Numerical Techniques

Teacher: Shubhra Gupta

Name : Manya Kaushik
Roll No : 1620008
Course : B.Sc. (H) Electronics
Semester : V

Index

Experiment - 1

Aim: Program to implement Bisection Method

Algorithm:

1. Start
2. Choose ‘a’ and ‘b’ initial guesses such that f(a)xf(b)<0 or they have opposite signs.
3. If f(a)xf(b)<0 then proceed further otherwise choose the value of ‘a’ and ‘b’ again.
4. Find:

   $x_m = \frac{a+b}{2}$
5. if $x_m = 0$ then x_m is the root of the equation otherwise proceed further
6. if f(a)xf(x_m)<0 then

   a=a ; b=x_m
7. if f(a)xf(x_m)>0 then

   a=x_m ; b=b
8. Check if |a-b| $\leq$ 0.0001 (acceptable error), if it is true then end otherwise go to step 3.

Program :

```
// Program to implement Bisection Method

#include<stdio.h>

#include<math.h>

#define   f(x)x*x*x-18

int main()

{


        float a, b, xn, f0, f1, f2, e;

        int itr = 1;

        up:

        printf("\nEnter the values of a and b:\n");
```

```
scanf("%f%f", &a, &b);
printf("Enter  error:\n");
scanf("%f", &e);
// Calculating  Values
f0 = f(a);
f1 = f(b);
/* Checking of valued. */
if( f0*f1 > 0.0)
{
        printf("Incorrect Initial Guesses.\n");
        goto up;
}
/* using bisection method*/
printf("\nitr\t\ta\t\tb\t\txn\t\tf(xn)\n");
do
{
        xn = (a+b)/2;
        f2 = f(xn);
        printf("%d\t\t%f\t%f\t%f\t%f\n",itr, a, b, xn, f2);

        if(f0*f2 < 0)
        {
                b = xn;
                f1 = f2;
        }
        else
        {
```

```
                a = xn;
                f0 = f2;
            }
            itr = itr + 1;

        }while(fabs(f2)>e);

        printf("\nRoot is: %f", xn);

        return 0;

}
```

Output:

```
Enter the values of a and b:
2.3
2.7
Enter  error:
0.0001
itr     a       b       xn      f(xn)
1       2.300000    2.700000    2.500000
    -2.375000
2       2.500000    2.700000    2.600000
    -0.424002
3       2.600000    2.700000    2.650000
    0.609627
4       2.600000    2.650000    2.625000
    0.087891
5       2.600000    2.625000    2.612500
    -0.169279
6       2.612500    2.625000    2.618750
    -0.041000
7       2.618750    2.625000    2.621875
    0.023369
8       2.618750    2.621875    2.620313
    -0.008831
9       2.620313    2.621875    2.621094
    0.007261
```

Experiment - 2

Aim: Program to implement Regula falsi method

Algorithm:

I. Choose ‘a’ and ‘b’ initial guesses such that f(a)xf(b)<0 or they have opposite signs.

II. If f(a)xf(b)<0 then proceed further otherwise choose the value of ‘a’ and ‘b’ again.

III. Find

$$x_n = \frac{af(b)-bf(a)}{f(b)-f(a)}$$

IV. if f(a)xf(x_m)<0 then

a=a ; b=x_m

V. if f(a)xf(x_m)>0 then

a=x_m ; b=b

VI. Check if |a-b| $\leq$ 0.0001 (acceptable error), if it is true then end otherwise go to step III.

Program :

```
//Program to implement Regula falsi method
#include<stdio.h>
#include<math.h>

#define    f(x)   x*exp(x) - 1

int main()
{
```

```
float a, b, xn, f0, f1, f2, e;
int itr = 1;
up:
printf("\nEnter the values of a and b:\n");
scanf("%f%f", &a, &b);
printf("Enter  error:\n");
scanf("%f", &e);
// Calculating  Values
f0 = f(a);
f1 = f(b);
/* Checking of valued. */
if( f0*f1 > 0.0)
{
        printf("Incorrect Initial Guesses.\n");
        goto up;
}
/* using regular falsi method*/
printf("\nitr\t\ta\t\tb\t\txn\t\tf(xn)\n");
do
{
        xn = (a*f1-b*f0)/(f1-f0);
        f2 = f(xn);
        printf("%d\t\t%f\t%f\t%f\t%f\n",itr, a, b, xn, f2);

        if(f0*f2 < 0)
        {
```

```
                b = xn;
                f1 = f2;
        }
        else
        {
                a = xn;
                f0 = f2;
        }
        itr = itr + 1;

    }while(fabs(f2)>e);

    printf("\nRoot is: %f", xn);

    return 0;

}
```

Output:

```
Enter the values of a and b:
0.5
1
Enter  error:
0.0001

itr     a       b       xn      f(xn)
1       0.500000    1.000000    0.546369
    -0.056436
2       0.546369    1.000000    0.560795
    -0.017452
3       0.560795    1.000000    0.565211
    -0.005332
4       0.565211    1.000000    0.566556
    -0.001623
5       0.566556    1.000000    0.566965
    -0.000493
6       0.566965    1.000000    0.567089
    -0.000150
7       0.567089    1.000000    0.567127
    -0.000046

Root is: 0.567127
```

Experiment - 3

Aim: Program to implement Newton Raphson Method.

Algorithm:

Step1: Define the function and its derivative

Step2: Define error and maximum number of

iterations

Step3: Read initial guess x0,

Step4: If f(x0)==0 then x0 is the root of the equation , exit

Step5: If f'(x0)==0 then x0 is not valid , exit.

Step6: Find x1=x0-(f(x0)/f'(x0))

Step7: Find |x1-x0|

Step8: If |x1-x0|<error then x1 is the root, else

x1=x0 and follow all steps from step 4

Program :

```
//Program to implement Newton Raphson Method
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#define f(x) sin(x)-(x/2)
//let h(x) is the derivative of given equation
#define h(x) cos(x)-(0.5)
int main()
{
```

```
double x0=0, x1=0, f0=0,h0=0, f1=0, error=0.001, difference;
int itr=0,max_no_iterations=1000;
printf("\n The value of initial guess x0: ");
scanf("%lf",&x0);
if(f(x0)==0)
{printf("\n Initial guess x0 is the root of the equation i.e x0= %lf",x0); }
else if(h(x0)==0)
{printf("\ninitial guess is not valid, use  other guess");     }
else {
do{
f0=f(x0);
h0=h(x0);
x1=x0 - (f0/h0);
difference=fabs(x1-x0);
x0=x1;
f1=f(x1);
itr++;
printf("\nitr=%d\t\taproximate_root=%lf\t\tvalue_of_function=%lf",itr,x1,f1);
}
while(difference>error&&itr<max_no_iterations);
printf("\n Difference =%lf", difference);
printf("\nRoot of the equation is %lf after implementing %d iterations", x1, itr);
}
return(0);
}
```

Output:

```
The value of initial guess x0: 2
itr=1       aproximate_root=1.900996
    value_of_function=-0.004520
itr=2       aproximate_root=1.895512
    value_of_function=-0.000014
itr=3       aproximate_root=1.895494
    value_of_function=-0.000000
 Difference =0.000017
Root of the equation is 1.895494 after
    implementing 3 iterations
```

Experiment - 4

Aim: program to implement secant method

Algorithm:

I. Define f(x)

II. Create a function to perform Secant method

III. Read ‘a’ and ‘b’ as initial guesses.

IV. Find:

do

$af(b) - bf(a)$

$f(b) - f(a)$

a=b ; b=x

VI. While(fabs(b-a)>0)

Print x and n

End

Program :

```
// Online C compiler to run C program online
#include<stdio.h>
#include<math.h>

#define    f(x)x*x*x-5*x+3

int main()
{
```

```
float a, b, xn, f0, f1, f2, e;
int itr = 1;

printf("\nEnter the values of a and b:\n");
scanf("%f%f", &a, &b);
printf("Enter  error:\n");
scanf("%f", &e);

f0 = f(a);
f1 = f(b);



printf("\nitr\t\ta\t\tb\t\t\t\txn\n");
do
{
        xn = (a*f1-b*f0)/(f1-f0);
        f2 = f(xn);
        printf("%d\t\t%f\t%f\t%f\n",itr, a, b, xn);



        a=b;
        f0=f1;
                b = xn;
                f1 = f2;



        itr = itr + 1;
```

```
        }while(fabs(a-b)>e);

        printf("\nRoot is: %f", xn);

        return 0;

}
```

Output:

```
Enter the values of a and b:
0
1
Enter  error:
0.0001
itr     a       b               xn
1       0.000000    1.000000    0.750000
2       1.000000    0.750000    0.627907
3       0.750000    0.627907    0.658147
4       0.627907    0.658147    0.656643
5       0.658147    0.656643    0.656620

Root is: 0.656620
```

Experiment - 5

Aim: To solve the problem using LaGrange's interpolation.

Algorithm:

Step1: Define two array X and Y, int i, a,y

Step2: Read the values of arrays X and Y where elements of array Y are the corresponding values of Y for X

Step3: Read the value of a to be calculated

Step4: Define the formula

Step5: Result of the formula will be the desired result

Step6: End

Program :

```
#include<stdio.h>

#include<stdlib.h>

void main()

{

int i,x[4], y[4]; //no of values are 4

double x1,x2,x3,x4,a,y1;

printf("Enter the value of x\n");

for(i=0;i<4;i++)

{

        scanf("%d",&x[i]);
```

```
}
printf("Enter the value of y\n");
for(i=0;i<4;i++)
{
        scanf("%d",&y[i]);
}
printf("Enter the value of a:\n");
scanf("%lf", &a);
x1= ((a-x[1])*(a-x[2])*(a-x[3])*y[0])/((x[0]-x[1])*(x[0]-x[2])*(x[0]-x[3]));
x2=((a-x[0])*(a-x[2])*(a-x[3])*y[1])/((x[1]-x[0])*(x[1]-x[2])*(x[1]-x[3]));
x3=((a-x[0])*(a-x[1])*(a-x[3])*y[2])/((x[2]-x[0])*(x[2]-x[1])*(x[2]-x[3]));
x4=((a-x[0])*(a-x[1])*(a-x[2])*y[3])/((x[3]-x[0])*(x[3]-x[1])*(x[3]-x[2]));
 y1 =  x1+x2+x3+x4;
printf("The value of function is : %lf",y1);
return (0);
}
```

Output:

```
/tmp/RoMSBfjq8S.o
Enter the value of x
1 3 4 6
Enter the  value of y
-3 9 30 132
Enter the value of a:
 1.5
 The value of function is : -1.875000
```

Experiment - 6

Aim: To solve the problem using linear regression.

Algorithm:

1)Read n (no of data points) from the user and declare a,b.

2)Use for loop to read the values of x and y from the user in x[i],y[i].

3) Initailize sumx1=0, sumx2=0, sumy1=0, sumxy=0

4)Use loop to calculate summation-

sumx1 = sumx1 + x[i]

sumx2 = sumx2+ x[i]*x[i]

sumy1 = sumy1 + y[i]

sumxy = sumxy + x[i]*y[i]

5)Use a and b-

a = (n*sumxy-sumx1*sumy1)/(n*sumx2-sumx1*sumx1)

b= (sumy1*sumx2-sumx1*sumxy)/(n*sumx2-(sumx1*sumx1))

6)Display value of a and b.

Program :

```
// To solve the solution of linear regression
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
double a0,a1,x[10],y[10];
double sumx=0,sumxy=0,sumx2=0,sumy=0;
```

```
int i,n;
printf("Linear Regresion \n");
printf("Enter the date value of n: ");
scanf("%d",&n);
printf("Enter the values of X:\n");
for(i=0;i<n;i++)
{
	scanf("%lf",&x[i]);
}

printf("Enter the value of Y:\n");
for(i=0;i<n;i++)
{
	scanf("%lf",&y[i]);
}

for(i=0;i<n;i++)
{
	sumx= sumx + x[i];
	sumx2 = sumx2 + x[i]*x[i];
	sumy= sumy + y[i];
	sumxy= sumxy + x[i]*y[i];
}


printf("X\t\tY\t\tX^2\t\t\tXY\n");
for(i=0;i<n;i++)
```

```
{
printf("%lf\t%lf\t%lf\t\t%lf\n",x[i],y[i],x[i]*x[i],x[i]*y[i]);
}

a0= ((sumy*sumx2) - (sumx*sumxy))/((n*sumx2)-(sumx*sumx));
a1= ((n*sumxy)-(sumx*sumy))/((n*sumx2)-(sumx*sumx));
printf("Y=%lfx + %lf",a1,a0);
getch();
```

Output:

```
Linear Regresion
Enter the date value of n: 9
Enter the values of X:
1.2
2.1
2.8
4.1
4.9
6.2
7.1
7.9
8.9
Enter the value of Y:
4.2
6.8
9.8
13.4
15.5
19.6
21.6
25.4
28.6
```

```
X               Y               X^2    X
Y
1.200000        4.200000        1.440000
5.040000
2.100000        6.800000        4.410000
14.280000
2.800000        9.800000        7.840000
27.440000
4.100000        13.400000       16.81000
0               54.940000
4.900000        15.500000       24.01000
0               75.950000
6.200000        19.600000       38.44000
0               121.520000
7.100000        21.600000       50.41000
0               153.360000
7.900000        25.400000       62.41000
0               200.660000
8.900000        28.600000       79.21000
0               254.540000
Y=3.104929x + 0.506355
[Process completed - press Enter]
```

Experiment - 7

Aim: Program to implement Trapezoidal rule.

Algorithm:

Call the function

Define h, n,a,b,ans,I,i array X, array Y

Define f(x), sum=0

Calculate h

Using ‘for’ loop from i=0 to i<n receive the values of array x

Using ‘for’ loop calculate the value of corresponding y as y= f(x[i])

Calculate sum of the middle term

Calculate integral as

I=h[y[0]+y[n]+2*Sum]/2

Print the result

Program :

```
//Program to implement Trapezoidal rule.#include<stdio.h>
#include<math.h>
 void main()
{
 float a,b,h,sum=0,I;
 int n,i;
 float x[50],y[50];
 float f(float);
```

```
printf("Finding the intergal uding Trapezoidal method\n");
 printf("Enter the value of a: ");
 scanf("%f", &a);
 printf("Enter the value of b:");
 scanf("%f",&b);
 printf("Enter the value of n: ");
 scanf("%d",&n);
 h=(b-a)/n;
printf("the value of h is %.1f",h);
 x[0]=a;
 x[n]=b;
for(i=1;i<n;i++)
   {
   x[i]=x[i-1]+h;
   }

   //loop for calculating the values of y
   for(i=0;i<n+1;i++)
   {
   y[i]=f(x[i]);
   }
   printf("\n\txn\t\t\tyn\n");
   for(i=0;i<n+1;i++)
   {
   printf("\t%lf\t\t%lf\n",x[i],y[i]);
   }
   //loop for finding sum of middle terms
```

```
for(i=1;i<n;i++)
   {
   sum=sum+y[i];
   }


 I=((y[0]+y[n]+(2*sum))*h/2);
 printf("The value of the integration is %f",I);
}
float f(float x)
{
 return(exp(-(x*x)));
}
```

Output:

```
Finding the intergal uding Trapezoidal m
ethod
Enter the value of a: 0
Enter the value of b:1
Enter the value of n: 10
the value of h is 0.1
        xn                      yn
        0.000000                1.000000
        0.100000                0.990050
        0.200000                0.960789
        0.300000                0.913931
        0.400000                0.852144
        0.500000                0.778801
        0.600000                0.697676
        0.700000                0.612626
        0.800000                0.527292
        0.900000                0.444858
        1.000000                0.367879
The value of the integration is 0.746211
[Process completed (code 40) - press Ent
er]
```

Experiment - 8

Aim: To solve the problem by simpsons rule

Algorithm :

• Start

• Define a,b,n,h,z,sum = 0,Result.

• Find h = (b-a)/n.

• Define x[i],f[i].

• for(i=0;i<=n;i++)

Run loop to find x[i],f[i].

• for(i=0;i<=n-1;i++)

Run if(i%2==0)

Sum = sum + 2*f(i)

Else sum = sum + 4*f(i).

• Compute Result from the formula

Result = h(f0+sum+f(n))/3.

• Stop

Program :

```
Program for simpsons rule*/
#include<stdio.h>
#include<math.h>
void main()
{
float a,b,n,h,x[50],f[50],result,z,sum = 0;
int i;
```

```
printf("enter the value of a and b:\n");
scanf("%f%f",&a,&b);
printf("Enter the value of interval(n) :\n");
scanf("%f",&n);
h= (b-a)/n;
printf("the value of h is=%f\n",h);
for(i=0;i<=n;i++)
{
 x[0] = a;
 x[i+1] = x[i] + h;
 scanf("x[%d]= %f\n",i,x[i]);
}
for (i=0;i<=n;i++)
{
 f[i] = exp(-x[i]*x[i]);
 scanf("f[%d] = %f\n",i,f[i]);
}
printf("step\t x\t \t f\t");
for(i=0;i<=n;i++)
{
printf("\n%d\t%f\t%f",i,x[i],f[i]);
}
for(i=1;i<=n-1;i++)
{
 if(i%2==0)
 {
```

```
 sum = sum + 2*f[i];
 }
else
 {
 sum = sum + 4*f[i];
 }}
z = f[0] + f[10];
{
result = h*(sum + z)/3;
printf("\n\nvalue by simpsons 1/3 rule = %f", result);
}
return 0;
}
```

Output:

```
enter the value of a and b:
0
1
Enter the value of interval(n) :
10
the value of h is=0.100000
step     x               f
0       0.000000        1.000000
1       0.100000        0.990050
2       0.200000        0.960789
3       0.300000        0.913931
4       0.400000        0.852144
5       0.500000        0.778801
6       0.600000        0.697676
7       0.700000        0.612626
8       0.800000        0.527292
9       0.900000        0.444858
10      1.000000        0.367879

value by simpsons 1/3 rule = 0.746825
[Process completed (code 39) - press Ent
er]
```

Experiment - 9

Aim: program to implement Euler's method

Algorithm:

1)Declare f(x,y).

2)Read xo,yo,xn,h from the user(where xn is the final value of x).

3)Apply for(x=x0;x<xn;x+=h) for computing values of y by using formula-

yn+1=yn+h*f(xn,yn).

4)Display the values of x and y on the screen.

5) print the final value of y

Input:

```
#include<stdio.h>
#include<math.h>
void main()
{
 double x0,y0,xn,h,x,y;
 double f(double x, double y);
 printf("Enter the initial value of x =");
 scanf("%lf",&x0);
 printf("Enter initial value of y w.r.t x: ");
 scanf("%lf",&y0);
printf("Enter the final value of x:");
 scanf("%lf",&xn);
 printf("Enter the value of h: ");
```

```
 scanf("%lf",&h);
 printf(" xvalue \t yvalue");
 for(x=x0;x<xn;x+=h)
 {
 y=y0+h*f(x,y0);
 printf("\n");
 printf("%lf \t %lf \n",x+h,y);
 y0=y;
 }
printf("the final value of y at x=%lf is =%lf",x,y);
}

double f(double x, double y)
{
 return (-(x*y));
}
```

Output:

```
Enter the initial value of x =0
Enter initial value of y w.r.t x: 1
Enter the final value of x:0.2
Enter the value of h: 0.05
 xvalue          yvalue
0.050000        1.000000

0.100000        0.997500

0.150000        0.992513

0.200000        0.985069
the final value of y at x=0.200000 is =0
.985069
[Process completed (code 47) - press Ent
```

Experiment - 10

Aim: Program to implement Gauss seidel Iteration

Algorithm :

• Start

• Define function f(x,y,z),s(x,y,z),t(x,y,z)

• Choose initial guesses x0,y0,z0 .

• Choose pre-specified tolerable error.

• Run do loop a. dx=x1,dy=y1,dz=z1; b. x1=f(x0,y0,z0),y1=s(x1,y0,z0),z1=t(x1,y1,z0); c. x0=x1,y0=y1,z0=z1;

• While |dx-x1| > error,|dy-y1|>error,|dz-z1|>error then goto (5) otherwise goto (7).

• Display x1,y1,z1 as root.

• Stop

Input:

```
#include <stdio.h>

#include <conio.h>

#include <math.h>


double f(double x, double y, double z)

{

  return ((95-11*y+4*z)/83);

}


double s(double x, double y,double z)

{

  return ((104-7*x-13*z)/52);
```

```
}

double t(double x, double y,double z)
{
  return ((71-3*x-8*y)/29);
}

void main()
{
printf("implement gauss seidel method\n");
  double x0,y0,z0,x1=0,y1=0,z1=0,dx,dy,dz,error=0.0001;
  int iteration=0;
  printf("Enter initial guesses:\n");
  scanf("%f%f%f",&x0,&y0,&z0);

  printf("iteration\t x\t y\t\t\t z");
  do
  {
   dx=x1;
   dy=y1;
   dz=z1;

    x1=f(x0,y0,z0);
    y1=s(x1,y0,z0);
    z1=t(x1,y1,z0);
    iteration++;
```

```
 printf("\n%d\t%f\t%f\t\t %f",iteration,x1,y1,z1);
  x0=x1;
  y0=y1;
  z0=z1;

 }while(fabs(dx-x1)>error, fabs(dy-y1)>error , fabs(dz-z1)>error);

 printf("\n\nFinally,\n");
 printf("x=%f \ny=%f \nz=%f \n",x1,y1,z1);
 printf("Iteration=%d",iteration);
 getch();
}
```

Output:

```
implement gauss seidel method
Enter initial guesses:
0
0
0
iteration        x       y
z
1       1.144578        1.845922
1.820651
2       0.987680        1.411880
1.956618
3       1.051756        1.369263
1.961746
4       1.057652        1.367187
1.961708

Finally,
x=1.057652
y=1.367187
z=1.961708
Iteration=4
[Process completed (code 10) - press Ent
er]
```

Experiment - 11

Aim: Solve with Runge Kutta method.

Program :

```
#include<stdio.h>
#include<math.h>
float f(float x,float y);
int main()
{
   float x0,y0,m1,m2,m3,m4,m,y,x,h,xn;
   printf("Enter x0,y0,xn,h:");
   scanf("%f %f %f %f",&x0,&y0,&xn,&h);
   x=x0;
   y=y0;
   printf("\n\nX\t\tY\n");
   while(x<xn)
   {
      m1=f(x0,y0);
      m2=f((x0+h/2.0),(y0+m1*h/2.0));
      m3=f((x0+h/2.0),(y0+m2*h/2.0));
      m4=f((x0+h),(y0+m3*h));
      m=((m1+2*m2+2*m3+m4)/6);
      y=y+m*h;
      x=x+h;
      printf("%f\t%f\n",x,y);
   }
}
float f(float x,float y)
{
   float m;
   m=x-y/x+y;
   return m;
}
```

Output:

```
Enter x0,y0,xn,h:0 2 2 0.5


X               Y
0.500000        1.621356
1.000000        1.242713
1.500000        0.864069
2.000000        0.485426

Process returned 16384 (0x4000)   execution time : 5.825 s
Press any key to continue.
```