**NM Lab Programs**

**Unit 1:**

**Program 1: Write a C Program for Bisection Method**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define E 0.00001

#define F(x) x\*x-4\*x-10

int main()

{

float x1,x2,x0,f1,f2,f0,r;

printf("Enter initial guesses\n");

scanf("%f%f",&x1,&x2);

f1 = F(x1);

f2 = F(x2);

if(f1\*f2>0)

{

printf(" Initial guesses do not bracket roots\n");

}

else

{

while(1)

{

x0 = (x1+x2)/2;

f0 = F(x0);

if(f1\*f0<0)

{

x2 = x0;

f2 = f0;

}

else

{

x1 = x0;

f1 = f0;

}

if(fabs(x2-x1)<E)

{

r = x2;

break;

}

}

printf("The root = %f\n",r);

printf("f(%f) = %f\n",r,F(r));

getch();

return 0;

}

Output:

Program 2: Write a C program for secant method

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define E 0.001

#define F(x) x\*x-4\*x-10

int main()

{

float x1,x2,x3,f1,f2,f3,r;

int i=0;

printf("Enter intial guesses\n");

scanf("%f%f",&x1,&x2);

while(1)

{

f1 = F(x1);

f2 = F(x2);

x3 = x2- (f2\*(x2-x1))/(f2-f1);

if(fabs((x3-x2)/x3)<E)

{

r = x3;

break;

}

else

{

x1 = x2;

x2 = x3;

}

i=i+1;

}

printf("The root = %f\n",r);

printf("f(%f) = %f\n",r,F(r));

printf("Number of iterations = %d\n",i);

getch();

return 0;

}

Output:

**Program 3: Program for Newton Raphson method**

#include<conio.h>

#include<stdio.h>

#include<math.h>

#define F(x) x\*x+4\*x-9

#define D(x) 2\*x+4

#define E 0.001

int main()

{

float x0,x1,f1,f2,r;

int i=1;

printf("Enter initial guess\n");

scanf("%f",&x0);

while(1)

{

f1 = F(x0);

f2 = D(x0);

x1 = x0-f1/f2;

if((fabs(x1-x0)/x1)<E||i>=20)

{

r=x1;

break;

}

x0 = x1;

}

printf("The root = %f",r);

getch();

return 0;

}

Enter initial guess

4

The root = 1.605551

Program 4: Program for Secant method

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define E 0.001

#define F(x) x\*x-6\*x+8

#define G(x) x\*x-5\*x+8

int main()

{

float x0,x1,r;

int i=0;

printf("Enter intial guess\n");

scanf("%f",&x0);

while(1)

{

x1 = G(x0);

if((fabs(x1-x0)<E)||i>=30)

{

r=x1;

break;

}

else

{

x0 = x1;

}

i=i+1;

}

printf("The root = %f\n",r);

printf("The number of iterations =%d",i);

getch();

return 0;

}

Output:

**Program 5:** **C program for Synthetic Division**

#include<stdio.h>

#include<conio.h>

#include<math.h>

int main()

{

int a[30],b[30];

int i, m, n, c;

printf ("Enter degree of polynomial\n");

scanf ("%d",&n);

printf ("Enter coefficients of dividend polynomial\n");

for(i=n; i>=0;i--)

{

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

}

printf ("Enter the constant term of divisor polynomial\n");

scanf ("%d", &c);

b[n]=0;

m=n;

while(m>0)

{

b[m-1]=a[m]+b[m]\*c;

m = m-1;

}

printf ("Quotient:");

m = n-1;

while(m>=0)

{

if(b[m]!=0)

printf("%dx^%d+",b[m],m);

m = m-1;

}

getch();

return 0;

}

**Output:**

Enter degree of polynomial

4

Enter coefficients of dividend polynomial

2 1 0 -6 4

Enter the constant term of divisor polynomial

2

**Quotient:** 2x^3+5x^2+10x^1+14x^0

**Program 6 :** **C program for Synthetic Division**

#include<stdio.h>

#include<conio.h>

#include<math.h>

int main()

{

int a[30],b[30];

int i, m, n, c;

printf ("Enter degree of polynomial\n");

scanf ("%d",&n);

printf ("Enter coefficients of dividend polynomial\n");

for(i=n; i>=0;i--)

{

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

}

printf ("Enter the constant term of divisor polynomial\n");

scanf ("%d", &c);

b[n]=0;

m=n;

while(m>0)

{

b[m-1]=a[m]+b[m]\*c;

m = m-1;

}

printf ("Quotient:");

m = n-1;

while(m>=0)

{

if(b[m]!=0)

printf("%dx^%d+",b[m],m);

m = m-1;

}

getch();

return 0;

}

**Output:**

Enter degree of polynomial

4

Enter coefficients of dividend polynomial

2 1 0 -6 4

Enter the constant term of divisor polynomial

2

**Quotient:** 2x^3+5x^2+10x^1+14x^0+

**Program 7: C Program for Calculating Multiple Roots using Newton Raphson Method**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define F(x) (a[4]\*x\*x\*x\*x+a[3]\*x\*x\*x+a[2]\*x\*x+a[1]\*x+a[0])

#define FD(x) (4\*a[4]\*x\*x\*x+3\*a[3]\*x\*x+2\*a[2]\*x+a[1])

#define E 0.0001

float a[20],q[20];

int main()

{

float x0,xr,fx0,fdx0,Er,c;

int i,n,m;

printf("Enter degree of polynomial\n");

scanf("%d",&n);

printf("Enter coefficients of dividend polynomial\n");

for(i=n;i>=0;i--)

scanf("%f",&a[i]);

printf("Enter initial guess\n");

scanf("%f",&x0);

while(n>1)

{

while(1)

{

fx0 = F(x0);

fdx0 = FD(x0);

xr = x0-fx0/fdx0;

Er = (xr-x0)/xr;

if(fabs(Er)<E)

{

printf("root%d=%f\n",n,xr);

break;

}

x0 = xr;

}

c = xr;

q[n]=0;

m = n-1;

while(m>=0)

{

q[m]=a[m+1]+q[m+1]\*c;

m--;

}

for(i=n;i>=0;i--)

{

a[i]=q[i];

}

n = n-1;

x0 = xr;

}

xr = a[0]/a[1];

printf("root%d=%f\n",n,xr);

getch();

return 0;

}

**Output:**

Enter degree of polynomial

4

Enter coefficients of dividend polynomial

1 -2 -13 38 -24

Enter initial guess

0.5

root4=1.000000

root3=2.000000

root2=3.000000

root1=4.000000

**Program 8:** **C program for evaluating the polynomial using Horner’s Method**

#include<stdio.h>

#include<conio.h>

#define P(x) (a[4]\*x\*x\*x\*x+a[3]\*x\*x\*x+a[2]\*x\*x+a[1]\*x+a[0])

float a[30],b[30];

int main()

{

float x;

int i,n;

printf("Enter degree of polynomial\n");

scanf("%d",&n);

printf("Enter %d coefficients of dividend polynomial\n",n+1);

for(i=n;i>=0;i--)

{

scanf("%f",&a[i]);

}

printf("Enter the polynomial at which polynomial to be evaluated\n");

scanf("%f",&x);

b[n]=a[n];

while(n>0)

{

b[n-1] = a[n-1]+b[n]\*x;

n = n-1;

}

printf("Value of polynomial p(%f)= %f",x,b[0]);

getch();

return 0;

}

Output:

Enter degree of polynomial

3

Enter 4 coefficients of dividend polynomial

3 -4 5 -6

Enter the polynomial at which polynomial to be evaluated

2

Value of polynomial p(2.000000)= 12.000000

**Unit 2:**

**Lab 9:** **C program for Lagrange Interpolation**

#include<stdio.h>

#include<conio.h>

int main()

{

float x[100], y[100], xp, yp=0, p;

int i,j,n;

printf("Enter number of data: ");

scanf("%d", &n);

printf("Enter data:\n");

for(i=1;i<=n;i++)

{

printf("x[%d] = ", i);

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

printf("y[%d] = ", i);

scanf("%f", &y[i]);

}

printf("Enter interpolation point: ");

scanf("%f", &xp);

for(i=1;i<=n;i++)

{

p=1;

for(j=1;j<=n;j++)

{

if(i!=j)

{

p = p\* (xp - x[j])/(x[i] - x[j]);

}

}

yp = yp + p \* y[i];

}

printf("Interpolated value at %.3f is %.3f.", xp, yp);

getch();

}

**Output:**

Enter number of data: 3

Enter data:

x[1] = 2

y[1] = 4

x[2] = 3

y[2] = 9

x[3] = 4

y[3] = 16

Enter interpolation point: 2.5

Interpolated value at 2.500 is 6.250

**Lab 10: C Program for Newton’s Interpolation**

#include<stdio.h>

#include<conio.h>

int main()

{

int n,i,j;

float v=0,p,xv,x[10],fx[10],a[10];

printf("Enter number of points\n");

scanf("%d",&n);

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

scanf("%f",&xv);

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

{

printf("Enter the value of x and fx at i = %d\n",i);

scanf("%f%f",&x[i],&fx[i]);

}

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

a[i] = fx[i];

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

{

for(j=n-1;j>i;j--)

{

a[j]=(a[j]-a[j-1])/(x[j]-x[j-1-i]);

}

}

v=0;

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

{

p=1;

for(j=0;j<=i-1;j++)

p = p\*(xv-x[j]);

v = v+a[i]\*p;

}

printf("Interpolation value = %f",v);

getch();

return 0;

}

Output:

Enter number of points

4

Enter the value of x

2.5

Enter the value of x and fx at i = 0

1 0

Enter the value of x and fx at i = 1

2 0.3010

Enter the value of x and fx at i = 2

3 0.4771

Enter the value of x and fx at i = 3

4 0.6021

Interpolation value = 0.400050

**Lab 11: C program for Newton’s Forward Difference Interpolation**

…………………………………………………………………………………………..

#include<stdio.h>

#include<conio.h>

int factorial(int n)

{

if(n<=1)

return 1;

else

return n\*factorial(n-1);

}

int main()

{

int n,i,j,k;

float v=0, p, xp,x[10],fx[10], fd[10], h, s;

printf ("Enter number of points\n");

scanf ("%d",&n);

printf ("Enter the value at which interpolated value is needed\n");

scanf ("%f",&xp);

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

{

printf ("Enter the value of x and fx at i=%d\n",i);

scanf ("%f%f",&x[i],&fx[i]);

}

h = x[1]-x[0];

s = (xp-x[0])/h;

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

{

fd[i]=fx[i];

}

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

{

for(j=n-1;j>i;j--)

{

fd[j]=(fd[j]-fd[j-1]);

}

}

v = fd[0];

for(i=1;i<n;i++)

{

p=1;

for(k=1;k<=i;k++)

{

p = p\*(s-k+1);

}

v = v+(fd[i]\*p)/factorial(i);

}

printf ("Interpolation value = %f",v);

getch ();

return 0;

}

First Run

Enter number of points

4

Enter the value at which interpolated value is needed

0.95

Enter the value of x and fx at i=0

0.9 0.7833

Enter the value of x and fx at i=1

1.0 0.8415

Enter the value of x and fx at i=2

1.1 0.8912

Enter the value of x and fx at i=3

1.2 0.9320

**Interpolation value = 0.813437**

**S**econd Run

Enter number of points

6

Enter the value at which interpolated value is needed

0.0045

Enter the value of x and fx at i=0

0 1.121

Enter the value of x and fx at i=1

0.001 1.123

Enter the value of x and fx at i=2

0.002 1.1255

Enter the value of x and fx at i=3

0.003 1.127

Enter the value of x and fx at i=4

0.004 1.128

Enter the value of x and fx at i=5

0.005 1.1285

**Interpolation value = 1.128400**

**Lab 12: C Program for Newton Backward difference formula**

#include<stdio.h>

#include<conio.h>

int factorial(int n)

{

if(n<=1)

return 1;

else

return n\*factorial(n-1);

}

int main()

{

int n,i,j,k;

float v=0, p,xp,x[10],fx[10], bd[10],h,s;

printf("Enter number of points\n");

scanf("%d",&n);

printf("Enter the value at which interpolated value is needed\n");

scanf("%f",&xp);

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

{

printf("Enter the value of x and fx at i=%d\n",i);

scanf("%f%f",&x[i],&fx[i]);

}

h = x[1]-x[0];

s = (xp-x[n-1])/h;

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

{

bd[i]=fx[i];

}

for(i=n-1;i>0;i--)

{

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

{

bd[j]=(bd[j+1]-bd[j]);

}

}

v = bd[n-1];

for(i=1;i<n;i++)

{

p=1;

for(k=1;k<=i;k++)

{

p = p\*(s+k-1);

}

v = v+(bd[n-i-1]\*p)/factorial(i);

}

printf("Interpolation value = %f",v);

getch();

return 0;

}

Output:

First Run:

Enter number of points

4

Enter the value at which interpolated value is needed

1.15

Enter the value of x and fx at i=0

0.9 0.7833

Enter the value of x and fx at i=1

1.0 0.8412

Enter the value of x and fx at i=2

1.1 0.8912

Enter the value of x and fx at i=3

1.2 0.9320

Interpolation value = 0.912831

Second Run:

Enter number of points

5

Enter the value at which interpolated value is needed

1979

Enter the value of x and fx at i=0

1974 40

Enter the value of x and fx at i=1

1976 43

Enter the value of x and fx at i=2

1978 48

Enter the value of x and fx at i=3

1980 52

Enter the value of x and fx at i=4

1982 57

Interpolation value = 50.117188

**Lab 13: C program for linear regression**

#include<stdio.h>

#include<conio.h>

int main()

{

int i,j,k,n;

float a=0,b=0, x[10],y[10],sx=0,sy=0,sxy=0,sx2=0;

printf("Enter number of points\n");

scanf("%d",&n);

printf("Enter value of x and fx\n");

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

{

scanf("%f%f",&x[i],&y[i]);

}

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

{

sx = sx+x[i];

sy = sy+y[i];

sxy = sxy+x[i]\*y[i];

sx2 = sx2+x[i]\*x[i];

}

b = ((n\*sxy)-(sx\*sy))/((n\*sx2)-(sx\*sx));

a = (sy/n)-(b\*sx/n);

printf("Fitted line is %f+%fx",a,b);

getch();

}

Output:

Enter number of points

5

Enter value of x and fx

1 3

2 5

3 7

4 10

5 12

Fitted line is 0.500000+2.300000x

**Lab 14: C program to implement the polynomial regression**

#include<stdio.h>

#include<conio.h>

#include<math.h>

int main()

{

int i,j,k,m,n;

float a[20][20],b[20],z[20],x[20],fx[20];

float sum, pivot, term;

printf("Enter number of data points\n");

scanf("%d",&n);

printf("Enter degree of polynomial\n");

scanf("%d",&m);

printf("Enter data points\n");

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

{

scanf("%f%f",&x[i],&fx[i]);

}

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

{

for(j=0;j<=m;j++)

{

sum=0;

for(k=0;k<n;k++)

{

sum = sum+pow(x[k],i+j);

}

a[i][j]=sum;

}

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

{

sum=0;

for(k=0;k<n;k++)

{

sum = sum+fx[i]\*pow(x[k],i);

}

b[i]=sum;

}

for(k=0;k<m;k++)

{

pivot = a[k][k];

if(pivot<0.0001)

printf("Method failed");

else

for(i=k+1;i<=m;i++)

{

term = a[i][k]/pivot;

for(j=0;j<=m;j++)

{

a[i][j]=a[i][j]-a[k][j]\*term;

}

b[i]=b[i]-b[k]\*term;

}

}

}

z[m] = b[m]/a[m][m];

for(i=m-1;i>=0;i--)

{

sum=0;

for(j=i+1;j<=2;j++)

{

sum = sum+a[i][j]\*z[j];

}

z[i]=(b[i]-sum)/a[i][i];

}

printf("The polynomial of regression is :\n");

printf("y=%f+%fx",z[0],z[1]);

for(i=2;i<=m;i++)

{

printf("+%fx^%d",z[i],i);

}

getch();

return 0;

}

**Lab 15: C program for nonlinear Regression with exponential Model**

#include<stdio.h>

#include<conio.h>

#include<math.h>

int main()

{

int n,i,j,k;

float a=0,b=0,r, x[10],y[10],sx, slogy=0, sxy=0, sx2=0;

printf("Enter number of points\n");

scanf("%d",&n);

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

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

{

scanf("%f%f",&x[i],&y[i]);

}

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

{

sx = sx+x[i];

slogy = slogy+log(y[i]);

sxy = sxy+x[i]\*log(y[i]);

sx2 = sx2+x[i]\*x[i];

}

b = ((n\*sxy)-(sx\*slogy))/((n\*sx2)-(sx\*sx));

r = (slogy/n)-(b\*sx/n);

a = exp(r);

printf("Fitted Curve is y = %fe^%fx",a,b);

getch();

}

Output:

Enter number of points

5

Enter the value of x and fx

2 4.077

4 11.084

6 30.128

8 81.897

10 222.62

Fitted Curve is y = 1.499900e^0.500009x

**Unit 3:**

**Lab 16: C Program for Two-Point Forward Difference Formula**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define PI 3.1416

#define f(x) sin(x)+1

int main()

{

float angle,h,x,d,x1,x2;

printf("Enter angle in degree\n");

scanf("%f",&angle);

printf("Enter increment\n");

scanf("%f",&h);

x = PI\*angle/180;

x1 = f(x+h);

x2 = f(x);

d = (x1-x2)/h;

printf("The value of derivative =%f\n",d);

getch();

return 0;

}

Output:

Enter angle in degree

45

Enter increment

0.1

The value of derivative =0.670602

**Lab 17: C Program for Two Point Backward Difference Formula**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define PI 3.1416

#define f(x) sin(x)+1

int main()

{

float angle,h,x,d,x1,x2;

printf("Enter angle in degree\n");

scanf("%f",&angle);

printf("Enter increment\n");

scanf("%f",&h);

x = PI\*angle/180;

x1 = f(x-h);

x2 = f(x);

d = (x2-x1)/h;

printf("The value of derivative =%f\n",d);

getch();

return 0;

}

Output:

Enter angle in degree

45

Enter increment

0.1

The value of derivative =0.741253

Lab 18: **C program for Three Point Formula**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define PI 3.1416

#define f(x) sin(x)+1

int main()

{

float angle,h,x,d,x1,x2;

printf("Enter angle in degree\n");

scanf("%f",&angle);

printf("Enter increment\n");

scanf("%f",&h);

x = PI\*angle/180;

x1 = f(x+h);

x2 = f(x-h);

d = (x1-x2)/(2\*h);

printf("The value of derivative =%f\n",d);

getch();

return 0;

}

Output:

Enter angle in degree

45

Enter increment

0.01

The value of derivative =0.707096

**Lab 19:** **C Program for computing differentiation using divided difference polynomial**

#include<stdio.h>

#include<conio.h>

int main()

{

int n,i,k,j;

float factor, term, vod, xv,x[10],fx[10],a[10];

printf("Enter the number of points\n");

scanf("%d",&n);

printf("Enter value of data points\n");

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

{

scanf("%f%f",&x[i],&fx[i]);

}

printf("Enter the value at which derivative is required\n");

scanf("%f",&xv);

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

{

a[i] = fx[i];

}

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

{

for(j=n-1;j>i;j--)

{

a[j] = (a[j]-a[j-1])/(x[j]-x[j-1-i]);

}

}

vod = a[1];

for(i=2;i<n;i++)

{

term = 0;

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

{

factor = 1;

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

{

if(k!=j)

factor = factor\*(xv-x[k]);

}

term = term+factor;

}

vod = vod+(a[i]\*term);

}

printf("Value of first derivative =%f\n",vod);

}

Output:

Enter the number of points

5

Enter value of data points

3 -13

5 23

11 899

27 17315

34 35606

Enter the value at which derivative is required

10

Value of first derivative =233.000000

**Lab 20:** **C Program for calculating value of derivative using forward difference**

#include<stdio.h>

#include<conio.h>

int fact(int n)

{

if(n==1)

return 1;

else

return n\*fact(n-1);

}

int main()

{

int n,i,j,k;

float val=0, p,xp,x[10],fx[10],fd[10],h,s,term1,term2, prev;

printf("Enter the number of points\n");

scanf("%d",&n);

printf("Enter value of x and fx\n");

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

{

scanf("%f%f",&x[i],&fx[i]);

}

printf("Enter the value at which derivative is needed\n");

scanf("%f",&xp);

h = x[1]-x[0];

s = (xp-x[0])/h;

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

{

fd[i]=fx[i];

}

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

{

for(j=n-1;j>i;j--)

{

fd[j] = (fd[j]-fd[j-1]);

}

}

val = fd[1];

prev = 1;

for(i=2;i<n;i++)

{

term1 = 1;

for(k=2;k<=i;k++)

{

term1 = term1\*(s-k+2);

}

term2 = (s-i+1)\*prev;

prev = (term1+term2);

val = val +(fd[i]\*(term1+term2))/(fact(i));

}

val = val/h;

printf("Value of first derivative = %f",val);

getch();

return 0;

}

Output:

Enter the number of points

7

Enter value of x and fx

1.0 2.71

1.2 3.32

1.4 4.05

1.6 4.95

1.8 6.04

2.0 7.38

2.2 9.02

Enter the value at which derivative is needed

1.2

Value of first derivative = 3.258337

**Lab 21: C Program for calculating derivative using backward divided differences**

#include<stdio.h>

#include<conio.h>

int fact(int n)

{

if(n==1)

return 1;

else

return n\*fact(n-1);

}

int main()

{

int n,i,j,k;

float val=0, p,xp,x[10],fx[10],bd[10],h,s,term1,term2, prev;

printf("Enter the number of points\n");

scanf("%d",&n);

printf("Enter value of x and fx\n");

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

{

scanf("%f%f",&x[i],&fx[i]);

}

printf("Enter the value at which derivative is needed\n");

scanf("%f",&xp);

h = x[1]-x[0];

s = (xp-x[n-1])/h;

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

{

bd[i]=fx[i];

}

for(i=n-1;i>0;i--)

{

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

{

bd[j] = (bd[j+1]-bd[j]);

}

}

val = bd[n-2];

prev = 1;

for(i=2;i<n;i++)

{

term1 = 1;

for(k=2;k<=i;k++)

{

term1 = term1\*(s+k-2);

}

term2 = (s+i-1)\*prev;

prev = (term1+term2);

val = val +(bd[n-i-1]\*(term1+term2))/(fact(i));

}

val = val/h;

printf("Value of first derivative = %f",val);

getch();

return 0;

}

Output:

Enter the number of points

7

Enter value of x and fx

1.0 2.71

1.2 3.32

1.4 4.05

1.6 4.95

1.8 6.04

2.0 7.38

2.2 9.02

Enter the value at which derivative is needed

2.2

Value of first derivative = 8.870843

**Lab 22: C program for calculating maxima and minima of tabulated functions**

#include<conio.h>

#include<stdio.h>

#include<math.h>

int main()

{

int n,i,j;

float val, x[10],fx[10],fd[10],h,s1,s2,x1,x2,a,b,c;

printf("Enter the number of poinnts\n");

scanf("%d",&n);

printf("Enter values of x and fx\n");

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

{

scanf("%f%f",&x[i],&fx[i]);

}

h = x[1]-x[0];

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

{

fd[i]=fx[i];

}

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

{

for(j=n-1;j>i;j--)

{

fd[j]=(fd[j]-fd[j-1]);

}

}

a = (1/2.0)\*fd[3];

c = fd[1]-((1/2.0)\*fd[2])+((1/3.0)\*fd[3]);

b = fd[2]-fd[3];

s1 = (-b+sqrt(b\*b-4\*a\*c))/(2\*a);

s2 = (-b-sqrt(b\*b-4\*a\*c))/(2\*a);

x1 = x[0]+s1\*h;

x2 = x[0]+s2\*h;

val = (fd[2]+(((6\*s1-6)\*fd[3])/6))/(h\*h);

if(val<0)

printf("Maxima exists at x =%f\n",x1);

else

printf("Minima exists at x =%f\n",x1);

val = (fd[2]+(((6\*s2-6)\*fd[3])/6))/(h\*h);

if(val<0)

printf("Maxima exists at x =%f\n",x2);

else

printf("Minima exists at x =%f\n",x2);

getch();

return 0;

}

Output:

Enter the number of points

4

Enter values of x and fx

0 -5

1 -7

2 -3

3 13

Minima exists at x =1.000000

Maxima exists at x =-1.000000

**Lab 23: Write a C program to integrate a function using Trapezoidal rule**

#include<stdio.h>

#include<conio.h>

#define F(x) (x\*x\*x+3)

int main()

{

float h, x0,x1,x2,x,v,f0,f1;

printf("Enter upper limit and lower limit\n");

scanf("%f%f",&x1,&x0);

h = x1-x0;

f1 = F(x1);

f0 = F(x0);

v = h\*(f0+f1)/2;

printf("The value of integration = %f",v);

getch();

return 0;

}

**Output:**

Enter upper limit and lower limit

8 2

The value of integration = 1578.000000

**Lab 24: Write a C program to integrate a given function using Simpson’s 1/3 rule**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define f(x) 3\*x\*x+2\*x-5

int main()

{

float x0,x1,x2,f0,f2,f1,i,h,a,r;

printf("Enter lower limit\n");

scanf ("%f",&x0);

printf ("Enter upper limit\n");

scanf ("%f",&x2);

h = (x2-x0)/2;

x1 = x0+h;

f0 = f(x0);

f1 = f(x1);

f2 = f(x2);

i = (f0+4\*f1+f2)\*h/3;

printf("The value of integration =%f\n",i);

getch();

return 0;

}

Output:

Enter lower limit

0

Enter upper limit

2

The value of integration =2.000000

**Lab 25: Write a C program to integrate a given function using Simpson’s 3.8 rule**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define f(x) sqrt(1-x\*x);

int main()

{

float x0,x1,x2,x3,h,I,f0,f1,f2,f3;

printf("Enter upper limit\n");

scanf("%f",&x3);

printf("Enter lower limit\n");

scanf("%f",&x0);

h = (x3-x0)/3;

x1 = x0+h;

x2 = x0+2\*h;

x3 = x0+3\*h;

f0 = f(x0);

f1 = f(x1);

f2 = f(x2);

f3 = f(x3);

I = 3\*h\*(f0+3\*f1+3\*f2+f3)/8;

printf("Integration =%f",I);

getch();

return 0;

}

Output:

Enter upper limit

1

Enter lower limit

0

Integration =0.758062

**Lab 26: Write C Program to calculate integral using Gaussian Integration**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define f(x) x\*x\*x+1

int main()

{

float a,b,z1,z2,c1,c2,x1,x2,v;

printf("Enter lower limit\n");

scanf("%f",&a);

printf("Enter upper limit\n");

scanf("%f",&b);

c1=c2=1;

z1 = -0.57735;

z2 = 0.57735;

x1 = (b-a)/2\*z1+(b+a)/2;

x2 = (b-a)/2\*z2+(b+a)/2;

v = (b-a)/2\*((f(x1))+f(x2));

printf("Value of integration =%f",v);

getch();

return 0;

}

Output:

Enter lower limit

2

Enter upper limit

4

Value of integration =61.999996

**Lab 27: Write a C program to integrate a given function using Romberg Integration**

#include<stdio.h>

#include<conio.h>

#include<math.h>

float f(float x)

{

if(x==0)

return 1.0;

else

return sin(x)/x;

}

int main(){

float x0,xn,t[10][10],h,sm,sl,a;

int i,k,c,r,m,p,q;

printf("Enter lower and upper limit:");

scanf("%f%f",&x0,&xn);

printf("enter p and q required T(p,q):");

scanf("%d%d",&p,&q);

h=xn-x0;

t[0][0]=h/2\*((f(x0))+ (f(xn)));

for(i=1;i<=p;i++){

sl=pow(2,i-1);

sm=0;

for(k=1;k<=sl;k++){

a=x0+(2\*k-1)\*h/pow(2,i);

sm=sm+(f(a));

}

t[i][0]=t[i-1][0]/2+sm\*h/pow(2,i);

}

for(c=1;c<=p;c++){

for(k=1;k<=c && k<=q;k++){

m=c-k;

t[m+k][k]=(pow(4,k)\*t[m+k][k-1]-t[m+k-1][k-1])/(pow(4,k)-1);

}

}

printf("Romberg estimate of integration =%f",t[p][q]);

return 0;

}

**Lab 28: Write a C program to solve system of equations using basic Gauss elimination method**

#include<conio.h>

#include<stdio.h>

#include<math.h>

#define N 10

int main()

{

int n,i,j,k,p,q;

float a[N][N],b[N],x[N];

float pivot, term, sum=0;

printf("Enter Dimension of systems of equations\n");

scanf("%d",&n);

printf("Enter coefficient matrix row wise\n");

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

{

for(j=0;j<n;j++)

{

scanf("%f",&a[i][j]);

}

}

printf("Enter RHS vector\n");

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

scanf("%f",&b[i]);

for(k=0;k<=n-2;k++)

{

pivot = a[k][k];

if(fabs(pivot)<0.000001)

{

printf("Method failed");

}

else

for(i=k+1;i<n;i++)

{

term = a[i][k]/pivot;

for(j=0;j<n;j++)

{

a[i][j] = a[i][j]-a[k][j]\*term;

}

b[i] = b[i]-b[k]\*term;

}

}

x[n-1] = b[n-1]/a[n-1][n-1];

for(i=n-2;i>=0;i--)

{

sum = 0;

for(j=i+1;j<=n-1;j++)

{

sum = sum+a[i][j]\*x[j];

}

x[i] = (b[i]-sum)/a[i][i];

}

printf("The solution vector....\n");

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

{

printf("x%d = %f\n",i+1,x[i]);

}

getch();

return 0;

}

Output:

Enter Dimension of systems of equations

3

Enter coefficient matrix row wise

2 1 1

3 2 3

1 4 9

Enter RHS vector

10 18 16

The solution vector....

x1 = 7.000000

x2 = -9.000000

x3 = 5.000000

**Lab 29:** **Write a** **C program to solve system of linear equations using Jacobi method**

#include<stdio.h>

#include<conio.h>

#include<math.h>

/\* Arranging given system of linear

equations in diagonally dominant

form:

20x + y - 2z = 17

3x + 20y -z = -18

2x - 3y + 20z = 25

\*/

/\* Equations:

x = (17-y+2z)/20

y = (-18-3x+z)/20

z = (25-2x+3y)/20

\*/

/\* Defining function \*/

#define f1(x,y,z) (17-y+2\*z)/20

#define f2(x,y,z) (-18-3\*x+z)/20

#define f3(x,y,z) (25-2\*x+3\*y)/20

int main()

{

float x0=0, y0=0, z0=0, x1, y1, z1, e1, e2, e3, e;

int count=1;

printf("Enter tolerable error:\n");

scanf("%f", &e);

printf("\nCount\tx\ty\tz\n");

do

{

x1 = f1(x0,y0,z0);

y1 = f2(x0,y0,z0);

z1 = f3(x0,y0,z0);

printf("%d\t%0.4f\t%0.4f\t%0.4f\n",count, x1,y1,z1);

e1 = fabs(x0-x1);

e2 = fabs(y0-y1);

e3 = fabs(z0-z1);

count++;

x0 = x1;

y0 = y1;

z0 = z1;

}while(e1>e && e2>e && e3>e);

printf("\nSolution: x=%0.3f, y=%0.3f and z = %0.3f\n",x1,y1,z1);

getch();

return 0;

}

**Lab 30:** **Write a C program to solve ordinary differential equation using Euler’s method**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define f(x,y) 2\*y/x

int main()

{

float x,xp,x0,y1,y,h,y0;

printf("Enter initial values of x and y\n");

scanf("%f%f",&x0,&y0);

printf("Enter x at which function to be evaluated \n");

scanf("%f",&xp);

printf("Enter the step size\n");

scanf("%f",&h);

y = y0;

x= x0;

for(x=x0;x<xp;x = x+h)

{

y = y+f(x,y)\*h;

}

printf("Function value at x = %f is %f\n",xp,y);

getch();

return 0;

}

**Output**

Enter initial values of x and y

1 2

Enter x at which function to be evaluated

2

Enter the step size

0.25

Function value at x = 2.000000 is 7.200000

**Lab 31:** **Write a** C **Program to solve ordinary differential equation using Heun’s Method**

#include<stdio.h>

#include<conio.h>

#define f(x,y) 2\*(y)/(x)

int main()

{

float x, xp, x0,y0, y,h,m1,m2;

printf("Enter initial value of x and y\n");

scanf("%f%f",&x0,&y0);

printf("Enter the value at which function is to be evaluated\n");

scanf("%f",&xp);

printf("Enter step size\n");

scanf("%f",&h);

y = y0;

x = x0;

for(x=x0;x<xp;x=x+h)

{

m1 = f(x,y);

m2 = f(x+h,y+h\*m1);

y = y+h/2\*(m1+m2);

}

printf("Function value at x %f =%f",x,y);

getch();

return 0;

}

Output:

Enter initial value of x and y

1 2

Enter the value at which function is to be evaluated

2

Enter step size

0.25

Function value at x 2.000000 =7.860846

**Lab 32: Write a C program to solve ordinary differential equation using Runge Kutta Method**

#include<stdio.h>

#include<conio.h>

#define f(x,y) 2\*(y)/(x)

int main()

{

float x, xp, x0,y0, y,h,m1,m2,m3,m4;

printf("Enter initial value of x and y\n");

scanf("%f%f",&x0,&y0);

printf("Enter the value at which function is to be evaluated\n");

scanf("%f",&xp);

printf("Enter step size\n");

scanf("%f",&h);

y = y0;

x = x0;

for(x=x0;x<xp;x=x+h)

{

m1 = f(x,y);

m2 = f(x+1/2.0\*h,y+1/2.0\*h\*m1);

m3 = f(x+1/2.0\*h,y+1/2.0\*h\*m2);

m4 = f(x+h,y+h\*m3);

y = y+(m1+2\*m2+2\*m2+m4)\*h/6;

}

printf("Function value at x %f =%f",x,y);

getch();

return 0;

}

Output:

Enter initial value of x and y

1 2

Enter the value at which function is to be evaluated

2

Enter step size

0.25

Function value at x 2.000000 =7.952565

**Lab 33 : Write a C Program for solving system of ODE’s by using Euler’s method**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define f1(x,y,z) z

#define f2(x,y,z) exp(-x)-2\*(z)-(y)

int main()

{

float x, xp, x0,y0,z0,y,z,ny,nz,h;

printf("Enter initial value of x and y and z\n");

scanf("%f%f%f",&x0,&y0,&z0);

printf("Enter the value at which function is to be evaluated\n");

scanf("%f",&xp);

printf("Enter step size\n");

scanf("%f",&h);

y = y0;

x = x0;

z = z0;

for(x=x0;x<xp;x=x+h)

{

ny = y+(f1(x,y,z))\*h;

nz = z+(f2(x,y,z))\*h;

y = ny;

z = nz;

}

printf("Function value at x %f =%f",x,y);

getch();

return 0;

}

Output:

Enter initial value of x and y and z

0 1 2

Enter the value at which function is to be evaluated

0.75

Enter step size

0.25

Function value at x 0.750000 =1.829925

**Lab 34** : **Write a**  **C program to solve second degree ODE using Euler’s method**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define f1(x, y, z) z

#define f2(x, y, z) 6\*(x)+3\*(y)-2\*(z)

int main()

{

float xp, x0,y0,z0,x,y,z,h,m1,m2;

printf("Enter initial value of x and y and z\n");

scanf("%f%f%f",&x0,&y0,&z0);

printf("Enter the value at which function is to be evaluated\n");

scanf("%f",&xp);

printf("Enter step size\n");

scanf("%f",&h);

y = y0;

x = x0;

z = z0;

for(x=x0;x<xp;x=x+h)

{

m1 = f1(x,y,z);

m2 = f2(x,y,z);

y = y+m1\*h;

z = z+m2\*h;

}

printf("Function value x at %f =%f", x, y);

getch();

return 0;

}

Output:

Enter initial value of x and y and z

0 0 1

Enter the value at which function is to be evaluated

0.2

Enter step size

0.1

Function value x at 0.200000 =0.180000