# ASSIGNMENT-1

Date: 01/12/14

* **Problem Statement:**

Write a program in C to find the root of the equation x3-9x+1 using bisection method.

* **Algorithm:**

Bisection Method :

steps:

begin

1. start

[ a and b are two starting values which enclosed the desired root, Steps 2, 3 and 4 are called initialization steps which prepare the ground for the succeeding steps in the algorithm]

1. compute a := f(a)
2. compute b := f(b)
3. [Initialization]
4. set i :=0
5. set x1 := 0
6. set x2 := 0
7. set delta :=1
8. if ((a\*b)<0) then
9. while (|delta| > epsilon) do

begin

1. set x1 := x2
2. compute x2 := (a+b)/2
3. compute k := f(x2)
4. set i := i+1
5. if (k<0) then [ If k < 0 then then the value of x2 stored in a ]

a := x2

1. if(k>0) then [ If k > 0 then then the value of x2 stored in b ]

b:= x2

1. delta=|x2-x1|

[end of while]

1. print ‘the root is =’,x2
2. print ‘Number of iterations = ‘,i
3. else

print ‘ No root exists between two initial value a and b ’

[end of if]

1. stop

* **Source Code:**
* **Discussion:**
* The method is guaranteed to converge to a root of f, if f is a continuous function on the interval [a, b] and f(a) and f(b) have opposite sings. The absolute error is the halved to each step so the method converges linearly, which is comparatively slow.
* Specifically, if p1=(a+b)/2 is the midpoint of the initial interval,andpn is the midpoint of the interval in the nth step, then the difference between pn and a solution p is bounded by |pn-p|<=|b-a|/2n.
* This formula can be used to determine in advance the number of iterations that the bisection method would need to converge to a root to within a certain tolerance

# ASSIGNMENT-2

Date: 16/12/14

* **Problem Statement:**

Write a program in C to compute the root of an equation using Newton Raphson Method. x3 -8x-4=0 (correct up to 4 decimal places).

* **Algorithm:**

Newton Raphson(X0, epsilon)

[X0 is the initial guess of the root & epsilon is the relative error. This algorithm compute the root for an equation f(X)=0 using algorithm Newton Raphson method. The relative margin b/w two successive values should be less than epsilon where the relative margin is taken as magnitude.]

begin

steps:

1. start
2. [Initializing]

set i:=0

1. compute f0:=f(X0) [Where f(X) is the given equation]
2. compute f’0:=f’(X0) [Where f’(X) is the derivate function of f(X)]
3. compute X1 := X0-(f0/f’0)
4. set i := i+1
5. if (|X1-X0|) ≥ epsilon then
   1. set X0:=X1
   2. print ith pass approximated is X1
   3. goto step 2

[end of if]

1. print the root calculated in X1
2. stop

* **Source Code:**
* **Discussion:**
* We should terminate the iteration when |xn+1-xn|<=tolerable error
* This method is sometimes known as NEWTON`s method of tangent.When f’(x)=0 or very small in the equation f(x)=0,then Newton Raphson method is fails evidently.When f’(x) is a simple expression and easily found then this method is used.
* If the initial approximation is very close to the root, then the convergence in Newton Raphson method is faster than iteration method.

# ASSIGNMENT-3

Date: 19/1/15

* **Problem Statement:**

Write a program in C to compute the unknowns for a set of equations using Gaussian Elimination Method.

* **Algorithm:**

Gaussian Elimination Method

begin

steps :

1. Input the no. of variables. Read n
2. Input adjacency matrix. Read matrix[i][j] [size of n×n where n is the number of row and column ]
3. for i=0,1,2,….,n do
   1. for j=0,1,2,….,n do
      1. if j>I then,
      2. Set a=matrix[j][i]
      3. Set b=matrix[i][i]
      4. For k=0,1,2,….,n+1 do
         1. Set matrix[j][k] -= (a/b) \* matrix[i][k]
         2. end for
         3. end if
      5. end for
   2. end for
4. Print the triangular matrix, matrix[i][j]
5. for i=n-1,…,0 do
   1. Set b = matrix[i][n]
   2. for j=n-1,…,i do
      1. Set b -= temp[n-j]\*matrix[i][j]
      2. end for
   3. Set temp[n-i] = b/matrix[i][i]
   4. Print the roots of the equation, temp[n-i]
   5. end for
6. Stop

* **Discussion:**
* It is a direct method for finding the solution or the values of unknown of a system of linear equations and is based on principle of elimination of unknowns in successive steps.
* The non-zero constants are called pivots and the corresponding equations are called pivotal equation.

# ASSIGNMENT-4

Date: 2/2/15

* **Problem Statement:**

Write a program in C to solve system of equation using Gauss Seidel Method.

* **Algorithm:**

Gauss-Seidel Iterative Method

begin

steps:

1. Start
2. Read the augmented matrix (aij), [ i=1 to n ; j=1 to (n+1) ]
3. Enter the initial approximation xi =0, [ i= 1 to n ]
4. [Initialization]
   * 1. Set i := 1
5. while (i ≤ n) then

do

a) set s = ai,n+1

b) set j := n

c) while (j ≥ 1) then

do

i ) if(i ≠ j) then

Set s := s-aijxj

ii) set j := j-1

[end of while]

d) set xi = s/aii

e) set i :=i+1

[end of while]

1. print the value of xi [ i=1 to n]
2. stop

* **Source Code:**
* **Discussion :**
* It is also an indirect method for finding a solution of a system of linear equation and is based on the successive better approximation of the values of the unknowns, using an iterative procedure.
* This method is almost identical with Gauss-Jacobi method, expect in considering the iteration formula.
* The sufficient condition for the convergence of the Gauss-Seidel iteration method is that the system of equation must be strictly diagonally dominant.

# ASSIGNMENT-5

Date: 01/12/14

* **Problem Statement:**

Write a program to implement matrix inversion problem.

* **Algorithm:**

[ In this algorithm we will inverse the matrix by Gauss-Jacobi matrix inversion. A is the input 2-D

Matri, in this matrix we allocate 10 location for the row and coloumn. ]

Begin

Steps

1. [initialization ]

I Set k : =0

Ii Read the number of equation N

1. repeat steps while ( k<N )

I set c:= 1/A[k][k]

Ii j:=k

3 repeat steps while j<(2\*N)

I A[k][j]:= ( A[k][j]\*c)

Ii set j:=j+1

[ end of while ]

1. [initialization ]

Set i : =0

1. repeat steps while ( I < N )

I if( i≠k)

a c:=-(A[I][K]/A[K][K])

b j:=k

c repeat steps while( j< (2\*N))

set j:=j+1

[ end of while ]

[end of if ]

[ end of while ]

1. set k:= k+1

[end of step 2 loop ]

1. display A
2. Exit

* **Source Code:**
* **Discussion:**
* Every matrix has a reduced row echelon form and Gauss-Jordan elimination is guaranteed to find it.
* Matrices containing zeros below each pivot are said to be in row echelon from.
* It is an improvement of Gauss-Elimination with the help of diagonal matrix.
* In linear algebra , Gauss-Jordan elimination is an algorithm for getting matrices in reduced row echelon from using elementary row operation.

# ASSIGNMENT-6

Date: 01/12/14

* **Problem Statement:**

Write a program to implement Euler Method .

* **Algorithm:**

[ This algorithm implement the Euler modified method where x is and y is the initial value , h is defined the step length, x be the value that to be find for the given function using this method ]

Begin

Steps

1. start
2. define the function f(x,y)
3. read x,y,h,q
4. while(x<q) repeat

do

i. d=y+(h\*eqsn(x,y));

ii. [initialization]

* + 1. Set i :=1
    2. Set c :=0

iii. while(c==0) repeat

do

a. b=y+((h/2)\*(eqsn(x,y)+eqsn(x+h,d)));

b. printf("\ny(%f)(%d)=%f",x+h,i,b);

c. delta=fabs(b-d);

d. if(delta<(0.0001))

c=1;

else

d=b;

e. set i= i+1

[end of while ]

iv. Set x= x+h

v. set y=b

[end of while ]

1. Print “the value of y(q) “, y
2. stop

* **Source Code:**
* **Discussion:**
* It is a simple and single-step but a rude numerical method for solving an ordinary initial value differential equation ,where the solution will be obtained as a set of tabulated values of variables x and y.
* The Euler method depends on step length h yield a better approximation result. Thus,

this method is too tedious to get a result up to a desired degree of accuracy.

# ASSIGNMENT-7

Date: 01/12/14

* **Problem Statement:**

Write a program to implement trapezoidal and simpson 1/3 method.

* **Algorithm:**

**Trapezoidal method ()**

[ using this method we calculate the value of integration of function f(x) where ,a be the lower value and b be the upper value of the integration and n be the number of intervals.h be the step length,y0,y1,y2 are the sum of the 1st and nth term, odd term, and even term respectively of the intervals n ]

Begin

Steps

1. start
2. read a,b,n
3. set h:=(b-a)/n
4. [ initialization ]

i. set i:=o

ii. set x:=a

1. while(i<=n)repeat steps

a if(i=0 or i=n)

y0=y0+f(x)

b if(i%2≠0)

y1=y1+f(x)

c if(i%2=0)

y2=y2+f(x)

d set x:=x+h

e set i:=i+1

[end of while]

1. set x:=(h/2)\*(y0+2\*(y1+y2))
2. print the value of x
3. stop

**Simpson 1/3 method()**

[ using this method we calculate the value of integration of function f(x) where ,a be the lower value and b be the upper value of the integration and n be the number of intervals.h be the step length,y0,y1,y2 are the sum of the 1st and nth term, odd term, and even term respectively of the intervals n ]

Begin

Steps

1. start
2. read a,b,n
3. set h:=(b-a)/n
4. [ initialization ]

i. set i:=o

ii. set x:=a

1. while(i<=n)repeat steps

a if(i=0 or i=n)

y0=y0+f(x)

b if(i%2≠0)

y1=y1+f(x)

c if(i%2=0)

y2=y2+f(x)

d set x:=x+h

e set i:=i+1

[end of while]

1. set x:=(h/3)\*(y0+(4\*y1)+(2\*y2))
2. print the value of x
3. stop

* **Source Code:**
* **Discussion:**
* We note that the output value has some absolute error.
* In simpson 1/3rd rule the sub-intervals mus be even number.
* Simpson’s 1/3rd rule is an extension of trapezoidal ruie where the integrand is approximation by second order polynomial.

# ASSIGNMENT-8

Date: 01/12/14

* **Problem Statement:**

Write a method to implement Runge -Kutta method.

* **Algorithm:**

[ This algorithm implement the Euler modified method where x is and y is the initial value , h is defined the step length, q be the value that to be find for the given function using this method ]

Begin

Steps

1. start
2. read h,x,y,q
3. while(x<q)

i k1=h\*f(x,y)

ii k2=h\*f(x+h/2,y+k1/2)

iii k3= h\*f(x+h/2,y+k2/2)

iv k4=h\*f(x+h/2,y+k3/2)

v y=y+((1/6)\*(k1+(2\*k2)+(2\*k3)+k4))

vii set x:=x+h

viii print the value of y

[ end of while ]

4 stop

* **Source Code:**

* **Discussion:**
* Here the above program contain both second order and fourth order Runge-Kutta method.but fourth oder method gives us perfect value than second order.
* The error estimate in this method is rather difficult.it can roughly be estimated by the formula |y(xn)-yn

# ASSIGNMENT-9

Date: 01/12/14

* **Problem Statement:**

Write a program to implement least square method.

* **Algorithm:**

[ by this algorithem we can find the solution equation of a given straight line by user.if we assume that the solution equation is y=a+xb,then at first we have to find the values of a and b. here w,x,y,z are temporary variables.here d[]is an array containing the value of x,and c[] is an arry containin the values of y ]

Begin

Steps

1. start
2. [initialization]

Set i:=0

1. read the value of n

4 repeat steps 2 to 6 while(i<n)

I w=w+d[i]

Ii x=x+c[i]

Iii y=y+(d[i]\*d[i])

Iv z=z+(d[i]\*c[i])

V set i:=i+1

[end of while]

5 b=((n\*z)-(w\*x))/((n\*y)-(w\*w))

6 a=(x/n)-(b\*(w/n))

7 print the equation of line is a+bx

8 stop

* **Source Code:**
* **Discussion:**
* This method is easy to implement as there is not so much equation to implement and the formula is simple.
* This method is also can be implemented using matrices where we take equation as input not the arguments.