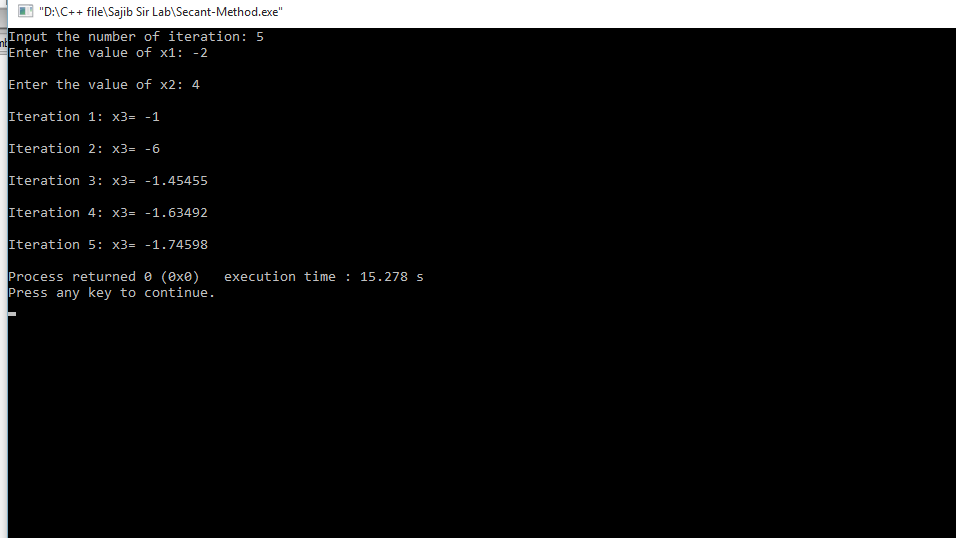
This procedure is continued till the desired level of accuracy is obtained. We can express the secant formula in general form as follows:

Note that are similar and both of them use two initial estimates. However, there is a major difference in their algorithms of implementation. The latest estimate replaces one of the end points of the interval such that the new interval brackets the root. But, the values are prefaced in strict sequence, i.e., is replaced by xi, and xi; by. The points may not bracket the root.

**Secant Method Algorithm:**

1. Start
2. Get values of x0, x1 and e  
   \*Here x0 and x1 are the two initial guesses  
   e is the stopping criteria, absolute error or the desired degree of accuracy\*
3. Compute f(x0) and f(x1)
4. Compute x2 = [x0\*f(x1) – x1\*f(x0)] / [f(x1) – f(x0)]
5. Test for accuracy of x2  
   If [ (x2 – x1)/x2 ] > e, \*Here [ ] is used as modulus sign\*  
   then assign x0 = x1 and x1 = x2  
   goto step 4  
   Else,  
   goto step 6
6. Display the required root as x2.
7. Stop

**Secant Method Output:**



***JACOBI ITERATION METHOD:***

**Jacobi Iteration Background:**

In numerical linear algebra, the **Jacobi method** is an **iterative algorithm** for determining the solutions of a strictly diagonally dominant system of linear equations. Each diagonal element is solved for, and an approximate value is plugged in. The process is then iterated until it converges. The Gauss–Seidel method is an [iterative technique](https://en.wikipedia.org/wiki/Iterative_method) for solving a square system of *n* linear equations with unknown **x**:

{\displaystyle A\mathbf {x} =\mathbf {b} }.

It is defined by the iteration {\displaystyle L\_{\*}\mathbf {x} ^{(k+1)}=\mathbf {b} -U\mathbf {x} ^{(k)},} where {\displaystyle \mathbf {x} ^{(k)}} is the *k*th approximation or iteration of {\displaystyle \mathbf {x} ,\,\mathbf {x} ^{(k+1)}}is the next or *k* + 1 iteration of {\displaystyle \mathbf {x} }and the matrix *A* is decomposed into a [lower triangular](https://en.wikipedia.org/wiki/Triangular_matrix) component {\displaystyle L\_{\*}}, and a [strictly upper triangular](https://en.wikipedia.org/wiki/Triangular_matrix#Strictly_triangular_matrix) component *U*: {\displaystyle A=L\_{\*}+U}

In more detail, write out *A*, **x** and **b** in their components:

xi+1 = g(xi) for i = 0,1,2……

Jacobi method extends this idea to a system of equations. It is a direct substitution method where the values of unknowns are improved by substituting directly the previous values.  
Let us consider a system of n equations in n unknowns .

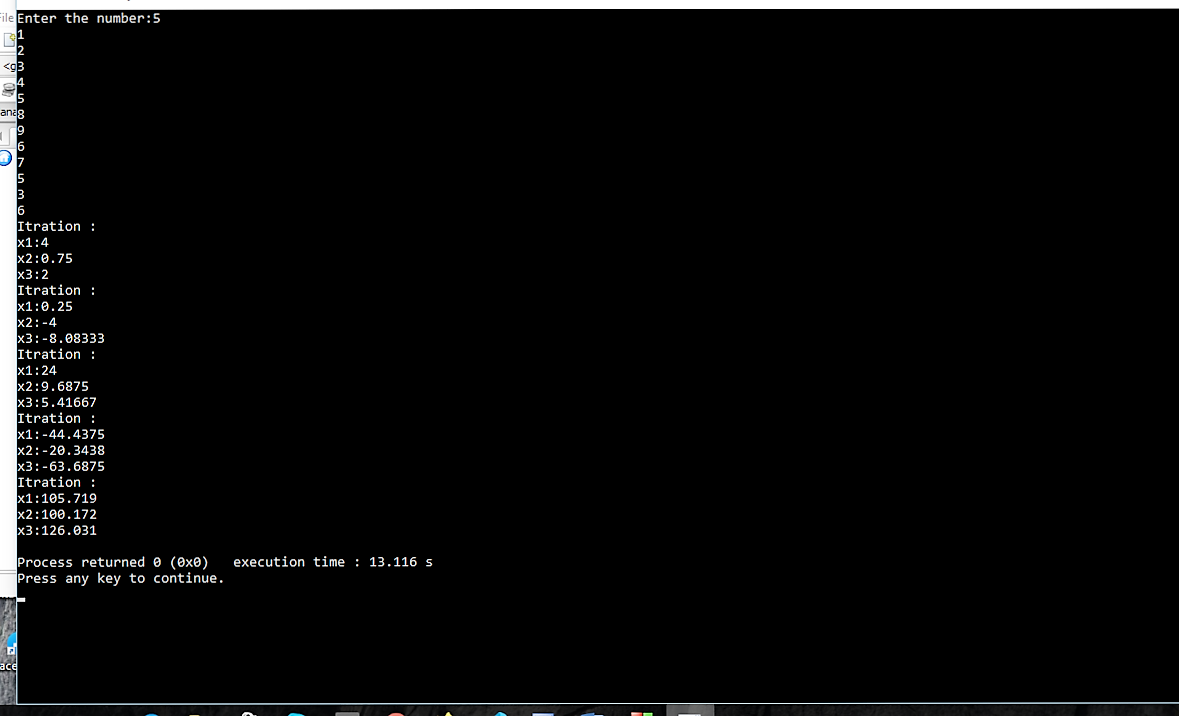
**Jacobi Iteration Algorithm:**

1. Obtain n, aij, and bi, values.  
 2. Set X0i = bi / aij, for i = 1 ... n   
 3. Set key = 0   
 4. For i = 1,2, ... n   
 (i)Set sum = b,   
 (ii) For j = 1,2….n (j =! i)   
 Set sum = Sum- aijx0j  Repeat j  
 (iii) Set xi = sum / aii   
 (iv) if key = 0 then   
 if > error then   
 set key = 1  
 Repeat i   
 5. If key = 1 then

set Xoi= Xi

go to step 3   
 6. Write results

**Jacobi Iteration Output:**

****

***GAUSS-SEIDEL METHOD:***

**Gauss–Seidel Background:**

In numerical linear algebra, the **Gauss**–**Seidel** method, also known as the Liebmann method or the method of successive displacement, is an iterative method used to solve a system of linear equations.

Gauss seidel method is an improved version of Jacobi iteration method. In Jacobi method we begin with the initial values

X1(0), X2(2),……… Xn(0)

And obtain next approximation

X1(1), X2(1),……… Xn(1)

Note that, in computing X2(1), we used X1(0) and not X1(1) which has Just

been computed. Since, at this point, both X1(0) and X1(1) are available, we can use X1(1) which is a better approximation for computing X2(1) Similarly, for computing X3(1), we can use X1(1) and X2(1) along with X4(0) …… Xn(0)

This idea can be extended to all subsequent computations. This approach is called the Gauss-Seidel method.

The Gauss-Seidel method uses the most recent values of x as soon as they become available at any point of iteration process. During the (k+1)th iteration of Gauss-Seidel method, xi takes the form

Xi(k+1)=

When i = 1, all superscripts in the right-hand side become (k) only.

Similarly, when i = n, all become (K + 1). Illustrates pictorially

the difference between the Jacobi and Gauss-Seidel method.

**Gauss–Seidel Algorithm:**

1. Obtain n, aij and bi, values

2. Set xi = b/aii for i=1 to n

3. Set key = 0

4. For i= 1 to n

(i) Set sum = bi

(ii) For j = 1 to n ( ji )

Set sum = sum – aij xj

Repeat j

(iii) Set dummy = sum / ajj

(iv) If key = 0 then

If >error then

Set key = 1

(v) set xi =dummy

Repeat i

1. If key = 1 then

Go to step 3

6. Write results

**Gauss–Seidel Output:**

