1. **Program to deduce error involved in polynomial equation**

#include<stdio.h>

#include<math.h>

#include<conio.h>

void main()

{

double abs\_err, rel\_err, p\_rel\_err, t\_val, a\_val;

printf(“\n INPUT TRUE VALUE:”);

scanf(“%lf”, &t\_val);

printf(“\n INPUT APPROXIMATE VALUE:”);

scanf(“%lf”, &a\_val);

abs\_err=fabs(t\_val-a\_val);

rel\_err=abs\_err/t\_val;

p\_rel\_err=rel\_err\*100;

printf(“\nABSOLUTE ERROR= %lf”, abs\_err);

printf(“\nRELATIVE ERROR= %lf”, rel\_err);

printf(“\nPERCENTAGE RELATIVE ERROR= %lf”, p\_rel\_err);

getch();

}

1. **PROGRAM: BISECTION METHOD.**

#include<stdio.h>

#include<math.h>

#include<conio.h>

#include<process.h>

#include<string.h>

#define EPS 0.00000005

#define F(x) (x)\*log10(x)-1.2

void Bisect();

int count=1,n;

float root=1;

void main()

{ clrscr();

printf("\n Solution by BISECTION method \n");

printf("\n Equation is ");

printf("\n\t\t\t x\*log(x) - 1.2 = 0\n\n");

printf("Enter the number of iterations:");

scanf("%d",&n);

Bisect();

getch();

}

void Bisect()

{ float x0,x1,x2;

float f0,f1,f2;

int i=0;

for(x2=1;;x2++)

{

f2=F(x2);

if (f2>0)

{ break; }

}

for(x1=x2-1;;x2--)

{

f1=F(x1);

if(f1<0)

{ break; }

}

printf("\t\t-----------------------------------------");

printf("\n\t\t ITERATIONS\t\t ROOTS\n");

printf("\t\t-----------------------------------------");

for(;count<=n;count++)

{

x0=((x1+x2)/2.0);

f0=F(x0);

if(f0==0)

{ root=x0; }

if(f0\*f1<0)

{ x2=x0; }

else

{ x1=x0;

f1=f0;

}

printf("\n\t\t ITERATION %d", count);

printf("\t :\t %f",x0);

if(fabs((x1-x2)/x1) < EPS)

{ printf("\n\t\t---------------------------------");

printf("\n\t\t Root = %f",x0);

printf("\n\t\t Iterations = %d\n", count);

printf("\t\t------------------------------------");

getch();

exit(0);

}

}

printf("\n\t\t----------------------------------------");

printf("\n\t\t\t Root = %7.4f",x0);

printf("\n\t\t\t Iterations = %d\n", count-1);

printf("\t\t------------------------------------------");

getch();

}

1. **PROGRAM: FALSE POSITION or REGULA-FALSI METHOD.**

#include<stdio.h>

#include<math.h>

#include<conio.h>

#include<string.h>

#include<process.h>

#define EPS 0.00005

#define f(x) 3\*x+sin(x)-exp(x)

void FAL\_POS();

void main()

{ clrscr();

printf("\n Solution by FALSE POSITION method\n");

printf("\n Equation is ");

printf("\n\t\t\t 3\*x + sin(x)-exp(x)=0\n\n");

FAL\_POS();

}

void FAL\_POS()

{ float f0,f1,f2;

float x0,x1,x2;

int itr;

int i;

printf("Enter the number of iteration:");

scanf("%d",&itr);

for(x1=0.0;;)

{ f1=f(x1);

if(f1>0)

{ break; }

else

{ x1=x1+0.1; }

}

x0=x1-0.1;

f0=f(x0);

printf("\n\t\t-----------------------------------------");

printf("\n\t\t ITERATION\t x2\t\t F(x)\n");

printf("\t\t--------------------------------------------");

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

{ x2=x0-((x1-x0)/(f1-f0))\*f0;

f2=f(x2);

if(f0\*f2>0)

{ x1=x2;

f1=f2;

}

else { x0=x2;

f0=f2;

}

if(fabs(f(2))>EPS)

{ printf("\n\t\t%d\t%f\t%f\n",i+1,x2,f2); }

}

printf("\t\t--------------------------------------------");

printf("\n\t\t\t\tRoot=%f\n",x2);

printf("\t\t-------------------------------------------");

getch();

}

1. **PROGRAM: NEWTON RAPHSON METHOD.**

# include <stdio.h>

# include <conio.h>

# include <math.h>

# include <process.h>

# include <string.h>

# define f(x) 3\*x -cos(x)-1

# define df(x) 3+sin(x)

void NEW\_RAP();

void main()

{ clrscr();

printf ("\n Solution by NEWTON RAPHSON method \n");

printf ("\n Equation is: ");

printf ("\n\t\t\t 3\*X - COS X - 1=0 \n\n ");

NEW\_RAP();

getch();

}

void NEW\_RAP()

{ long float x1,x0, df0,f0,f1;

int i=1,itr;

float EPS,error;

for(x1=0;;x1 +=0.01)

{ f1=f(x1);

if (f1 > 0)

{ break; }

}

x0=x1-0.01;

f0=f(x0);

printf(" Enter the number of iterations: ");

scanf(" %d",&itr);

printf(" Enter the maximum possible error: ");

scanf("%f",&EPS);

if (fabs(f0) > f1)

{ printf("\n\t\t The root is near to %.4f\n",x1); }

if(f1 > fabs(f(x0)))

{ printf("\n\t\t The root is near to %.4f\n",x0); }

x0=(x0+x1)/2;

for(;i<=itr;i++)

{ f0=f(x0);

df0=df(x0);

x1=x0 - (f0/df0);

printf("\n\t\t The %d approximation to the root is:%f",i,x1);

error=fabs(x1-x0);

if(error<EPS)

{ break; }

x0 = x1;

}

if(error>EPS)

{ printf("\n\n\t NOTE:- ");

printf("The number of iterations are not sufficient.");

}

printf("\n\n\n\t\t\t ------------------------------");

printf("\n\t\t\t The root is %.4f ",x1);

printf("\n\t\t\t ------------------------------"); }

1. **PROGRAM: ITERATION METHOD.**

#include<stdio.h>

#include<math.h>

#include<conio.h>

#define EPS 0.00005

#define F(x) (x\*x\*x + 1)/2

#define f(x) x\*x\*x - 2\*x + 1

void ITER();

void main ()

{

clrscr();

printf("\n\t Solution by ITERATION method - ");

printf("\n\t Equation is - ");

printf("\n\t\t\t\t X\*X\*X - 2\*X + 1 = 0\n");

ITER();

getch();

}

void ITER()

{

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

int i=0,n;

for(x1=1;;x1++)

{ f1=F(x1);

if (f1>0)

break;

}

for(x0=x1-1;;x0--)

{ f0=f(x0);

if(f0<0)

break;

}

x2=(x0+x1)/2;

printf("Enter the number of iterations:- ");

scanf("%d",&n);

printf("\n\t\t The 1 approximation to the root is:- %f",x2);

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

{ f2=F(x2);

printf("\n\t\t The %d approximation to the root is:- %f",i+2,f2);

x2=F(x2);

error=fabs(f2-f1);

if(error<EPS)

break;

f1=f2;

}

if(error>EPS)

printf("\n\n\t NOTE:- The number of iterations are not sufficient.");

printf("\n\n\n\t\t\t------------------------------");

printf("\n\t\t\t The root is %.4f",f2);

printf("\n\t\t\t-----------------------------");

}

1. **PROGRAM: TRAPEZOIDAL METHOD.**

# include <stdio.h>

# include <math.h>

# include <string.h>

float fun(float);

void main()

{

float result=1, a,b, h,sum;

int i,j,n; clrscr();

printf("\n\n Enter the range - ");

printf("\n\n Lower Limit a - ");

scanf("%f" ,&a);

printf("\n\n Upper Limit b - ");

scanf("%f" ,&b);

printf("\n\n Enter number of subintervals - ");

scanf("%d" ,&n);

h=(b-a)/n;

sum=fun(a)+fun(b);

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

{ sum+=2\*fun(a+i); }

result=sum\*h/2;

printf("n\n\n\n Value of the integral is %6.4f\t",result);

getch();

}

float fun(float x)

{ float temp;

temp = 1/(1+(x\*x));

return temp;

}

1. **PROGRAM: SIMPSON’S 1/3rd METHOD OF NUMERICAL INTEGRATION**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<string.h>

float fun(float);

void main()

{

float result=1,a,b,sum,h;

int i,j,n;

printf("\n Enter the range - ");

printf("\n Lower Limit a - ");

scanf("%f",&a);

printf("\n Upper limit b - ");

scanf("%f",&b);

printf("\n\n Enter number of subintervals - ");

scanf("%d",&n);

h=(b-a)/n;

sum=fun(a)+4\*fun(a+h)+fun(b);

for(i=3;i<n;i+=2)

{ sum+=2\*fun(a+(i-1)\*h)+4\*fun(a+i\*h); }

result=sum\*h/3;

printf("\n\nValue of integral is %6.4f\t",result);

getch();}

float fun(float x)

{ float temp;

temp=1/(1+(x\*x));

return temp;

}

1. **PROGRAM: SIMPSON’S 3/8th METHOD OF NUMERICAL INTEGRATION**

#include<stdio.h>

#include<conio.h>

float fun(int);

void main()

{ int n,a,b,i;

float h, sum=0, result;

printf("enter range");

scanf("%d",&n);

printf("enter lower limit");

scanf("%d",&a);

printf("enter upper limit");

scanf("%d",&b);

h=(b-a)/n;

sum=fun(a)+fun(b);

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

{ if (i%2==0)

sum+=2\*fun(a+i\*h);

else

sum+=3\*fun(a+i\*h);

}

result=sum\*3/8\*h;

printf("%f", result);

getch();

}

float fun(int x)

{ float val;

val=1/(1+(x\*x));

return(val);

}

1. **PROGRAM: LAGRANGE’S INTERPOLATION FORMULA.**

#include<stdio.h>

#define MAX 10

void main()

{

float x[MAX],y[MAX],k=0,z,nr,dr; int i,j,m;

printf("\n enter the range ");

scanf("%d",&m);

printf("\n enter the x value ");

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

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

printf("\n enter the y value ");

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

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

printf("\n enter value OF Z to be calculated ");

scanf("%f",&z);

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

{ nr=1;dr=1;

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

{ if (j!=i)

{ nr=nr\*(z-x[j]);

dr=dr\*(x[i]-x[j]);

}

}

k=k+((nr/dr)\*y[i]);

}

printf("\n final result=%f\n",k);

getch();

}

1. **Program to implement Newton’s Divided Difference formula.**

#include<stdio.h>

#include<math.h>

void main()

{

float x[10],y[10][10],sum,p,u,temp;

int i,n,j,k=0,f,m;

float fact(int);

printf("\nhow many record you will be enter: ");

scanf("%d",&n);

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

{ printf("\n\nenter the value of x%d: ",i);

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

printf("\n\nenter the value of f(x%d): ",i);

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

}

printf("\n\nEnter X for finding f(x): ");

scanf("%f",&p);

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

{ k=i;

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

{ y[i][j]=(y[i-1][j+1]-y[i-1][j])/(x[k]-x[j]);

k++;

}

}

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("\n x(i)\t y(i)\t y1(i) y2(i) y3(i) y4(i)");

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

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

{ printf("\n %.3f",x[i]);

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

{ printf(" ");

printf(" %.3f",y[j][i]);

}

printf("\n");

}

i=0;

do { if(x[i]<p && p<x[i+1])

k=1;

else

i++;

}while(k != 1);

f=i;

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

{ k=f;

temp=1;

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

{ temp = temp \* (p - x[k]);

k++;

}

sum = sum + temp\*(y[i][f]);

}

printf("\n\n f(%.2f) = %f ",p,sum);

getch();

}

1. **PROGRAM: STERLING’S METHOD OF INTERPOLATION.**

#include<stdio.h>

#include<math.h>

#include<process.h>

void main()

{

int n,I,j;

float ax[10],h,p,x,y ,y1,y2,y3,y4;;

float diff[20][20];

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

scanf("%d",%n);

printf(”\n Enter the values for x \n”);

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

{ printf("\n Enter the value for x%d-",i+1);

scanf("%f”,&ax[i]);

}

printf("\n Enter the values for y \n");

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

{ printf("\n Enter the value for y%d-",i+1);

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

}

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

printf("\n which you want the value of y");

scanf("%f",&x);

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

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

{ diff[i][1]=ay[i+1]-ay[i]; }

for(j=2;j<=4;j++)

{ for(i=0;i<n-j;i++)

{ diff[i][j]=diff[i+1][j-1]-diff[i][j-1]; }

}

i=0;

do { i++;

}while(ax[i]<x);

i--;

p=(x-ax[i])/h;

y1=p\*(diff[i][1]+diff[i-1][1])/2;

y2=p\*p\*diff[i-1][2]/2;

y3=p\*(p\*p-1)\*(diff[i-1][3]+diff[i-2][3])/6;

y4=p\*p\*(p\*p-1)\*diff[i-2][4]/24;

y=ay[i]+y1+y2+y3+y4;

printf("\n\n When x=%6.2f, y=%6.8f",x,y);

getch();

}

1. **PROGRAM: NEWTON’S FORWORD METHOD OF INTERPOLATION**

# include <stdio.h>

# include <math.h>

# include <string.h>

void main()

{

int n,I,j;

float ax[10], ay[10],x,y=0,h,p;

float diff[20][20] ,y1,y2,y3,y4;

printf("\n Enter the number of terms - ");

scanf("%d",&n);

printf("Enter the value in the form of x - ");

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

{ printf("Enter the value of x%d - ",i+1);

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

}

printf("\n Enter the value in the form of y - ");

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

{ printf ("Enter the value of y%d - ", i+1);

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

}

printf("\nEnter the value of x for");

printf("\nwhich you want the value of y - ");

scanf("%f",&x);

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

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

{ diff[i][1]=ay[i+1]-ay[i]; }

for(j=2;j<=4;j++)

{ for(i=0;i<n-j;i++)

{ diff[i][j]=diff[i+1][j-1]-diff[i][j-1]; }

}

i=0;

do

{ i++;

}while(ax[i]<x);

i--;

p=(x-ax[i])/h;

y1=p\*diff[i-1][1];

y2=p\*(p+1)\*diff[i-1][2]/2;

y3=(p+1)\*p\*(p-1)\*diff[i-2][3]/6;

y4=(p+2)\*(p+1)\*p\*(p-1)\*diff[i-3][4]/24;

y=ay[i]+y1+y2+y3+y4;

printf("\nwhen x=%6.4f, y=%6.8f ",x,y); }

1. **PROGRAM: NEWOTN’S BACKWARD METHOD OF INTERPOLATION.**

#include<stdio.h>

#include<math.h>

#include<string.h>

void main()

{

int n,i,j,k;

float mx[10],my[10],x,x0=0,y0,sum,h,fun,p,diff[20][20],y1,y2,y3,y4;

printf("\n enter the no. of terms - ");

scanf("%d",&n);

printf("\n enter the value in the form of x - ");

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

{

printf("\n enter the value of x%d- ",i+1);

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

}

printf("\n enter the value in the form of y - ");

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

{

printf("\n\n enter the value of y%d- ",i+1);

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

}

printf("\n enter the value of x for");

printf("\nwhich you want the value of of y -");

scanf("%f",&x);h=mx[1]-mx[0];

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

{ diff[i][1]=my[i+1]-my[i]; }

for(j=2;j<=4;j++)

{ for(i=0;i<n-j;i++)

{ diff[i][j]=diff[i+1][j-1]-diff[i][j-1]; }

}

i=0;

while(!mx[i]>x)

{ i++; }

x0=mx[i];

sum=0;

y0=my[i];

fun=1;

p=(x-x0)/h;

sum=y0;

for(k=1;k<=4;k++)

{

fun=(fun\*(p-(k-1))/k);

sum=sum+fun\*diff[i][k];}

printf("\n when x=%6.4f,y=%6.8f",x,sum);

printf("\n press enter to exit");

getch();

}

1. **PROGRAM: GAUSS’S FORWORD METHOD OF INTERPOLATION.**

# include <stdio.h>

# include <math.h>

# include <string.h>

void main()

{

int n,I,j;

float ax[10],ay[10],x,nr,dr,y=0,h,p,diff[20][20],y1,y2,y3,y4;

printf(" Enter the number of terms - ");

scanf("%d",&n);

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

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

{

printf(" Enter the value of x%d - ",i+1);

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

}

printf(" Enter the value in the form of y - ");

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

{

printf("Enter the value of y%d - ",i+1);

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

}

printf("\nEnter the value of x for - ");

printf("\nwhich you want the value of y - ");

scanf ("%f",&x);

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

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

{ diff[i][1]=ay[i+1]-ay[i]; }

for(j=2;j<=4;j++)

{

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

{ diff[i][j]=diff[i+1][j-1]-diff[i][j-1]; }

}

i=0;

do {

i++;

}while(ax[i]<x);

i--;

p=(x-ax[i])/h;

y1=p\*diff[i][1];

y2=p\*(p-1)\*diff[i-1][2]/2;

y3=(p+1)\*p\*(p-1)\*diff[i-2][3]/6;

y4=(p+1)\*p\*(p-1)\*(p-2)\*diff[i-3][4]/24;

y=ay[i]+y1+y2+y3+y4;

printf("\nwhen x=%6.4f,y=%6.8f ",x,y);

getch();

}