Contents

Lab 1. solution of non linear equation

1. Bisection method

```
#include<stdio.h>
#include<conio.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);
        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);
getch();
```

output:

```
Enter two initial guesses:
0
1
Enter tolerable error:
0.001

Step            x0              x1              x2              f(x2)
1               0.000000        1.000000        0.500000        0.053222
2               0.500000        1.000000        0.750000        -0.856061
3               0.500000        0.750000        0.625000        -0.356691
4               0.500000        0.625000        0.562500        -0.141294
5               0.500000        0.562500        0.531250        -0.041512
6               0.500000        0.531250        0.515625        0.006475
7               0.515625        0.531250        0.523438        -0.017362
8               0.515625        0.523438        0.519531        -0.005404
9               0.515625        0.519531        0.517578        0.000545

Root is: 0.517578
```

2. Netwon Raphson Method

```
#include<conio.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);
      getch();
}
```

Output:

```
Enter initial guess:
0
Enter tolerable error:
0.0001
Enter maximum iteration:
10

Step            x0              f(x0)           x1              f(x1)
1               0.000000        -2.000000       0.666667        0.000000
2               0.666667        0.214113        0.607493        0.214113
3               0.607493        0.001397        0.607102        0.001397

Root is: 0.607102
```

3. Secant method

```
#include<stdio.h>
#include<conio.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);
    getch();
```

output:

```
Enter initial guesses:
1
2
Enter tolerable error:
0.00001
Enter maximum iteration:
10

Step            x0              x1              x2              f(x2)
1               1.000000        2.000000        2.200000        1.248001
2               2.000000        2.200000        2.088968        -0.062123
3               2.200000        2.088968        2.094233        -0.003557
4               2.088968        2.094233        2.094553        0.000011
5               2.094233        2.094553        2.094552        0.000001

Root is: 2.094552
```

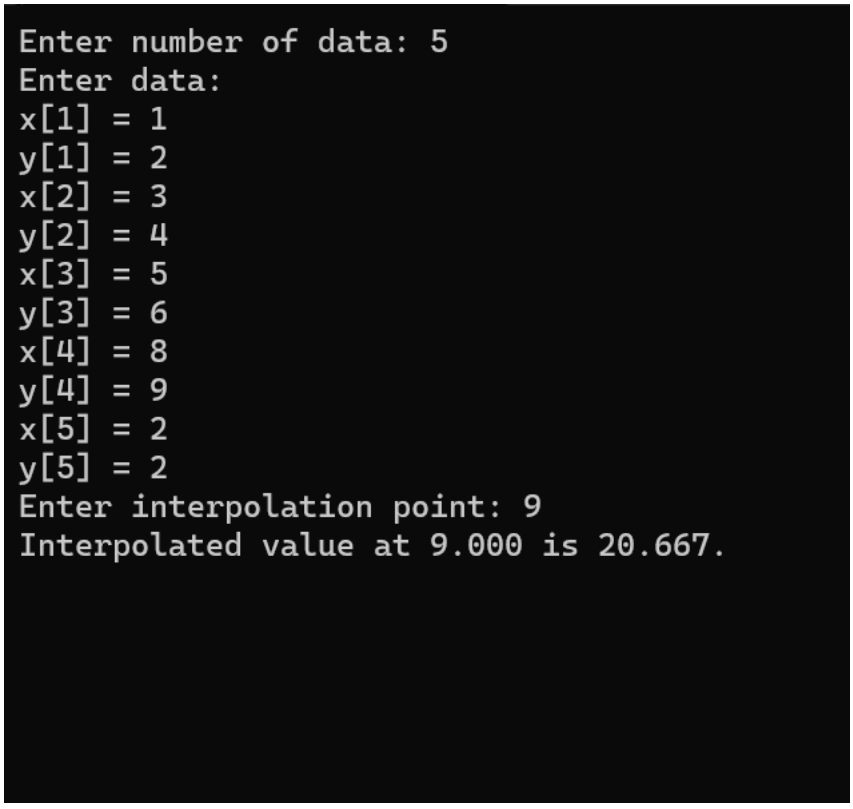
Lab2: interpolation and approximation

1. Lagrange interpolation

```
#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);
        getch();
}
```

Output:



2. Newton interpolation using forward method

```
#include<stdio.h>
#include<conio.h>
int main()
{
 float x[20], y[20][20];
 int i,j, n;
 /* Input Section */
 printf("Enter number of data?\n");
 scanf("%d", &n);
 printf("Enter data:\n");
 for(i = 0; i < n ; i++)
 {
  printf("x[%d]=", i);
  scanf("%f", &x[i]);
  printf("y[%d]=", i);
  scanf("%f", &y[i][0]);
 }
 for(i = 1; i < n; i++)
 {
  for(j = 0; j < n-i; j++)
  {
   y[j][i] = y[j+1][i-1] - y[j][i-1];
  }
 }
 printf("\nFORWARD DIFFERENCE TABLE\n\n");
 for(i = 0; i < n; i++)
 {
```

```
 printf("%0.2f", x[i]);
 for(j = 0; j < n-i ; j++)
 {
  printf("\t%0.2f", y[i][j]);
 }
 printf("\n");
}
getch();
return 0;
}
```

Output:

```
Enter number of data?
5
Enter data:
x[0]=2
y[0]=3
x[1]=4
y[1]=5
x[2]=6
y[2]=5
x[3]=4
y[3]=3
x[4]=4
y[4]=2

FORWARD DIFFERENCE TABLE

2.00    3.00    2.00    -2.00   0.00    3.00
4.00    5.00    0.00    -2.00   3.00
6.00    5.00    -2.00   1.00
4.00    3.00    -1.00
4.00    2.00
```

3. Newton interpolation using backward method

```
#include<stdio.h>
#include<conio.h>
int main()
{
 float x[20], y[20][20];
 int i,j, n;
 printf("Enter number of data?\n");
 scanf("%d", &n);
 printf("Enter data:\n");
 for(i = 0; i < n ; i++)
 {
  printf("x[%d]=", i);
  scanf("%f", &x[i]);
  printf("y[%d]=", i);
  scanf("%f", &y[i][0]);
 }
 for(i = 1; i < n; i++)
 {
  for(j = n-1; j > i-1; j--)
  {
   y[j][i] = y[j][i-1] - y[j-1][i-1];
  }
 }


 printf("\nBACKWARD DIFFERENCE TABLE\n\n");
 for(i = 0; i < n; i++)
```

```
{
 printf("%0.2f", x[i]);
 for(j = 0; j <= i ; j++)
 {
  printf("\t%0.2f", y[i][j]);
 }
 printf("\n");
}
getch();
return 0;
}
```

```
Enter number of data?
5
Enter data:
x[0]=1
y[0]=2
x[1]=3
y[1]=5
x[2]=4
y[2]=6
x[3]=1
y[3]=2
x[4]=3
y[4]=4

BACKWARD DIFFERENCE TABLE

1.00    2.00
3.00    5.00    3.00
4.00    6.00    1.00    -2.00
1.00    2.00    -4.00   -5.00   -3.00
3.00    4.00    2.00    6.00    11.00   14.00
```

4. Newton interpolation using dividend method

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int x[10], y[10], p[10];
   int k,f,n,i,j=1,f1=1,f2=0;
   printf("\nEnter the number of observations:\n");
   scanf("%d", &n);
   printf("\nEnter the different values of x:\n");
   for (i=1;i<=n;i++)
      scanf("%d", &x[i]);
   printf("\nThe corresponding values of y are:\n");
   for (i=1;i<=n;i++)
      scanf("%d", &y[i]);
   f=y[1];
   printf("\nEnter the value of 'k' in f(k) you want to evaluate:\n");
   scanf("%d", &k);
   do
   {
      for (i=1;i<=n-1;i++)
      {
         p[i] = ((y[i+1]-y[i])/(x[i+j]-x[i]));
         y[i]=p[i];
      }
      f1=1;
      for(i=1;i<=j;i++)
         {
            f1*=(k-x[i]);
         }
      f2+=(y[1]*f1);
      n--;
      j++;
   }

   while(n!=1);
   f+=f2;
   printf("\nf(%d) = %d", k , f);
   getch();
```

Output:

```
Enter the number of observations:
5

Enter the different values of x:
1
2
3
4
5

The corresponding values of y are:
2
4
6
8
10

Enter the value of 'k' in f(k) you want to evaluate:
9

f(9) = 18
--------------------------------
Process exited after 35.83 seconds with return value 13
Press any key to continue . . .
```

Lab3: numerical differentiation and integration

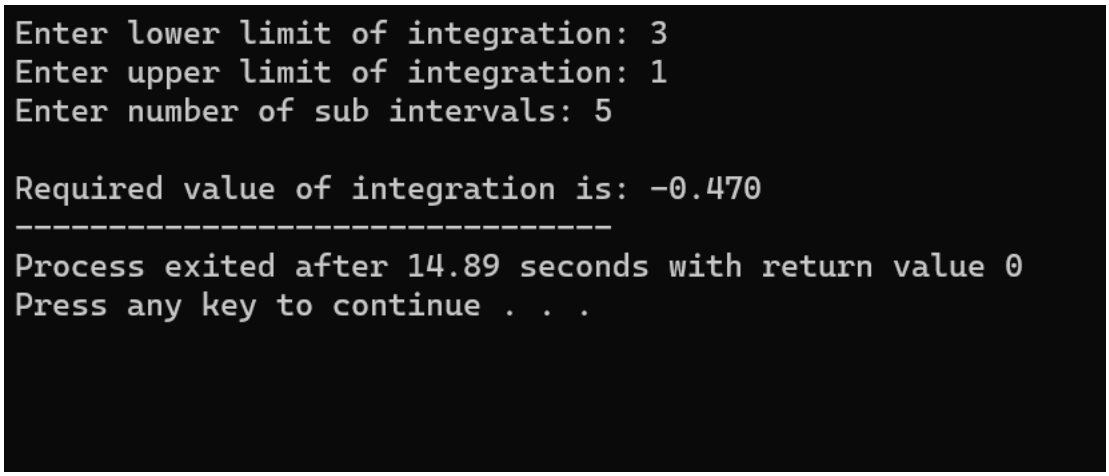
1. Trapezoidal Rule

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

/* Define function here */
#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);
```

```
 getch();
 return 0;
}
```

Output:

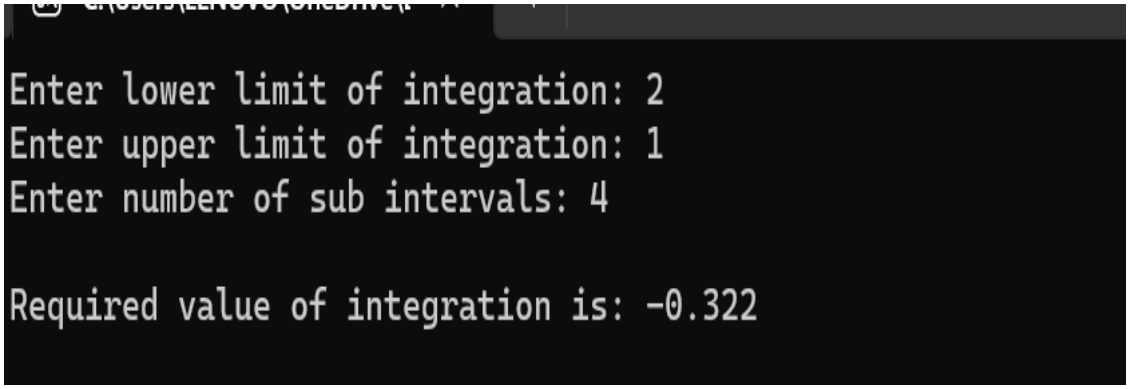


2. Simson's 1/3 rule

```
#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);
getch();
return 0;
}
```

Output:



3. Simson;s 3/8 Rule

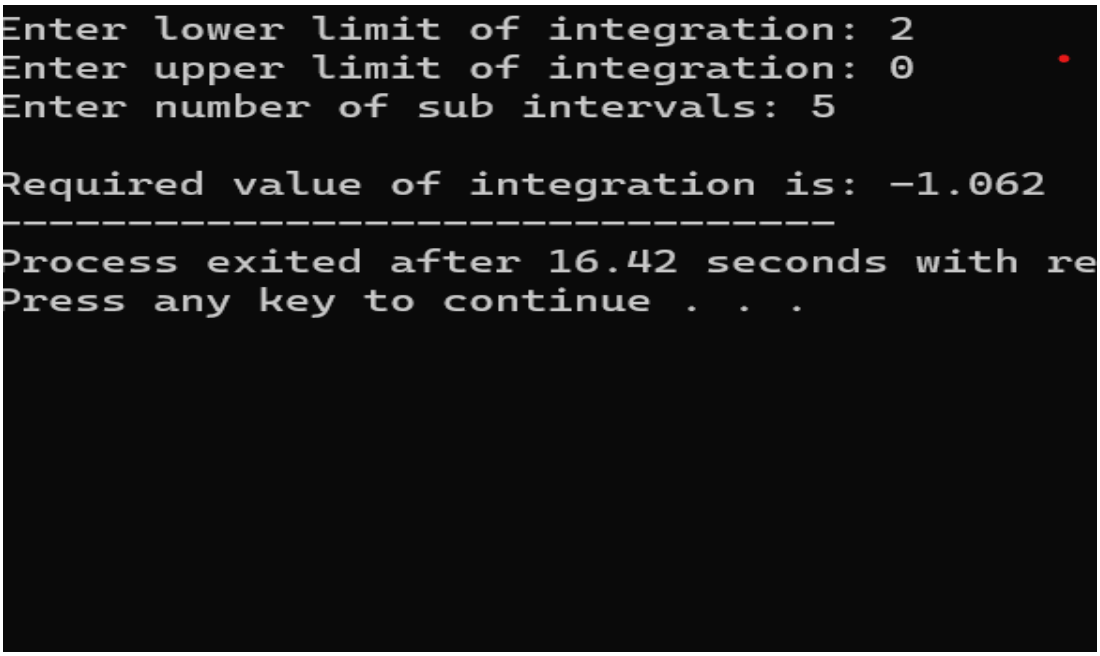
```
#include<stdio.h>
#include<conio.h>
#include<math.h>

/* Define function here */
#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%3 == 0)
```

```
 {
  integration = integration + 2 * f(k);
 }
 else
 {
  integration = integration + 3 * f(k);
 }
}
integration = integration * stepSize*3/8;
printf("\nRequired value of integration is: %.3f", integration);
getch();
return 0;
}
```

Output:



Lab 4:solution of linear algebraic equation

1. Gauss elimination method

```
#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 unknowns: ");
        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]);
    }
    getch();
    return(0);
}
```

Output:

```
Enter number of unknowns: 2
a[1][1] = 1
a[1][2] = 2
a[1][3] = 3
a[2][1] =
4
a[2][2] = 6
a[2][3] = 5

Solution:
x[1] = -4.000
x[2] = 3.500

--------------------------------
Process exited after 6.17 seconds with return value 0
Press any key to continue . . .
```

2. Gauss Jordan method

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define SIZE 10
int main()
{
        float a[SIZE][SIZE], x[SIZE], ratio;
        int i,j,k,n;
        printf("Enter number of unknowns: ");
        scanf("%d", &n);
        printf("Enter coefficients of Augmented Matrix:\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;i++)
        {
                if(a[i][i] == 0.0)
                {
                        printf("Mathematical Error!");
                        exit(0);
                }
                for(j=1;j<=n;j++)
                {
```

```
                    if(i!=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];
                        } }} }
        for(i=1;i<=n;i++)
        {
            x[i] = a[i][n+1]/a[i][i];
        }
        printf("\nSolution:\n");
        for(i=1;i<=n;i++)
        {
            printf("x[%d] = %0.3f\n",i, x[i]);
        }
        getch();
        return(0);
}
```

Output:

```
Enter number of unknowns: 2
Enter coefficients of Augmented Matrix:
a[1][1] = 1
a[1][2] = 2
a[1][3] = 3
a[2][1] = 4
a[2][2] = 5
a[2][3] = 6

Solution:
x[1] = -1.000
x[2] = 2.000
```

3. Matrix inversion using Gauss Jordan method

```
#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 unknowns: ");
        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]);
                }
        }
       /* Applying Gauss Elimination */
        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]);
        }
        getch();
        return(0);
}
```

Output:

```
Enter number of unknowns: 2
a[1][1] = 1
a[1][2] = 2
a[1][3] = 3
a[2][1] =
4
a[2][2] = 6
a[2][3] = 5

Solution:
x[1] = -4.000
x[2] = 3.500

--------------------------------
Process exited after 6.17 seconds with return value 0
Press any key to continue . . .
```

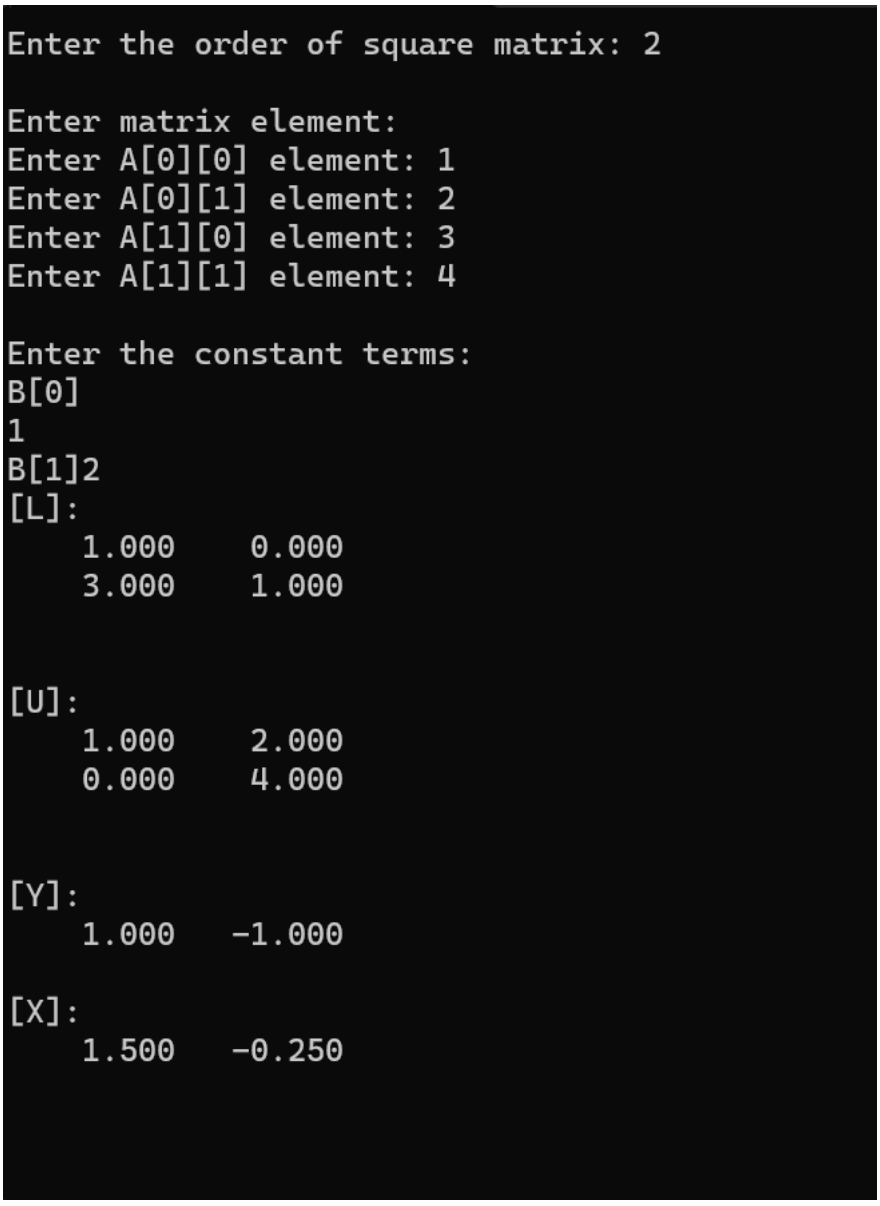
4. Matrix Factorization using Doolittle LU Decomposition

```
#include<stdio.h>
#include<conio.h>
void main()
{
   float A[20][20]= {0},L[20][20]= {0}, U[20][20];
   float B[20]= {0}, X[20]= {0},Y[20]= {0};
   int i,j,k,n;
   printf("Enter the order of square matrix: ");
   scanf("%d",&n);
   printf("\nEnter matrix element:\n");
   for(i=0; i<n; i++)
   {
      for(j=0; j<n; j++)
      {
         printf("Enter A[%d][%d] element: ", i,j);
         scanf("%f",&A[i][j]);
      }
   }
   printf("\nEnter the constant terms: \n");
   for(i=0; i<n; i++)
   {
      printf("B[%d]",i);
      scanf("%f",&B[i]);
   }
   for(j=0; j<n; j++)
   {
      for(i=0; i<n; i++)
      {
         if(i<=j)
         {
            U[i][j]=A[i][j];
            for(k=0; k<i-1; k++)
               U[i][j]-=L[i][k]*U[k][j];
            if(i==j)
               L[i][j]=1;
            else
               L[i][j]=0;
         }
```

```
        else
        {
           L[i][j]=A[i][j];
           for(k=0; k<=j-1; k++)
              L[i][j]-=L[i][k]*U[k][j];
           L[i][j]/=U[j][j];
           U[i][j]=0;
        }
    }
}
printf("[L]: \n");
for(i=0; i<n; i++)
{
    for(j=0; j<n; j++)
       printf("%9.3f",L[i][j]);
    printf("\n");
}
printf("\n\n[U]: \n");
for(i=0; i<n; i++)
{
    for(j=0; j<n; j++)
       printf("%9.3f",U[i][j]);
    printf("\n");
}
for(i=0; i<n; i++)
{
    Y[i]=B[i];
    for(j=0; j<i; j++)
    {
       Y[i]-=L[i][j]*Y[j];
    }
}
printf("\n\n[Y]: \n");
for(i=0; i<n; i++)
{
    printf("%9.3f",Y[i]);
}
for(i=n-1; i>=0; i--)
{
    X[i]= Y[i];
    for(j=i+1; j<n; j++)
    {
       X[i]-=U[i][j]*X[j];
```

```
        }
        X[i]/=U[i][i];
    }
    printf("\n\n[X]: \n");
    for(i=0; i<n; i++)
    {
        printf("%9.3f",X[i]);
    }
    getch();
}
```

Output:



,

5. Matrix factorization using Cholesky's method

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
double *cholesky(double *A, int n) {
    double *L = (double*)calloc(n * n, sizeof(double));
    if (L == NULL)
        exit(EXIT_FAILURE);
    for (int i = 0; i < n; i++)
        for (int j = 0; j < (i+1); j++) {
            double s = 0;
            for (int k = 0; k < j; k++)
                s += L[i * n + k] * L[j * n + k];
            L[i * n + j] = (i == j) ?
                        sqrt(A[i * n + i] - s) :
                        (1.0 / L[j * n + j] * (A[i * n + j] - s));
        }
    return L;
}
void show_matrix(double *A, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            printf("%2.5f ", A[i * n + j]);
        printf("\n");
    }
```

```
}
int main() {
    int n = 3;
    double m1[] = {25, 15, -5,
                   15, 18,  0,
                   -5,  0, 11};
    double *c1 = cholesky(m1, n);
    show_matrix(c1, n);
    printf("\n");
    free(c1);
    n = 4;
    double m2[] = {18, 22,  54,  42,
                   22, 70,  86,  62,
                   54, 86, 174, 134,
                   42, 62, 134, 106};
    double *c2 = cholesky(m2, n);
    show_matrix(c2, n);
    free(c2);

    return 0;
}
```

Output:

```
5.00000 0.00000 0.00000
3.00000 3.00000 0.00000
-1.00000 1.00000 3.00000

4.24264 0.00000 0.00000 0.00000
5.18545 6.56591 0.00000 0.00000
12.72792 3.04604 1.64974 0.00000
9.89949 1.62455 1.84971 1.39262

--------------------------------
Process exited after 0.06707 seconds with return value 0
```

6. Jacob iterative method

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#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
/* Main function */
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
 {
  /* Calculation */
```

```
 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;
}
```

Output:

```
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

7. Gauss sedial iterative method

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#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
/* Main function */
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
 {
  /* Calculation */
  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);

  /* Error */
  e1 = fabs(x0-x1);
  e2 = fabs(y0-y1);
  e3 = fabs(z0-z1);
  count++;
  /* Set value for next iteration */
  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;
}
```

Output:

```
Enter tolerable error:
0.001

Count   x       y       z
1       0.8500  -1.0275 1.0109
2       1.0025  -0.9998 0.9998
3       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

8. Power method

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

#define SIZE 10

int main()
{
        float a[SIZE][SIZE], x[SIZE],x_new[SIZE];
        float temp, lambda_new, lambda_old, error;
        int i,j,n, step=1;

        /* Inputs */
        printf("Enter Order of Matrix: ");
```

```
scanf("%d", &n);
printf("Enter Tolerable Error: ");
scanf("%f", &error);
/* Reading Matrix */
printf("Enter Coefficient of Matrix:\n");
for(i=1;i<=n;i++)
{
        for(j=1;j<=n;j++)
        {
                printf("a[%d][%d]=",i,j);
                scanf("%f", &a[i][j]);
        }
}
/* Reading Intial Guess Vector */
printf("Enter Initial Guess Vector:\n");
for(i=1;i<=n;i++)
{
        printf("x[%d]=",i);
        scanf("%f", &x[i]);
}
/* Initializing Lambda_Old */
lambda_old = 1;
/* Multiplication */
up:
for(i=1;i<=n;i++)
{
        temp = 0.0;
        for(j=1;j<=n;j++)
        {
                temp = temp + a[i][j]*x[j];
        }
        x_new[i] = temp;
}
/* Replacing */
for(i=1;i<=n;i++)
{
       x[i] = x_new[i];
}
/* Finding Largest */
lambda_new = fabs(x[1]);
for(i=2;i<=n;i++)
{
        if(fabs(x[i])>lambda_new)
```

```
                {
                        lambda_new = fabs(x[i]);
                }
        }
        /* Normalization */
        for(i=1;i<=n;i++)
        {
                x[i] = x[i]/lambda_new;
        }
        /* Display */
        printf("\n\nSTEP-%d:\n", step);
        printf("Eigen Value = %f\n", lambda_new);
        printf("Eigen Vector:\n");
        for(i=1;i<=n;i++)
        {
                printf("%f\t", x[i]);
        }
        /* Checking Accuracy */
        if(fabs(lambda_new-lambda_old)>error)
        {
                 lambda_old=lambda_new;
                 step++;
                 goto up;
        }
        getch();
        return(0);
}
```

Output:

```
Enter Order of Matrix: 1
Enter Tolerable Error: 2
Enter Coefficient of Matrix:
a[1][1]=3
Enter Initial Guess Vector:
x[1]=4


STEP-1:
Eigen Value = 12.000000
Eigen Vector:
1.000000

STEP-2:
Eigen Value = 3.000000
Eigen Vector:
1.000000

STEP-3:
Eigen Value = 3.000000
Eigen Vector:
1.000000
```

Lab5. Solution of ordinary differential equation

1. Talyor series

```
#include<stdio.h>
#include<math.h>
int main()
{

   int x,i;
   int fact = 1,n;
   float sum=0;

   printf("\n\nEnter the value of x in the series :  ");
   scanf("%d",&x);

   printf("\nEnter the number of terms in the series  :   ");
   scanf("%d",&n);

   for(i=1;i<n;i++)
   {
      fact = fact*i;
      sum = sum + (pow(x,i)/fact) ;
   }
   sum= sum +1; //Since series starts with 1
   printf("\nThe sum of the taylor series is :  %.2f\n\n",sum);
   return 0;
}
```

Output:

```
Enter the value of x in the series :  2

Enter the number of terms in the series  :   3

The sum of the taylor series is :  5.00


--------------------------------
Process exited after 3.488 seconds with return value 0
Press any key to continue . . .
```

2. Picard's method

```
#include <math.h>
#include <stdio.h>
#define Y1(x) (1 + (x) + pow(x, 2) / 2)
#define Y2(x) (1 + (x) + pow(x, 2) / 2 + pow(x, 3) / 3 + pow(x, 4) / 8)
#define Y3(x) (1 + (x) + pow(x, 2) / 2 + pow(x, 3) / 3 + pow(x, 4) / 8 + pow(x, 5) / 15 + pow(x, 6) / 48)
int main()
{
	double start_value = 0, end_value = 3,
		allowed_error = 0.4, temp;
	double y1[30], y2[30], y3[30];
	int count;
	for (temp = start_value, count = 0;
		temp <= end_value;
		temp = temp + allowed_error, count++) {
		y1[count] = Y1(temp);
		y2[count] = Y2(temp);
		y3[count] = Y3(temp);
	}
	printf("\nX\n");
	for (temp = start_value;
		temp <= end_value;
		temp = temp + allowed_error) {
		printf("%.4lf ", temp);
	}
	printf("\n\nY(1)\n");
	for (temp = start_value, count = 0;
		temp <= end_value;
```

```
            temp = temp + allowed_error, count++) {
            printf("%.4lf ", y1[count]);
        }
        printf("\n\nY(2)\n");
        for (temp = start_value, count = 0;
            temp <= end_value;
            temp = temp + allowed_error, count++) {

            printf("%.4lf ", y2[count]);
        }

        printf("\n\nY(3)\n");
        for (temp = start_value, count = 0;
            temp <= end_value;
            temp = temp + allowed_error, count++) {
            printf("%.4lf ", y3[count]);
        }
        return 0;
}
```

Output:

```
X
0.0000 0.4000 0.8000 1.2000 1.6000 2.0000 2.4000 2.8000

Y(1)
1.0000 1.4800 2.1200 2.9200 3.8800 5.0000 6.2800 7.7200

Y(2)
1.0000 1.5045 2.3419 3.7552 6.0645 9.6667 15.0352 22.7205

Y(3)
1.0000 1.5053 2.3692 3.9833 7.1131 13.1333 24.3249 44.2335
--------------------------------
Process exited after 0.09781 seconds with return value 0
Press any key to continue . . .
```

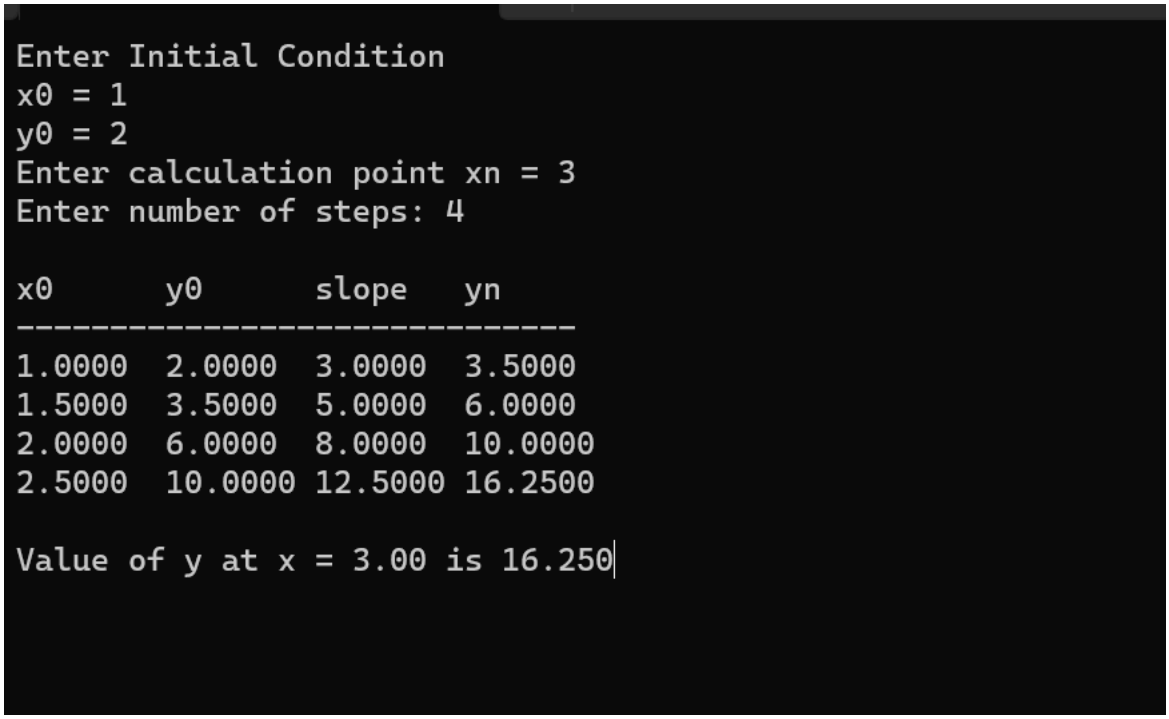
3. Euler's method

```
#include<stdio.h>
#include<conio.h>
#define f(x,y) x+y
int main()
{
 float x0, y0, xn, h, yn, slope;
 int i, n;
 printf("Enter Initial Condition\n");
 printf("x0 = ");
 scanf("%f", &x0);
 printf("y0 = ");
 scanf("%f", &y0);
 printf("Enter calculation point xn = ");
 scanf("%f", &xn);
 printf("Enter number of steps: ");
 scanf("%d", &n);
 /* Calculating step size (h) */
 h = (xn-x0)/n;
 /* Euler's Method */
 printf("\nx0\ty0\tslope\tyn\n");
 printf("------------------------------\n");
 for(i=0; i < n; i++)
 {
  slope = f(x0, y0);
  yn = y0 + h * slope;
  printf("%.4f\t%.4f\t%0.4f\t%.4f\n",x0,y0,slope,yn);
  y0 = yn;
```

```
 x0 = x0+h;
 }
 /* Displaying result */
 printf("\nValue of y at x = %0.2f is %0.3f",xn, yn);

 getch();
 return 0;
}
```

Output:



4. Heun's method

```
#include<conio.h>
#include<stdio.h>
#define f(x,y) 2*y/x
void main()
{
float x,y,h,xn,l;
printf("Program for Solution of Ordinary Differential Equation\nHeun's Method\n");
printf("Enter value for x and y\n");
scanf("%f%f",&x,&y);
printf("Enter value for h and last of x\n");
scanf("%f%f",&h,&xn);
while(x+h<=xn)
{
l=(h/2)*(f(x,y)+f(x+h,y+h*f(x,y)));
y=y+l;
x=x+h;

printf("y = %f\tx = %f\n",y,x);
}
 getch();
}
```

Output:

```
Program for Solution of Ordinary Differential Equation
Heun's Method
Enter value for x and y
1
2
Enter value for h and last of x
1
2
y = 8.500000    x = 2.000000

--------------------------------
Process exited after 12.4 seconds with return value 51
Press any key to continue . . .
```

5. Range-kutta method

```
#include<stdio.h>
#include<conio.h>

#define f(x,y) (y*y-x*x)/(y*y+x*x)

int main()
{
 float x0, y0, xn, h, yn, k1, k2, k3, k4, k;
 int i, n;

 printf("Enter Initial Condition\n");
 printf("x0 = ");
 scanf("%f", &x0);
 printf("y0 = ");
 scanf("%f", &y0);
 printf("Enter calculation point xn = ");
 scanf("%f", &xn);
 printf("Enter number of steps: ");
 scanf("%d", &n);
 h = (xn-x0)/n;
 printf("\nx0\ty0\tyn\n");
 for(i=0; i < n; i++)
 {
  k1 = h * (f(x0, y0));
  k2 = h * (f((x0+h/2), (y0+k1/2)));
  k3 = h * (f((x0+h/2), (y0+k2/2)));
```

```
 k4 = h * (f((x0+h), (y0+k3)));
 k = (k1+2*k2+2*k3+k4)/6;
 yn = y0 + k;
 printf("%0.4f\t%0.4f\t%0.4f\n",x0,y0,yn);
 x0 = x0+h;
 y0 = yn;
}
printf("\nValue of y at x = %0.2f is %0.3f",xn, yn);
getch();
return 0;
}
```

Output:

```
Enter Initial Condition
x0 =
1
y0 = 2
Enter calculation point xn = 0.4
Enter number of steps: 4

x0      y0      yn
1.0000  2.0000  1.9049
0.8500  1.9049  1.7996
0.7000  1.7996  1.6837
0.5500  1.6837  1.5575

Value of y at x = 0.40 is 1.557
```

6. Boundary value problem

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
float f1(float x, float y, float z)
{
   return(z);
}
float f2(float x, float y, float z)
{
   return(x + y);
}
float shoot(float x0, float y0, float z0, float xn, float h, int p)
{
   float x, y, z, k1, k2, k3, k4, l1, l2, l3, l4, k, l, x1, y1, z1;
   x = x0;
   y = y0;
   z = z0;
   do
   {
      k1 = h * f1(x, y, z);
      l1 = h * f2(x, y, z);
      k2 = h * f1(x + h / 2.0, y + k1 / 2.0, z + l1 / 2.0);
      l2 = h * f2(x + h / 2.0, y + k1 / 2.0, z + l1 / 2.0);
      k3 = h * f1(x + h / 2.0, y + k2 / 2.0, z + l2 / 2.0);
      l3 = h * f2(x + h / 2.0, y + k2 / 2.0, z + l2 / 2.0);
```

```
        k4 = h * f1(x + h, y + k3, z + l3);
        l4 = h * f2(x + h, y + k3, z + l3);
        l = 1 / 6.0 * (l1 + 2 * l2 + 2 * l3 + l4);
        k = 1 / 6.0 * (k1 + 2 * k2 + 2 * k3 + k4);
        y1 = y + k;
        x1 = x + h;
        z1 = z + l;
        x = x1;
        y = y1;
        z = z1;
        if (p == 1)
        {
            printf("\n%f\t%f", x, y);
        }
    } while (x < xn);
    return(y);
}
main()
{
    float x0, y0, h, xn, yn, z0, m1, m2, m3, b, b1, b2, b3, e;
    int p = 0;
    printf("\n Enter x0,y0,xn,yn,h:");
    scanf("%f%f%f%f%f", &x0, &y0, &xn, &yn, &h);
    printf("\n Enter the trial M1:");
    scanf("%f", &m1);
    b = yn;
    z0 = m1;
    b1 = shoot(x0, y0, z0, xn, h, p = 1);
```

```
printf("\nB1 is %f", b1);
if (fabs(b1 - b) < 0.00005)
{
   printf("\n The value of x and respective z are:\n");
   e = shoot(x0, y0, z0, xn, h, p = 1);
   return(0);
}
else
{
   printf("\nEnter the value of M2:");
   scanf("%f", &m2);
   z0 = m2;
   b2 = shoot(x0, y0, z0, xn, h, p = 1);
   printf("\nB2 is %f", b2);
}
if (fabs(b2 - b) < 0.00005)
{
   printf("\n The value of x and respective z are\n");
   e = shoot(x0, y0, z0, xn, h, p = 1);
   return(0);
}
else
{
   printf("\nM2=%f\tM1=%f", m2, m1);
   m3 = m2 - ((m2 - m1) / (b2 - b1)) * (b2 - b);
   if (b1 - b2 == 0)
      exit(0);
   printf("\nExact value of M =%f", m3);
```

```
        z0 = m3;
        b3 = shoot(x0, y0, z0, xn, h, p = 0);
    }
    if (fabs(b3 - b) < 0.000005)
    {
        printf("\nThere is solution :\n");
        e = shoot(x0, y0, z0, xn, h, p = 1);
        exit(0);
    }
    do
    {
        m1 = m2;
        m2 = m3;
        b1 = b2;
        b2 = b3;
        m3 = m2 - ((m2 - m1) / (b2 - b1)) * (b2 - b);
        z0 = m3;
        b3 = shoot(x0, y0, z0, xn, h, p = 0);
    } while (fabs(b3 - b) < 0.0005);
    z0 = m3;
    e = shoot(x0, y0, z0, xn, h, p = 1);
}
```

Output:

```
  Enter x0,y0,xn,yn,h:1
2
3
4
5

  Enter the trial M1:1

6.000000        164.291672
B1 is 164.291672
Enter the value of M2:2

6.000000        190.125000
B2 is 190.125000
M2=2.000000     M1=1.000000
Exact value of M =-5.204840
6.000000        -1.#IND00
--------------------------------
Process exited after 14.26 seconds with return value 4290772992
Press any key to continue . . .
```

7. Shooting method

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
float f1(float x,float y,float z)
{
   return(z);
}
float f2(float x,float y,float z)
{
   return(x+y);
}
float shoot(float x0,float y0,float z0,float xn,float h,int p)
{
   float x,y,z,k1,k2,k3,k4,l1,l2,l3,l4,k,l,x1,y1,z1;
   x=x0;
   y=y0;
   z=z0;
   do
   {
      k1=h*f1(x,y,z);
      l1=h*f2(x,y,z);
      k2=h*f1(x+h/2.0,y+k1/2.0,z+l1/2.0);
      l2=h*f2(x+h/2.0,y+k1/2.0,z+l1/2.0);
      k3=h*f1(x+h/2.0,y+k2/2.0,z+l2/2.0);
      l3=h*f2(x+h/2.0,y+k2/2.0,z+l2/2.0);
      k4=h*f1(x+h,y+k3,z+l3);
```

```
        l4=h*f2(x+h,y+k3,z+l3);
        l=1/6.0*(l1+2*l2+2*l3+l4);
        k=1/6.0*(k1+2*k2+2*k3+k4);
        y1=y+k;
        x1=x+h;
        z1=z+l;
        x=x1;
        y=y1;
        z=z1;
        if(p==1)
        {
           printf("\n%f\t%f",x,y);
        }
    }while(x<xn);
    return(y);
}
 void main()
{
    float x0,y0,h,xn,yn,z0,m1,m2,m3,b,b1,b2,b3,e;
    int p=0;
    printf("\n  Enter x0,y0,xn,yn,h:");
    scanf("%f%f%f%f%f",&x0,&y0,&xn,&yn,&h);
    printf("\n  Enter the trial M1:");
    scanf("%f",&m1);
    b=yn;
    z0=m1;
    b1=shoot(x0,y0,z0,xn,h,p=1);
    printf("\nB1 is %f",b1);
```

```
if(fabs(b1-b)<0.00005)
{
   printf("\n  The value of x and respective z are:\n");
   e=shoot(x0,y0,z0,xn,h,p=1);
   return(0);
}
else
{
printf("\nEnter the value of M2:");
scanf("%f",&m2);
z0=m2;
b2=shoot(x0,y0,z0,xn,h,p=1);
printf("\nB2 is %f",b2);
}
if(fabs(b2-b)<0.00005)
{
    printf("\n  The value of x and respective z are\n");
    e= shoot(x0,y0,z0,xn,h,p=1);
    return(0);
}
else
{
   printf("\nM2=%f\tM1=%f",m2,m1);
   m3=m2+(((m2-m1)*(b-b2))/(1.0*(b2-b1)));
   if(b1-b2==0)
   exit(0);

   printf("\nExact value of M =%f",m3);
```

```
        z0=m3;
        b3=shoot(x0,y0,z0,xn,h,p=0);
    }
    if(fabs(b3-b)<0.000005)
    {
        printf("\nThere is solution :\n");
        e=shoot(x0,y0,z0,xn,h,p=1);
        exit(0);
    }
      do
      {
         m1=m2;
         m2=m3;
         b1=b2;
         b2=b3;
         m3=m2+(((m2-m1)*(b-b2))/(1.0*(b2-b1)));
         z0=m3;
         b3=shoot(x0,y0,z0,xn,h,p=0);

      }while(fabs(b3-b)<0.0005);
      z0=m3;
      e=shoot(x0,y0,z0,xn,h,p=1);
    }
```

Output:

```
Enter Initial Condition
x0 = 1
y0 = 2
Enter calculation point xn = 3
Enter number of steps: 4

x0      y0      slope   yn
------------------------------
1.0000  2.0000  3.0000  3.5000
1.5000  3.5000  5.0000  6.0000
2.0000  6.0000  8.0000  10.0000
2.5000  10.0000 12.5000 16.2500

Value of y at x = 3.00 is 16.250
```