

**Tribhuvan University**

**GREENFIELD NATIONAL COLLEGE**

**Bafal- Sitapaila, Kathmandu**

**Lab Report**

**of**

**“Numerical Methods”**

**Submitted By: Submitted To:**

Rabindra Pangeni

Lecturer/ Faculty Member

BCA Department

Name: Manish karki

Semester: IV

Program: Bachelor of Computer Application

TU Reg. No: 6-2-717-6-2022

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.N** | **Title** | **Signature** |
| 1 | C program for Bisection method. |  |
| 2 | C program for Newton Raphson method. |  |
| 3 | C program for Secant method. |  |
| 4 | C program for Fixed Point method. |  |
| 5 | C program for Lagrange Interpolation Method |  |
| 6 | C program for Newton’s Forward Difference Polynomial. |  |
| 7 | C program for Newton’s Backward Difference Interpolation. |  |
| 8 | C program for Linear Regression. |  |
| 9 | C program for Exponential Regression Model(using linearization). |  |
| 10 | C program for Naive Gauss Elimination method. |  |
| 11 | C program for Jacobi Iteration method. |  |
| 12 | C program for Gauss-Seidel method. |  |
| 13 | C program for Solving Linear Equations using Doolittle LU Decomposition. |  |
| 14 | C program for calculating Eigenvalue and Eigenvector of a matrix. |  |
| 15 | C program for calculating derivative using Forward Difference formula. |  |
| 16 | C program for calculating Maxima and Minima of tabulated functions. |  |
| 17 | C program for Trapezoidal method. |  |
| 18 | C program for computing integral value by using Simpson’s 1/3 rule. |  |
| 19 | C program for computing integral value by using Simpson’s 3/8 rule |  |
| 20 | C program to estimate Romberg estimate T(p,q) of integration. |  |
| 21 | C program to solve ODE by using Taylor’s series method. |  |
| 22 | C program to solve ODE by using Euler’s method. |  |
| 23 | C program for solving elliptic PDE’s by using finite difference method. |  |

# 1. Write a program to implement bisection method for solving non-linear equations.

Source Code:

#include<stdio.h>

#include<math.h>

#define f(x) cos(x) - x \* exp(x)

void main()

{

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

int step = 1;

up:

printf("\nEnter two initial guesses:\n");

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

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

scanf("%f", &e);

/\* Calculating Functional Value \*/

f0 = f(x0);

f1 = f(x1);

if( f0 \* f1 > 0.0){

printf("Incorrect Initial Guesses.\n");

goto up;

}

printf("\nStep\t\tx0\t\tx1\t\tx2\t\tf(x2)\n");

do{

x2 = (x0 + x1)/2;

f2 = f(x2);

printf("%d\t\t%f\t%f\t%f\t%f\n",step, x0, x1, x2, f2);

if( f0 \* f2 < 0){

x1 = x2;

f1 = f2;

}

else{

x0 = x2;

f0 = f2;

}

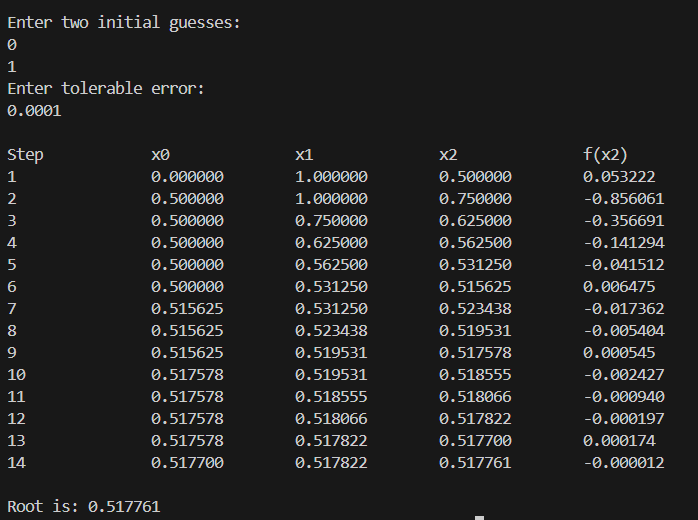
step = step + 1;

}while(fabs(f2)>e);

printf("\nRoot is: %f", x2);

}

OUTPUT



**2.** **Write a program to implement newton raphson method for solving linear**  **equations.**

Source Code:

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

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

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

void main()

{

float x0, x1, f0, f1, g0, e;

int step = 1, N;

printf("\nEnter initial guess:\n");

scanf("%f", &x0);

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

scanf("%f", &e);

printf("Enter maximum iteration:\n");

scanf("%d", &N);

printf("\nStep\t\tx0\t\tf(x0)\t\tx1\t\tf(x1)\n");

do

{

g0 = g(x0);

f0 = f(x0);

if (g0 == 0.0)

{

printf("Mathematical Error.");

exit(0);

}

x1 = x0 - f0 / g0;

printf("%d\t\t%f\t%f\t%f\t%f\n", step, x0, f0, x1,

f1);

x0 = x1;

step = step + 1;

if (step > N)

{

printf("Not Convergent.");

exit(0);

}

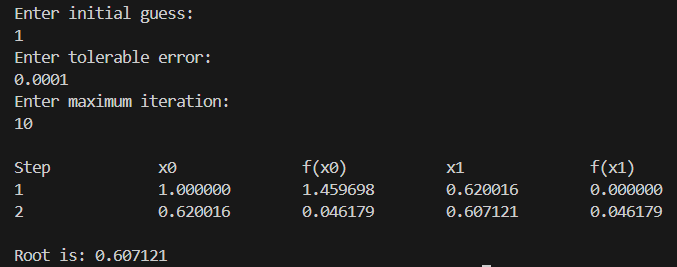
f1 = f(x1);

} while (fabs(f1) > e);

printf("\nRoot is: %f", x1);

}

OUTPUT



**3. Write a program to implement secant method for solving non-linear equations.**

**Source Code:**

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

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

void main()

{

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

int step = 1, N;

printf("\nEnter initial guesses:\n");

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

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

scanf("%f", &e);

printf("Enter maximum iteration:\n");

scanf("%d", &N);

printf("\nStep\t\tx0\t\tx1\t\tx2\t\tf(x2)\n");

do{

f0 = f(x0);

f1 = f(x1);

if(f0 == f1){

printf("Mathematical Error.");

exit(0);

}

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

f2 = f(x2);

printf("%d\t\t%f\t%f\t%f\t%f\n",step,x0,x1,x2, f2);

x0 = x1;

f0 = f1;

x1 = x2;

f1 = f2;

step = step + 1;

if(step > N){

printf("Not Convergent.");

exit(0);

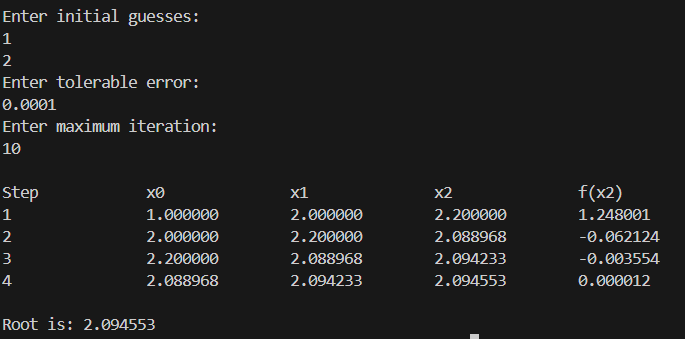
}

}while(fabs(f2)>e);

printf("\nRoot is: %f", x2);

}

OUTPUT

****

**4. Write a program to implement fixed point method for solving non-linear equation.**

**Source Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

#define G(x) (a3 \* x \* x \* x + a2 \* x \* x + a0) / (-a1)

float a0, a1, a2, a3;

int main()

{

float x0, x1, E, Er;

printf("Enter coeffients a3,a2,a1 and a0");

scanf("%f%f%f%f", &a3, &a2, &a1, &a0);

printf("Enter initial guesss and E\n");

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

while (1)

{

x1 = G(x0);

Er = (x1 - x0) / x1;

if (fabs(Er) < E)

{

printf("Root=%f\n", x1);

break;

}

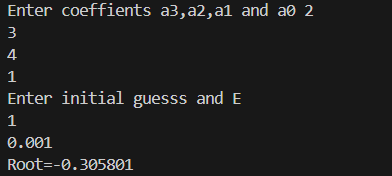
x0 = x1;

}

getch();

}

Output:



**5. Write a program to implement the Lagrange Interpolation Method.**

**Source Code:**

#include<stdio.h>

#include<conio.h>

void 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]);

}

}

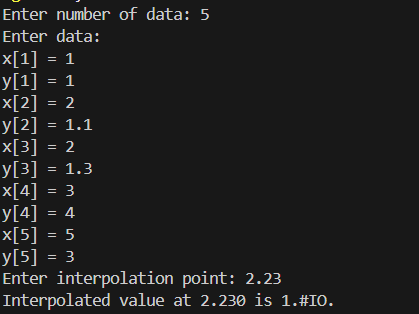
yp = yp + p \* y[i];

}

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

}

OUTPUT



**6. Write a program to implement the Newton’s Forward Difference polynomial**.

**Source Code:**

#include <stdio.h>

float v = 0, p, xp;

int n, i;

float x[10], fx[10], h, s;

float fd[10];

int factorial(int num);

int main()

{

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

scanf("%d", &n);

printf("Enter the value at which interpolated value is to be calculated\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 (int j = 1; j < n; j++)

{

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

{

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

}

}

v = fd[0];

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

{

p = 1;

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

{

p \*= (s - k);

}

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

}

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

return 0;

}

int factorial(int num)

{

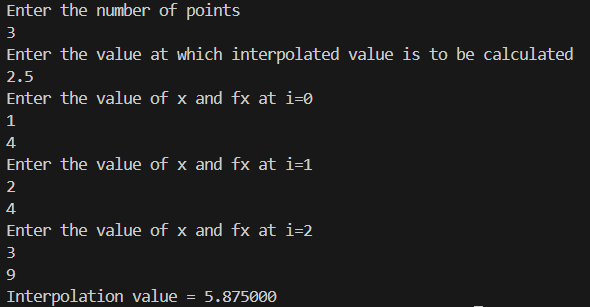
if (num <= 1)

return 1;

return num \* factorial(num - 1);

}

Output:



**7. Write a program to implement the Newton’s Backward Difference Interpolation.**

**Source Code:**

#include <stdio.h>

// Function prototype for factorial

int factorial(int num);

int main()

{

float v = 0, p, xp;

int n, i;

float x[10], fx[10], h, s;

float fd[10];

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

scanf("%d", &n);

printf("Enter the value at which interpolated value is to be calculated\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 (int j = 1; j < n; j++)

{

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

{

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

}

}

v = fd[0];

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

{

p = 1;

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

{

p \*= (s - k);

}

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

}

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

return 0;

}

int factorial(int num)

{

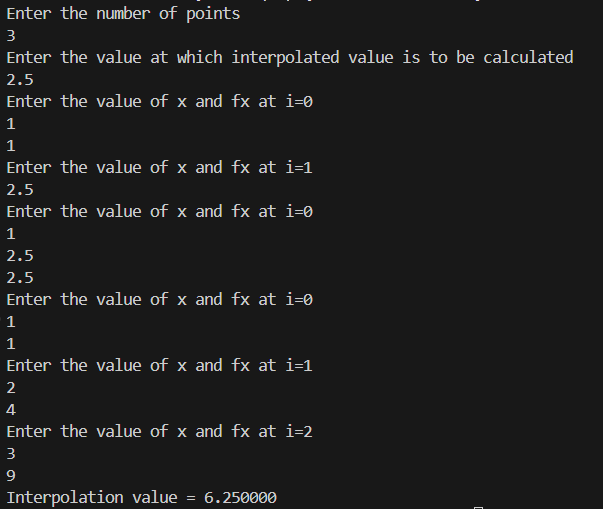
if (num <= 1)

return 1;

return num \* factorial(num - 1);

}

Output:



**8. Write a program to implement the linear regression.**

**Source Code:**

#include<stdio.h>

#include<conio.h>

int main()

{

int n,i,j,k;

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

printf("Enter the 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];

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));

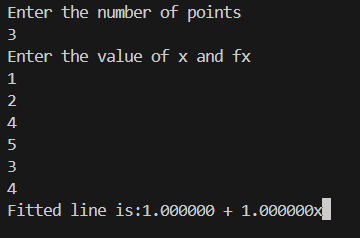
a=(sy/n)-(b\*sx/n);

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

getch();

}

**Output:**

****

**9. Write a program to implement the exponential regression model (using linearization).**

**Source Code:**

#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 = 0, slgy = 0, sxy = 0, sx2 = 0;

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

scanf("%d", &n);

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

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

{

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

}

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

{

sx = sx + x[i];

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

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

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

}

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

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

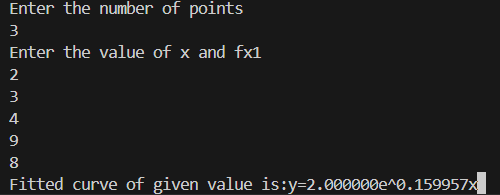
a = exp(r);

printf("Fitted curve of given value is:y=%fe^%fx", a, b);

getch();

}

**Output:**

****

**10. Write a program to implement the Naive Gauss Elimination method.**

**Source Code:**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<stdlib.h>

#define SIZE 10

int main()

{

float a[SIZE][SIZE], x[SIZE], ratio;

int i,j,k,n;

printf("Enter number of variables: ");

scanf("%d", &n);

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

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

{

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

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

}

}

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

if(a[i][i] == 0.0){

printf("Mathematical Error!");

exit(0);

}

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

ratio = a[j][i]/a[i][i];

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

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

}}

}

x[n] = a[n][n+1]/a[n][n];

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

x[i] = a[i][n+1];

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

x[i] = x[i] - a[i][j]\*x[j];

}

x[i] = x[i]/a[i][i];

}

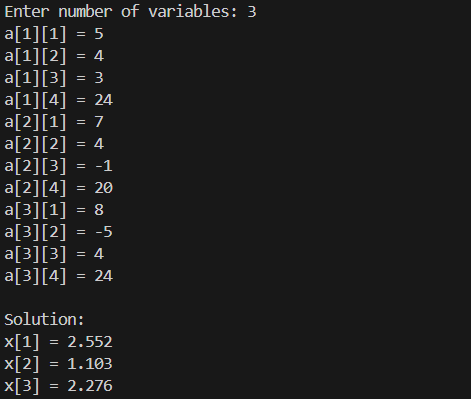
printf("\nSolution:\n");

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

printf("x[%d] = %0.3f\n",i, x[i]); }

return(0);}

OUTPUT



**11. Write a program to implement the Jacobi Iteration method.**

**Source Code:**

#include <stdio.h>

#include <math.h>

#define f1(x, y, z) (-1 + 2 \* y - 3 \* z) / 5

#define f2(x, y, z) (2 + 3 \* x - z) / 9

#define f3(x, y, z) (-3 + 2 \* x - y) / 7

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;

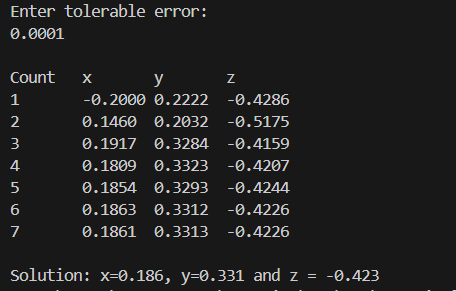
} while (e1 > e && e2 > e && e3 > e);

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

return 0;

}

OUTPUT



**12. Write a program to implement the Gauss-Seidel method.**

**Source Code:**

#include <stdio.h>

#include <math.h>

#define f1(x, y, z) (-1 + 2 \* y - 3 \* z) / 5

#define f2(x, y, z) (2 + 3 \* x - z) / 9

#define f3(x, y, z) (-3 + 2 \* x - y) / 7

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(x1, y0, z0);

z1 = f3(x1, y1, 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;

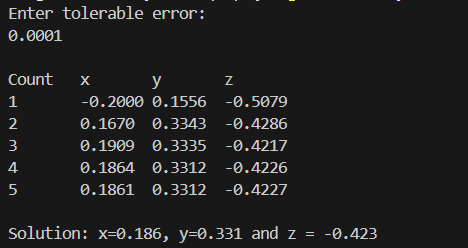
} while (e1 > e && e2 > e && e3 > e);

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

return 0;

}

**OUTPUT:**



**13. Write a program for solving system of linear equation using Doolittle LU Decomposition.**

**Source Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

int main()

{

int n, i, j, k;

float sum = 0, a[10][10], b[10], x[10], z[10], u[10][10], l[10][10];

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

scanf("%d", &n);

printf("Enter coefficients Matrix\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 (j = 0; j < n; j++)

u[0][j] = a[0][j];

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

l[i][i] = 1;

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

l[i][0] = a[i][0] / u[0][0];

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

{

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

{

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

{

sum = sum + (l[i][k] \* u[k][j]);

}

u[i][j] = a[i][j] - sum;

sum = 0;

}

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

{

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

{

sum = sum + (l[j][k] \* u[k][j]);

}

l[i][j] = (a[i][j] - sum) / u[j][j];

sum = 0;

}

}

z[0] = b[0];

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

{

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

sum = sum + (l[i][j] \* z[j]);

z[i] = b[i] - sum;

sum = 0;

}

x[n - 1] = z[n - 1] / u[n - 1][n - 1];

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

{

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

sum = sum + (u[i][j] \* x[j]);

x[i] = (z[i] - sum) / u[i][i];

sum = 0;

}

printf("Solution:\n");

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

{

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

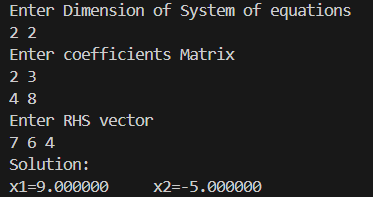
}

getch();

return 0;

}

**Output:**



**14. Write a program for calculating Eigenvalue and Eigenvector of a matrix.**

**Source Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

int main()

{

int n, i, j;

float el, k, E[10], a[10][10], nx[10], x[10], y[10];

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

scanf("%d", &n);

printf("Enter coefficients row-wise\n");

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

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

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

printf("Enter guess vector\n");

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

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

printf("Enter Accuracy Limit\n");

scanf("%f", &el);

while (1)

{

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

y[i] = a[i][0] \* x[0] + a[i][1] \* x[1] + a[i][2] \* x[2];

k = y[0];

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

if (k < y[i])

k = y[i];

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

nx[i] = 1 / k \* y[i];

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

{

E[i] = (nx[i] - x[i]) / nx[i];

if (E[i] > el)

break;

}

if (i == n)

{

printf("Largest Eigenvalue is:%f\n", k);

printf("Eigenvector is:\n");

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

printf("%f\t", nx[i]);

break;

}

else

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

x[i] = nx[i];

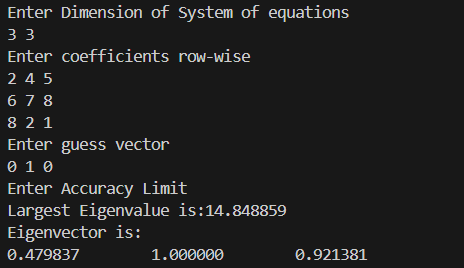
}

getch();

return 0;

}

**Output:**

****

**15. Write a program for calculating derivative using forward difference formula.**

**Source Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

#define PI 3.1415

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

int main()

{

float angle, h, x, d;

printf("Enter Angle in Degree:\n");

scanf("%f", &angle);

printf("Enter increment h:\n");

scanf("%f", &h);

x = PI / 180 \* angle;

d = ((f(x + h)) - (f(x))) / h;

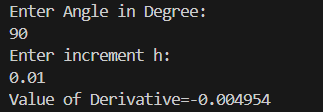
printf("Value of Derivative=%f\n", d);

getch();

return 0;

}

**Output:**

****

**16. Write a program for calculating maxima and minima of tabulated functions.**

**Source Code:**

#include <stdio.h>

#include <conio.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 points\n");

scanf("%d", &n);

printf("Enter values of x & f(x)\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

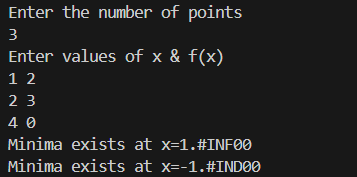
printf("Minima exists at x=%f\n", x2);

getch();

return 0;

}

**Output:**

****

**17. Write a program implementing the Trapezoidal method.**

**Source Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

#define f(x) 1 / (1 + pow(x, 2))

int main()

{

float lower, upper, integration = 0.0, stepSize, k;

int i, subInterval;

printf("Enter lower limit of integration: ");

scanf("%f", &lower);

printf("Enter upper limit of integration: ");

scanf("%f", &upper);

printf("Enter number of sub intervals: ");

scanf("%d", &subInterval);

stepSize = (upper - lower) / subInterval;

integration = f(lower) + f(upper);

for (i = 1; i <= subInterval - 1; i++)

{

k = lower + i \* stepSize;

integration = integration + 2 \* f(k);

}

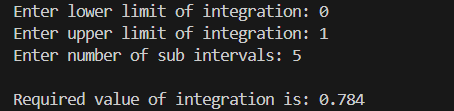
integration = integration \* stepSize / 2;

printf("\nRequired value of integration is: %.3f", integration);

return 0;

}

# Output:



**18. Write a program for computing integral value by using Simpson’s 1/3 rule.**

**Source Code:**

#include<stdio.h>

#include<conio.h>

#include<math.h>

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

int main()

{

float lower, upper, integration=0.0, stepSize, k;

int i, subInterval;

printf("Enter lower limit of integration: ");

scanf("%f", &lower);

printf("Enter upper limit of integration: ");

scanf("%f", &upper);

printf("Enter number of sub intervals: ");

scanf("%d", &subInterval);

stepSize = (upper - lower)/subInterval;

integration = f(lower) + f(upper);

for(i=1; i<= subInterval-1; i++)

{

k = lower + i\*stepSize;

if(i%2==0){

integration = integration + 2 \* f(k);

}

else{

integration = integration + 4 \* f(k);

}

}

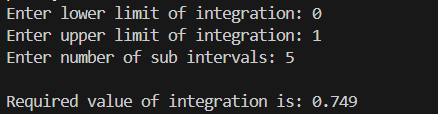
integration = integration \* stepSize/3;

printf("\nRequired value of integration is: %.3f", integration);

return 0;

}

**Output:**



**19. Write a program for computing integral value by using Simpson’s 3/8 rule.**

**Source Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

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

int main()

{

float h, x0, x1, x2, x3, fx0, fx1, fx2, fx3, v;

int n = 3;

printf("Enter Lower Limit\n");

scanf("%f", &x0);

printf("Enter Upper Limit\n");

scanf("%f", &x3);

h = (x3 - x0) / n;

x1 = x0 + h;

x2 = x0 + 2 \* h;

fx0 = f(x0);

fx1 = f(x1);

fx2 = f(x2);

fx3 = f(x3);

v = 3 / 8.0 \* h \* (fx0 + 3 \* fx1 + 3 \* fx2 + fx3);

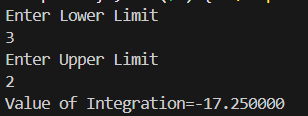
printf("Value of Integration=%f\n", v);

getch();

return 0;

}

**Output:**

****

**20. Write a program to estimate Romberg estimate T(p,q) of integration.**

**Source Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

#define f(x) 1 / (x)

int main()

{

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

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

printf("Enter Lower & Upper Limit\n");

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

printf("Enter p & q of required T(p,q)\n");

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);

}

}

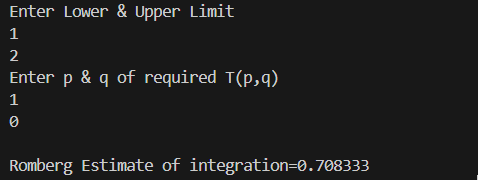
printf("\nRomberg Estimate of integration=%f\n", T[p][q]);

getch();

return 0;

}

**Output:**

****

**21. Write a program to solve ODE by using Taylor’s series method.**

**Source Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

int fact(int n)

{

if (n == 1)

return 1;

else

return (n \* fact(n - 1));

}

int main()

{

float x, x0, yx0, yx, fdy, sdy, tdy;

printf("Enter initial values if x & y be:\n");

scanf("%f%f", &x0, &yx0);

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

scanf("%f", &x);

fdy = (x0) \* (x0) + (yx0) \* (yx0);

sdy = 2 \* (x0) + 2 \* (yx0)\*fdy;

tdy = 2 + 2 \* yx0 \* sdy + 2 \* fdy \* fdy;

yx = yx0 + (x - x0) \* fdy + (x - x0) \* (x - x0) \* sdy / fact(2) + (x - x0) \* (x - x0) \* (x - x0) \* tdy / fact(3);

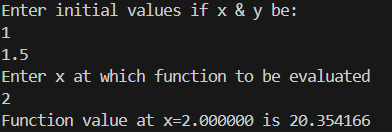
printf("Function value at x=%f is %f\n", x, yx);

getch();

return 0;

}

**Output:**

****

**22. Write a program to solve ODE by using Euler’s method.**

**Source Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

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

int main()

{

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

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

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

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

scanf("%f", &h);

y = y0;

x = x0;

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

{

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

}

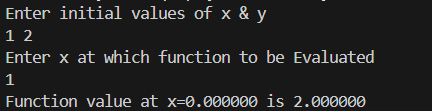
printf("Function value at x=%f is %f\n", xp, y);

getch();

return 0;

}

**Output:**

****

**23. Write a program for solving elliptic PDE’s by using finite difference method.**

**// Assumption: T11=x1  T12=x2 T22=x3 T23=x4**

**Source Code:**

#include<stdio.h>

#include<conio.h>

#include<math.h>

int main()

{

int n,i,j,k;

float sum,error, E[10],a[10][10],b[10],new\_x[10],old\_x[10],tl,tr,tu,tb;

printf("Enter Dimension of plate\n");

scanf("%d",&n);

printf("Enter temperatures at left, right, bottom & upper part of plate\n");

scanf("%f%f%f%f",&tl,&tr,&tb,&tu);

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

a[i][i]=-4;

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

a[i][n-i]=0;

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

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

{

if(i!=j && j!=(n-1))

a[i][j]=1;

}

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

b[i]=0;

k=0;

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

{

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

{

if((i-1)==0)

b[k]=b[k]-tl;

if((i+1)==n)

b[k]=b[k]-tr;

if((j-1)==0)

b[k]=b[k]-tb;

if((j+1)==n)

b[k]=b[k]-tu;

k++;

}

}

printf("Enter Accuracy Limit\n");

scanf("%f",&error);

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

{

new\_x[i]=0;

}

while(1)

{

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

{

sum=b[i];

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

{

if(i!=j)

sum=sum-a[i][j]\*new\_x[j];

}

old\_x[i]=new\_x[i];

new\_x[i]=sum/a[i][i];

E[i]=fabs(new\_x[i]-old\_x[i])/fabs(new\_x[i]);

}

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

{

if(E[i]>error)

break;

}

if(i==(n+1))

break;

else

continue;

}

printf("Solution:\n");

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

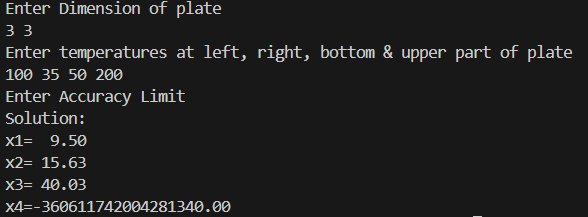
printf("x%d=%6.2f\n",i+1,new\_x[i]);

getch();

return 0;

}

**Output:**

****