NUMERICAL METHODS OF COMPUTATIONAL PROGRAMMING LAB
MATH-204-F

IV SEMESTER

DRONACHARYA
College of Engineering

Department Of Electronics & Communication Engg
Dronacharya College Of Engineering
Khentawas, Gurgaon – 123506

NUMERICAL METHODS LAB
IV SEM.

LIST OF EXPERIMENTS

Program 1

OBJECTIVES: To find the roots of non linear equations using Bisection method.

SOURCE CODE:

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


/*  define prototype for USER-SUPPLIED function f(x)  */

    double ffunction(double x);


/*  EXAMPLE for "ffunction"   */

    double ffunction(double x)

    {
        return (x * sin(x) - 1);
    }

/* ------------------------------------------------------ */

/*  Main program for algorithm 2.2  */

    void main()

{
    double Delta = 1E-6;          /* Tolerance for width of interval */
    int Satisfied = 0;            /* Condition for loop termination  */
    double A, B;                  /* Endpoints of the interval [A,B] */
    double YA, YB;        /* Function values at the interval-borders */
    int Max;      /* Calculation of the maximum number of iterations */
    int K;                                          /* Loop Counter */
    double C, YC;   /* Midpoint of interval and function value there */


    printf("-------------------------------------------------------\n");
    printf("Please enter endpoints A and B of the interval [A,B]\n");
    printf("EXAMPLE : A = 0 and B = 2. Type:   0 2 \n");
    scanf("%lf %lf", &A, &B);
    printf("The interval ranges from %lf to %lf\n", A,B);

    YA = ffunction(A);    /* compute function values */
    YB = ffunction(B);
    Max = (int) ( 1 + floor( ( log(B-A) - log(Delta) ) / log(2) ) );
    printf("Max = %d\n",Max);

    /* Check to see if the bisection method applies */
```

```
    if( ( (YA >= 0) && (YB >=0) ) || ( (YA < 0) && (YB < 0) )   ) {
      printf("The values ffunction(A) and ffunction(B)\n");
      printf("do not differ in sign.\n");
      exit(0);     /* exit program */
    }

    for(K = 1; K <= Max ; K++) {

        if(Satisfied == 1)  break;

        C = (A + B) / 2;     /* Midpoint of interval */
        YC = ffunction(C);   /* Function value at midpoint */


  if( YC == 0) {         /* first 'if'          */
            A = C;       /* Exact root is found */
            B = C;
  }
  else if( ( (YB >= 0) && (YC >=0) ) || ( (YB < 0) && (YC < 0) )   ) {
       B = C;      /* Squeeze from the right */
      YB = YC;
  }
  else {
       A = C;      /* Squeeze from the left */
      YA = YC;
  }

  if( (B-A) < Delta ) Satisfied = 1; /* check for early convergence */

    }  /* end of 'for'-loop */

    printf("-----------------------------------------------\n");
    printf("The maximum number of iterations is : %d\n",Max);
    printf("The number of performed iterations is : %d\n",K - 1);
    printf("-----------------------------------------------\n");
    printf("The computed root of f(x) = 0 is : %lf \n",C);
    printf("-----------------------------------------------\n");
    printf("The accuracy is +- %lf\n", B-A);
    printf("-----------------------------------------------\n");
    printf("The value of the function f(C) is %lf\n",YC);

}   /* End of main program */
```

Quiz

Q.1)What is use of Bisection Method?
Answer: Bisection method is used to find the root of non-linear equation.

Q.2) What is Bisection method?
Answer: The **bisection method** in mathematics is a root-finding method which repeatedly bisects an interval and then selects a subinterval in which a root must lie for further processing. It is a very simple and robust method, but it is also relatively slow. Because of this, it is often used to obtain a rough approximation to a solution which is then used as a starting point for more rapidly converging methods

Q.3) What are the observations of Bisection method?
Answer: There are two observation of Bisection method. They are
Obs. 1: Since the new interval containing the root, is exactly half the length of the previous one, the interval width is reduced by a factor of ½ at each step. At the end of the nth step, the new interval will therefore, be of length $(b-a)/2^n$
Obs. 2: As the error decreases with each step by a factor of ½, the convergence in the bisection method is linear.

Q.4) What are the disadvantages of Bisection Method?
Answer:The convergce process in the bisection method is very slow. It depends only on the choice of end points of the interval [a,b]. The function f(x) does not have any role in finding the point c (which is just the mid-point of a and b). It is used only to decide the next smaller interval [a,c] or [c,b].

Q.5) Which property is used in Bisection method?
Answer: Bisection method uses intermediate value property.

Program 2

OBJECTIVES: To find the roots of non linear equations using Newton's method.

Sorce code:

```
Algorithm 2.5 (Newton-Raphson Iteration). To find a root
 f(x) = 0 given one initial approximation  p_0  and using the iteration

                          f(p_(k-1))
      p_k  =  p_(k-1)  -  ----------     for k = 1, 2, ...
                          f'(p_(k-1))

------------------------------------------------------------------------
----
*/

/* User has to supply a function named :  ffunction
   and its first derivative :            dffunction

   An example is included in this program */

#include<stdio.h>
#include<stdlib.h>
#include<math.h>


/*  define prototype for USER-SUPPLIED function f(x)  */

    double ffunction(double x);
    double dffunction(double x);


/*  EXAMPLE for "ffunction"   */

    double ffunction(double x)

    {
        return ( pow(x,3) - 3 * x + 2 );
    }

/*  EXAMPLE for "dffunction" , first derivative of ffunction. */

    double dffunction(double x)

    {
        return ( 3 * pow(x,2) - 3 );
    }


/* ------------------------------------------------------ */
```

```
/*  Main program for algorithm 2.5  */

    void main()

{
    double Delta = 1E-6;       /* Tolerance  */
    double Epsilon = 1E-6;     /* Tolerance  */
    double Small = 1E-6;       /* Tolerance  */

    int Max = 99;  /* Maximum number of iterations  */
    int Cond = 0;  /* Condition fo loop termination */
    int K;         /* Counter for loop              */

    double P0;    /* INPUT : Must be close to the root */
    double P1;    /* New iterate    */
    double Y0;    /* Function value */
    double Y1;    /* Function value */
    double Df;    /* Derivative     */
    double Dp;
    double RelErr;


    printf("-----------------------------------------------\n");
    printf("Please enter initial approximation of root !\n");
    scanf("%lf",&P0);
    printf("-----------------------------------------------\n");
    printf("Initial value for root: %lf\n",P0);

    Y0 = dffunction(P0);

    for ( K = 1; K <= Max ; K++) {

        if(Cond) break;

        Df = dffunction(P0);      /* Compute the derivative  */

         if( Df == 0) {       /* Check division by zero */
             Cond = 1;
             Dp   = 0;
         }

         else Dp = Y0/Df;

        P1 = P0 - Dp;       /* New iterate */
        Y1 = ffunction(P1);   /* New function value */

        RelErr = 2 * fabs(Dp) / ( fabs(P1) + Small );  /* Relative
error */

        if( (RelErr < Delta)  && (fabs(Y1) < Epsilon) ) { /* Check for
*/

            if( Cond != 1) Cond = 2;                     /*
convergence */

        }
```

```
        P0 = P1;
        Y0 = Y1;
    }

    printf("-----------------------------------------------\n");
    printf("The current %d -th iterate is %lf\n",K-1, P1);
    printf("Consecutive iterates differ by %lf\n",Dp);
    printf("The value of f(x) is %lf\n",Y1);
    printf("-----------------------------------------------\n");

if(Cond == 0) printf("The maximum number of iterations was exceeded
!\n");

if(Cond == 1) printf("Division by zero was encountered !\n");

if(Cond == 2) printf("The root was found with the desired tolerance
!\n");

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

}   /* End of main program */
```

Quiz

Q.1)What is the use of Newton's Raphsons method?

Answer: Newton Raphsons method is used to find the root of non Linear equation.

Q2)Explain Newton Raphson method in detail.

Answer: The Newton-Raphson method uses an iterative process to approach one root of a function. The specific root that the process locates depends on the initial, arbitrarily chosen x-value.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Here, x_n is the current known x-value, $f(x_n)$ represents the value of the function at x_n, and $f'(x_n)$ is the derivative (slope) at x_n. x_{n+1} represents the next x-value that you are trying to find. Essentially, f'(x), the derivative represents f(x)/dx (dx = delta-x). Therefore, the term f(x)/f'(x) represents a value of dx.

$$\frac{f(x)}{f'(x)} = \frac{f(x)}{f(x)/\Delta x} = \Delta x$$

The more iterations that are run, the closer dx will be to zero (0).

Q.3)What are the observation of Netwons Raphsons method?

Obs. 1: Newton's method is useful in cases of large values of f'(x) i.e. when the graph of f(x) while crossing the x-axis is nearly vertical.

Obs. 2: This method replace the part of curve between the point A0 and x-axis by means of the tangent to the curve at A0.

Obs. 3:Newton method is used to improve the result obtained by other method. It is applicable to the solution of both algebraic and transcendental equations.

Obs.4: Newton's formula converges provided the initial approximation x0 is choosen sufficiently close to the root.

Obs. 5:Newton's method has a quadratic converges.

Q.4) Which of the following method converges faster-Regula Falsi method or Newton-Raphson method?

Answer: Newton Raphson method.

Q.5)What is the other name for Newton-Raphson method

Answer: The other name for Newton Raphson method is Newton Iteration method

Program 3

OBJECTIVES: Curve fitting by least square approximations

```
Algorithm 5.2 (Least-Squares Polynomial).

 To construct the least-squares polynomial of degree  M  of the form

                                 2        3            M-1            M
 P_M(x)  =  c_1 + c_2*x + c_3*x  + c_4*x  +...+ c_M*x    + c_(M+1)*x

 that fits the N data points (x_1, y_1), ... , (x_N, y_N).

-------------------------------------------------------------------------
----
*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

/* ------------------------------------------------------ */

/*  Main program for algorithm 5.2  */

/*  remember : in C the fields begin with element 0  */

#define DMAX  15    /* Maximum degree of polynomial */
#define NMAX  20    /* Maximum number of points     */


void main(void)
{
    extern void FactPiv();

    int R, K, J;                    /* Loop counters     */
    double X[NMAX-1], Y[NMAX-1];    /* Points (x,y)      */
    double A[DMAX][DMAX];           /* A                 */
    double B[DMAX];                 /* B                 */
    double C[DMAX];
    double P[2*DMAX];
    int N;                          /* Number of points :     INPUT */
    int M;                          /* Degree of polynomial : INPUT */
    double x, y;
    int p;


    printf("Try the examples on page 277 or 281 of the book !\n");
    printf("-------------------------------------------------\n");
```

```
    do  /* force proper input */
    {
      printf("Please enter degree of polynomial [Not more than
%d]\n",DMAX);
      scanf("%d", &M);
    } while( M > DMAX);

    printf("----------------------------------------\n");
    do  /* force proper input */
    {
      printf("Please enter number of points [Not more than
%d]\n",NMAX);
      scanf("%d", &N);
    } while( N > NMAX);

    printf("You say there are %d points.\n", N);
    printf("-----------------------------------------------------\n");
    printf("Enter points in pairs like :  2.4, 4.55:\n");

    for (K = 1; K <= N; K++)
    {
      printf("Enter %d st/nd/rd pair of points:\n", K);
      scanf("%lf, %lf", &X[K-1], &Y[K-1]);
      printf("You entered the pair (x,y) = %lf, %lf\n", X[K-1], Y[K-
1]);
    }


    /* Zero the array */

    for (R = 1; R <= M+1; R++) B[R-1] = 0;

    /* Compute the column vector */

    for (K = 1; K <= N; K++)
    {
      y = Y[K-1];
      x = X[K-1];
      p = 1;

      for( R = 1; R <= M+1; R++ )
      {
         B[R-1] += y * p;
         p = p*x;
      }
    }

    /* Zero the array */

    for (J = 1; J <= 2*M; J++) P[J] = 0;

    P[0] = N;

    /* Compute the sum of powers of x_(K-1) */

    for (K = 1; K <= N; K++)
    {
```

```
        x = X[K-1];
        p = X[K-1];

        for (J = 1; J <= 2*M; J++)
        {
            P[J] += p;
            p = p * x;
        }
    }

    /* Determine the matrix entries */

    for (R = 1; R <= M+1; R++)
    {
        for( K = 1; K <= M+1; K++) A[R-1][K-1] = P[R+K-2];
    }



    /* Solve the linear system of M + 1 equations : A*C = B
       for the coefficient vector C = (c_1,c_2,..,c_M,c_(M+1)) */


    FactPiv(M+1, A, B);

}   /* end main */

/*-------------------------------------------------------------*/

void FactPiv(N, A, B)
int N;
double A[DMAX][DMAX];
double *B;
{

    int K, P, C, J;                 /* Loop counters               */
    int    Row[NMAX];               /* Field with row-number       */
    double X[DMAX], Y[DMAX
    ];
    double SUM, DET = 1.0;

    int  T;


    /* Initialize the pointer vector */

    for (J = 1; J<= N; J++) Row[J-1] = J - 1;



    /* Start LU factorization */

    for (P = 1; P <= N - 1; P++)
    {

      /* Find pivot element */
```

```
      for (K = P + 1; K <= N; K++)
      {
        if ( fabs(A[Row[K-1]][P-1]) > fabs(A[Row[P-1]][P-1]) )
        {
        /* Switch the index for the p-1 th pivot row if necessary */
           T        = Row[P-1];
           Row[P-1] = Row[K-1];
           Row[K-1] = T;
           DET      = - DET;
        }


      } /* End of simulated row interchange */

        if (A[Row[P-1]][P-1] == 0)
        {
           printf("The matrix is SINGULAR !\n");
           printf("Cannot use algorithm --> exit\n");
           exit(1);
        }

     /* Multiply the diagonal elements */

     DET = DET * A[Row[P-1]][P-1];

     /* Form multiplier */

     for (K = P + 1; K <= N; K++)
     {
       A[Row[K-1]][P-1]= A[Row[K-1]][P-1] / A[Row[P-1]][P-1];

       /* Eliminate X_(p-1) */

       for (C = P + 1; C <= N + 1; C++)
       {
          A[Row[K-1]][C-1] -= A[Row[K-1]][P-1] * A[Row[P-1]][C-1];
       }
     }

  } /* End of  L*U  factorization routine */


    DET = DET * A[Row[N-1]][N-1];



    /* Start the forward substitution */

    for(K = 1; K <= N; K++) Y[K-1] = B[K-1];

    Y[0] = B[Row[0]];
    for ( K = 2; K <= N; K++)
    {
      SUM =0;
      for ( C = 1; C <= K -1; C++) SUM += A[Row[K-1]][C-1] * Y[C-1];
      Y[K-1] = B[Row[K-1]] - SUM;
    }
```

```
    if( A[Row[N-1]][N-1] == 0)
    {
      printf("The matrix is SINGULAR !\n");
      printf("Cannot use algorithm --> exit\n");
      exit(1);
    }


    /* Start the back substitution */

    X[N-1] = Y[N-1] / A[Row[N-1]][N-1];

    for (K = N - 1; K >= 1; K--)
    {
      SUM = 0;
      for (C = K + 1; C <= N; C++)
      {
           SUM += A[Row[K-1]][C-1] * X[C-1];
      }

      X[K-1] = ( Y[K-1] - SUM) / A[Row[K-1]][K-1];

    }  /* End of back substitution */

    /* Output */

    printf("-------------------------------------------------------:\n");
    printf("The components of the vector with the solutions are:\n");
    for( K = 1; K <= N; K++)  printf("X[%d] = %lf\n", K, X[K-1]);

}




/*
-------------------------------------------------------------------------
----


");

}   /* End of main programm */
```

Quiz

Q.1) Write down working procedure of Least square method

Answer: (a) To fit the straight line y=a+bx

i) Substitute the observed the set of n values in this equation.

ii) Form normal equation for each constant i.e. $\sum y=na+b\sum x$, $\sum xy=a\sum x+b\sum x^2$

iii) Solve these normal equations as simultaneous equation for a & b.

iv) Substitute the values of a & b in y=a+bx, which is required line of best fit

b)To fit the parabola: $y=a+bx+cx^2$

i)Form the normal equation $\sum y=n+b\sum x+c\sum x^2$

ii)Solve these equation simultaneous equation for a, b, c.

iii)Substitute the values of a, b, c in $y=a+bx+cx^2$

c)In general, the curve $y=a+bx+cx^2+\ldots+kx^{m-1}$ can be fitted to a given data by writing m normal equations.

Q.2) What is observation of least square method?

Answer: On calculating $\partial^2E/\partial a^2$, $\partial^2E/\partial b^2$, $\partial^2E/\partial c^2$ and substituting the values of a, b, c just obtained, we will observe that each is positive i.e. E is a minimum.

Q.3) What is the use of Least square method?

Answer: Least square method is used to fit a unique curve to a given data.

Q.4)What is the error for the curve $y= a+bx+cx^2$?

Answer: The error for the curve $y= a+bx+cx^2$ is $e_i=y_i-n_i$

Q.5) Whether E value is minimum or maximum in Least square method?

Answer: E value is minimum in Least Square method

Program 4

OBJECTIVES: To solve the system of linear equations using gauss elimination method

Source Code:

```
(Gauss-Elimination-Iteration).

 To solve the linear system  AX = B  by starting with  P_0 = 0
 and generating a sequence  { P_K}  that converges to the
 solution  P  (i.e., AP = B).  A sufficient condition for the
 method to be applicable is that  A  is diagonally dominant.

-------------------------------------------------------------------------
----
*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

/* ------------------------------------------------------- */

/*  Main program for algorithm 3.5  */

/*  remember : in C the fields begin with element 0  */

#define N 4

void main(void)
{
       Float a[N][N+1],x[N],t,s;
       Int i,j,k;
       Printf("Enter the element of the  argumented matrix row
wise\n");

       For(i=0;i<N;i++)
       For(j=0;j<N+1;j++)
            Scanf("%f",&a[i][j]);
       For(j=0;j<N-1;j++)
       For(i=j+1;i<N;i++)
       {
            t=a[i][j]/a[j][j];
            for(k=0;k<N+1;k++)
            a[i][k]=a[j][k]*t;
       }

       Printf("The upper triangular matrix is:\n");

       For(i=0;i<N;i++)
```

```
        For(j=0;j<N+1;j++)
                printf("%8.4f%",a[i][j]);

                Printf("\n");

        For(i=N-1;i>=0;i--)
        {

        s=0;

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

                s+=a[i][j]*x[j];

                x[i]=(a[i][N]-s)/a[i][i];

        }

        Printf("The solution is :\n");

        For(i=0;i<N;i++)

        Printf("x[%3d]=%7.4f\n", i+1,x[i]);

}
```

Q.1)What is Gauss Elimination method?

Answer: Gaussian elimination is a method of solving a linear system $Ax = b$(consisting of $m$equations in $n$unknowns) by bringing the augmented matrix

$$[A\ b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]$$

to an upper triangular form

$$\left[\begin{array}{cccc|c} c_{11} & c_{12} & \cdots & c_{1n} & d_1 \\ 0 & c_{22} & \cdots & c_{2n} & d_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & c_{mn} & d_m \end{array}\right].$$

This elimination process is also called the forward elimination method.

Q.2) Explain the procedure of Gauss elimination method.

Answer: Consider a linear system.

1. Construct the augmented matrix for the system;
2. Use elementary row operations to transform the augmented matrix into a triangular one;
3. Write down the new linear system for which the triangular matrix is the associated augmented matrix;
4. Solve the new system. You may need to assign some parametric values to some unknowns, and then apply the method of back substitution to solve the new system.

Q.3)Solve the following system via Gaussian elimination

$$\begin{cases} 2x - 3y - z + 2w + 3v = 4 \\ 4x - 4y - z + 4w + 11v = 4 \\ 2x - 5y - 2z + 2w - v = 9 \\ 2y + z + 4v = -5 \end{cases}$$

Answer:The augmented matrix is

$$\left(\begin{array}{ccccc|c} 2 & -3 & -1 & 2 & 3 & 4 \\ 4 & -4 & -1 & 4 & 11 & 4 \\ 2 & -5 & -2 & 2 & -1 & 9 \\ 0 & 2 & 1 & 0 & 4 & -5 \end{array}\right).$$

We use elementary row operations to transform this matrix into a triangular one. We keep the first row and use it to produce all zeros elsewhere in the first column. We have

$$\left(\begin{array}{ccccc|c} 2 & -3 & -1 & 2 & 3 & 4 \\ 0 & 2 & 1 & 0 & 5 & -4 \\ 0 & -2 & -1 & 0 & -4 & 5 \\ 0 & 2 & 1 & 0 & 4 & -5 \end{array}\right).$$

Next we keep the first and second row and try to have zeros in the second column. We get

$$\left(\begin{array}{ccccc|c} 2 & -3 & -1 & 2 & 3 & 4 \\ 0 & 2 & 1 & 0 & 5 & -4 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 \end{array}\right).$$

Next we keep the first three rows. We add the last one to the third to get

$$\left(\begin{array}{ccccc|c} 2 & -3 & -1 & 2 & 3 & 4 \\ 0 & 2 & 1 & 0 & 5 & -4 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right).$$

This is a triangular matrix. Its associated system is

$$\begin{cases} 2x - 3y - z + 2w + 3v = 4 \\ 2y + z + 5v = -4 \\ v = 1 \end{cases}$$

Clearly we have $v = 1$. Set $z=s$ and $w=t$, then we have

$$y = -2 - \frac{1}{2}z - \frac{5}{2}v = -\frac{9}{2} - \frac{1}{2}s.$$

The first equation implies

$$x = 2 + \frac{3}{2}y + \frac{1}{2}z - \frac{3}{2}w - v.$$

Using algebraic manipulations, we get

$x = -\frac{25}{4} - \frac{1}{4}s - t.$

Putting all the stuff together, we have

$$\begin{pmatrix} x \\ y \\ z \\ w \\ v \end{pmatrix} = \begin{pmatrix} -\frac{25}{4} - \frac{1}{4}s - t \\ -\frac{9}{2} - \frac{1}{2}s \\ s \\ t \\ 1 \end{pmatrix}.$$

Q.4) What is use of Gauss elimination method?

Answer: Gauss elimination is used to solve linear equation of the form Ax = b.

Q.5)What is the order of matrix in Gauss elimination method?

Answer: The matrix order is same as given matrix A.

Program 5

OBJECTIVES: To Solve The System Of Linear Equations Using Gauss - Seidal Iteration Method

(Code for GAUSS SEIDEL METHOD)

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define ESP 0.0001
#define X1(x2,x3) ((17 - 20*(x2) + 2*(x3))/20)
#define X2(x1,x3) ((-18 - 3*(x1) + (x3))/20)
#define X3(x1,x2) ((25 - 2*(x1) + 3*(x2))/20)


void main()
{
  double x1=0,x2=0,x3=0,y1,y2,y3;
  int i=0;
  clrscr();
  printf("\n_____\n");
  printf("\n   x1\t\t   x2\t\t   x3\n");
  printf("\n_____\n");
  printf("\n%f\t%f\t%f",x1,x2,x3);
  do
  {
   y1=X1(x2,x3);
   y2=X2(y1,x3);
   y3=X3(y1,y2);
   if(fabs(y1-x1)<ESP && fabs(y2-x2)<ESP && fabs(y3-x3)<ESP )
   {
     printf("\n_____\n");
     printf("\n\nx1 = %.3lf",y1);
     printf("\n\nx2 = %.3lf",y2);
     printf("\n\nx3 = %.3lf",y3);
     i = 1;
   }
   else
   {
     x1 = y1;
     x2 = y2;
     x3 = y3;
     printf("\n%f\t%f\t%f",x1,x2,x3);
   }
  }while(i != 1);
getch();
}

/*

     OUT PUT
       ___________

_____

   x1              x2              x3
```

```
_____________________________________________

0.000000          0.000000       0.000000
0.850000         -1.027500       1.010875
1.978588         -1.146244       0.880205
2.084265         -1.168629       0.866279
2.105257         -1.172475       0.863603
2.108835         -1.173145       0.863145
2.109460         -1.173262       0.863065
2.109568         -1.173282       0.863051
_____________________________________________


x1 = 2.110

x2 = -1.173

x3 = 0.863


*/
```

Quiz

Q.1)Write down converges property of Gauss Seidal method.

Answer: The convergence properties of the Gauss–Seidel method are dependent on the matrix *A*. Namely, the procedure is known to converge if either:

- *A* is symmetric positive-definite, or
- *A* is strictly or irreducibly diagonally dominant.

The Gauss–Seidel method sometimes converges even if these conditions are not satisfied.

Q.2) Explain Gauss Seidal method.

Answer: Given a square system of *n* linear equations with unknown **x**:

$$A\mathbf{x} = \mathbf{b}$$

where:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Then *A* can be decomposed into a lower triangular component L_*, and a strictly upper triangular component *U*:

$$A = L_* + U \quad \text{where} \quad L_* = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad U = \begin{bmatrix} 0 & a_{12} & \cdots \\ 0 & 0 & \cdots \\ \vdots & \vdots & \ddots \\ 0 & 0 & \cdots \end{bmatrix}$$

The system of linear equations may be rewritten as:

$$L_*\mathbf{x} = \mathbf{b} - U\mathbf{x}$$

The Gauss–Seidel method is an iterative technique that solves the left hand side of this expression for **x**, using previous value for **x** on the right hand side. Analytically, this may be written as:

$$\mathbf{x}^{(k+1)} = L_*^{-1}(\mathbf{b} - U\mathbf{x}^{(k)}).$$

However, by taking advantage of the triangular form of L_*, the elements of $\mathbf{x}^{(k+1)}$ can be computed sequentially using forward substitution:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j>i} a_{ij} x_j^{(k)} - \sum_{j<i} a_{ij} x_j^{(k+1)} \right), \quad i = 1, 2, \ldots, n.$$

Note that the sum inside this computation of $x_i^{(k+1)}$ requires each element in $\mathbf{x}^{(k)}$ except $x_i^{(k)}$ itself.

The procedure is generally continued until the changes made by an iteration are below some tolerance.

Q.3)What is the use of Gauss Seidal method.
Answer: In numerical linear algebra, the **Gauss–Seidel method**, also known as the **Liebmann method** or the **method of successive displacement**, is an iterative method used to solve a linear system of equations. It is named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig von Seidel, and is similar to the Jacobi method. Though it can be applied to any matrix with non-zero elements on the diagonals, convergence is only guaranteed if the matrix is either diagonally dominant, or symmetric and positive definite.

Q.4)Sove the following equations using Gauss Seidal Method.

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6, \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25, \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11, \\ 3x_2 - x_2 + 8x_4 &= 15. \end{aligned}$$

Solving for x_1, x_2, $x_3$and $x_4$gives:

$$\begin{aligned} x_1 &= x_2/10 - x_3/5 + 3/5, \\ x_2 &= x_1/11 + x_3/11 - 3x_4/11 + 25/11, \\ x_3 &= -x_1/5 + x_2/10 + x_4/10 - 11/10, \\ x_4 &= -3x_2/8 + x_3/8 + 15/8. \end{aligned}$$

Suppose we choose (0, 0, 0, 0) as the initial approximation, then the first approximate solution is given by

$$x_1 = 3/5 = 0.6,$$
$$x_2 = (3/5)/11 + 25/11 = 3/55 + 25/11 = 2.3272,$$
$$x_3 = -(3/5)/5 + (2.3272)/10 - 11/10 = -3/25 + 0.23272 - 1.1 = -0.9873,$$
$$x_4 = -3(2.3272)/8 + (-0.9873)/8 + 15/8 = 0.8789.$$

Using the approximations obtained, the iterative procedure is repeated until the desired accuracy has been reached. The following are the approximated solutions after four iterations.

x_1	x_2	x_3	x_4
0.6	2.32727	−0.987273	0.878864
1.03018	2.03694	−1.01446	0.984341
1.00659	2.00356	−1.00253	0.998351
1.00086	2.0003	−1.00031	0.99985

method is used to find it.

The exact solution of the system is (1, 2, −1, 1).

Q.5) Which method is similar to Jacobi method?

Answer: Gauss Seidal method is similar to Jacobi method.

Program 6

OBJECTIVES: To Solve The System Of Linear Equations Using Gauss - Jorden Method.

(Code Gauss Jordan method)

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void main()
{float a[6][6],b[6],x[6],t,s;
int i,j,n,k;
clrscr();
cout<<"Enter the maximum no. of matrix"<<endl;
cin>>n;
cout<<"Enter th elements of matrix"<<endl;
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
cin>>a[i][j];
}
}
cout<<"enter the right constant"<<endl;
for(i=0;i<n;i++)
{
cin>>a[i][n];
}
for(k=0;k<n;k++)
{
for(i=0;i<n;i++)
if(i!=k)
{
for(j=k+1;j<n+1;j++)
{
a[i][j]=a[i][j]-(a[i][k]/a[k][k])*a[k][j]);
}}}
cout<<"the solution is"<<endl;
for(i=0;i<n;i++)
{
x[i]=(a[i][n]/a[i][i]);
cout<<"x["<<i<<"]="<<x[i]<<endl;
}
getch();
}
```

QUIZ

Q.1)How Gauss Jorden method is different from Gauss Elimination Method?

Answer: Gauss Jorden method is similar method except that instead of first converting A into upper triangular form, it is directly converted into the unit matrix.

Q.2) What happens when A reduced to I?

Answer: As soon as A reduces to I , the other matrix represents A^{-1}

Q.3) What are the numerical method of gauss elimination and gauss Jordan method?

Answer: In Computer Programming, Algebra, Linear Algebra

Q.4)Expalin Gauss Jorden method.

Answer:Here two matrices A and I are written side by side and the same transformations are performed on both. As soon as A is reduced to I, the other matrix represents A^{-1}

Q.5)What is the Gauss Jordan method?

Answer: Gauss Jordan method is used to solve linear equation of the form Ax = b.

Program 7

OBJECTIVES: To integrate numerically using trapezoidal rule.

```
(Trapezoidal Rule).
 Composite Trapezoidal
     b
     /              h                              M-1
     | f(x)dx   =   - * [ f(A) - f(B)]  +  h * sum  f(x_k)
     /              2                              k=1
     a

 by sampling f(x) at the  M + 1  equally spaced points

 x_k = A + h*k  for  k = 0,1,2...,M.

 Notice that  x_0 = A  and  x_M = B.

--------------------------------------------------------------------------
----
*/

/* User has to supply a function named : ffunction
   An example is included in this program */

#include<stdio.h>
#include<stdlib.h>
#include<math.h>


/*  define prototype for USER-SUPPLIED function f(x)  */

    double ffunction(double x);


/*  EXAMPLE for "ffunction"   */

    double ffunction(double x)

    {
        return ( 1 / ( 1 + pow(x,2) ) );
    }

/* ------------------------------------------------------ */

/*  Main program for algorithm 7.1  */

    void main()

{
    int K;                    /* loop counter                  */
    int M;                    /* INPUT : number of subintervals */
```

```
    double A,B;                /* INPUT : boundaries of integral */
    double H;                  /* Subinterval width              */
    double SUM = 0;            /* approx. integral value         */
    double X;


    printf("-----------------------------------------------\n");
    printf("Please enter the boundaries of the integral [A,B]\n");
    printf("EXAMPLE: A = -1 and B = 1, so type: -1 1\n");
    scanf("%lf %lf",&A, &B);
    printf("The boundaries of the integral are: %lf %lf\n",A, B);
    printf("-----------------------------------------------\n");
    printf("Please enter the number of SUBINTERVALS.\n");
    scanf("%d",&M);

    printf("You say : %d subintervals.\n",M);

    H = (B - A)/M;              /*  Subinterval width             */

    for ( K = 1; K <= M-1; K++ ) {
         X = A + H*K;
         SUM = SUM + ffunction(X);
    }

    SUM = H * ( ffunction(A) + ffunction(B) + 2*SUM )/2;


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

    printf(" The approximate value of the integral of f(X)\n");
    printf(" on the interval %lf %lf\n", A,B);
    printf(" using %d subintervals computed using the
trapezoidal\n",M);
    printf(" rule is : %lf\n",SUM);

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

}   /* End of main program */
```

Quiz

Q.1)What is use of Trapezoidal rule?

Answer: Trapezoidal rule is used to solve numerical differentiation.

Q.2)What is numerical differentiation?

Answer: The process of evaluating a definite integral from a set of tabulated values of the integrand f(x) is called numerical integration. This process when applied to a function of a single variable, is known as quadrate.

Q.3) What is observation of Trapezoidal rule?

Answer: Obs. : The area of each strip(trapezium) is found separately. Then the are under the curve and ordinates at x_0 and x_0+nh is approximately equal to the sum of the areas of the n trapeziums.

Q.4) Write an equation of Trapezoidal rule.

Answer: $\int_{x0}^{x0+h} f(x)dx=h/2[(y_0+y_n)+2(y_1+y_2+..+y_{n-1})]$

Q.5) Which formula is used in Trapezoidal rule?

Answer: Newton-Cotes formula is used in Trapezoidal rule.

Program 8

OBJECTIVES: To Integrate Numerically Using Simpson's Rules.

(Code for SIMPSON'S 1/3 RULE)

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
  float x[10],y[10],sum=0,h,temp;
  int i,n,j,k=0;
  float fact(int);
  clrscr();
  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[i]);
  }
  h=x[1]-x[0];
  n=n-1;
  sum = sum + y[0];
  for(i=1;i<n;i++)
  {
    if(k==0)
    {
     sum = sum + 4 * y[i];
     k=1;
    }
    else
    {
     sum = sum + 2 * y[i];
     k=0;
    }
   }
   sum = sum + y[i];
   sum = sum * (h/3);
   printf("\n\n  I = %f  ",sum);
getch();
}

/*
_____________________________________________

         OUT PUT
_____________________________________________


how many record you will be enter: 5
```

```
enter the value of x0: 0

enter the value of f(x0): 1

enter the value of x1: 0.25

enter the value of f(x1): 0.8

enter the value of x2: 0.5

enter the value of f(x2): 0.6667

enter the value of x3: 0.75

enter the value of f(x3): 0.5714

enter the value of x4: 1

enter the value of f(x4): 0.5

       I = 0.693250

*/
```

QUIZ

Q.1) What is use of Simpsons one-third rule?

Answer: Simpsons one-third rule is used to find numerical integration of given equation.

Q.2) List Simpsons rule.

Answer: Simpsons one-third rule and Simpsons three eight rule.

Q.3) Write an observation about Simpsons one third rule.

Answer: While applying Simpsons one third rule, the given integral must be divided into even number of equal subintervals, since we find the area of two strips at a time.

Q.4)Write down Simpsons one-third rule.

Answer: $\int_{x0}^{x0+h} f(x)dx=h/3[(y_0+y_n)+4(y_1+y_3+..+y_{n-1})+ (y_2+y_{n-2})]$

Q.5)Simpsons one-third rule is applied on which point of parabola?

Answer: Putting n=2 in Newtons quadrate formula and taking the curve through (x_0, y_0), (x_1, y1), (x2, y2) as a parabola.

Program 9

OBJECTIVES: To find the largest eigen value of matrix by power method.

```
 /*To compute the dominant value  Lambda_1  and its associated
 eigenvector  V_1  for the  n x n  matrix  A. It is assumed
 that the  n  eigenvalues have the dominance property

 |Lambda_1| > |Lambda_2| >= |Lambda_3| >= ... >= |Lambda_n| > 0

-------------------------------------------------------------------------
----
*/
```

Source code:

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

#define MaxOrder 50

#define MAX(a,b) a > b ? a : b

/* ------------------------------------------------------ */

/*  Main program for algorithm 11.1  */

void main(void)

{
    int i, j;                 /* Loop counter                  */
    int N;                    /* Order of Matrix               */
    double Epsilon = 1E-7;    /* Tolerance                     */
    double Max = 100;         /* Maximum number of iterations  */
    double X[MaxOrder], Y[MaxOrder];
    double A[MaxOrder][MaxOrder]; /* Matrix                    */
    double C1, DC, DV, Lambda = 0;
    int Count = 0, Iterating = 1;
    double Sum, Err = 1;
    double MaxElement;


    printf("--------------Power Method---------------\n");
    printf("--------- Example 11.5 on page 550 ------------\n");
    printf("-----------------------------------------------\n");

    printf("Please enter order of Matrix  A ( < %d !)\n", MaxOrder +
1);
    scanf("%d",&N);
    if( N > MaxOrder )
    {
```

```
    printf(" Number of steps must be less than %d\n",MaxOrder+1);
    printf(" Terminating. Sorry\n");
    exit(0);
    }
    printf("Please enter elements of matrix  A  row by row:\n");

    for ( i = 0; i < N; i++)
    {
        for ( j = 0; j < N; j++)
        {
            printf(" Enter Element No. %d of row %d\n", j+1, i+1);
            scanf("%lf", &A[i][j]);
            printf("You entered A[%d][%d]= %lf\n", i+1, j+1, A[i][j] );
            printf("-----------------------------------------------
\n");
        }
    }

    printf("\n");

    /* Initialize vector X */

    for ( i = 0; i < N; i++)   X[i] = 1.0;

    while( (Count <= Max) && (Iterating == 1) )
    {

    /* Perform Matrix-Vector multiplication */

    for ( i = 0; i < N; i++ )
    {
        Y[i] = 0;
        for ( j = 0; j < N; j++ ) Y[i] += A[i][j] * X[j];
    }

    /* Find largest element of vector Y */
    /* Do what function MaxElement(X,N) in the book does */

    MaxElement = 0;
    for ( j = 0; j < N; j++ )
    {
        if( fabs(Y[j]) > fabs(MaxElement) ) MaxElement = Y[j];
    }

    C1 = MaxElement;

    DC = fabs(Lambda - C1);

    for ( i = 0; i < N; i++) Y[i] *= 1.0/C1;

    /* Do what function DIST(X,Y,N) in the book does */

    Sum = 0;
    for ( i = 0; i < N; i++) Sum += pow( ( Y[i] - X[i] ), 2.0);
    DV  = sqrt(Sum);
    Err = MAX(DC,DV);
```

```
    /* Update vector X and scalar Lambda */

    for ( i = 0; i < N; i++) X[i] = Y[i];
    Lambda = C1;

    Iterating = 0;

    if( Err > Epsilon) Iterating = 1;

    Count++;

    } /* End of while loop */


    /* Output vector X and scalar Lambda */

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

    for ( j = 0; j < N; j++) printf("X[%d] = %lf\n", j, X[j]);

    printf("---------------------------------------\n");
    printf("Lambda = %lf\n", Lambda);


}   /* End of main program */
```

Quiz

Q.1)What is the use of Power method?

Answer: In many engineering application, it is required to compute the numerically largest eigen value and the corresponding eigen vector. Power method is used to find it.

Q.2)What is the observation of Power method?

Answer: OBS: Rewritting $AX=YX$ as $A^{-1}AX=Y\ A^{-1}X$

We have $A^{-1}X=1/YX$

If we use this equation, then power method yields the smallest eigen values.

Q.3)Does power method require normalization?

Answer: Yes

Q.4)When does power method yields the smallest eigen value?

Answer: We have $A^{-1}X=1/YX$

If we use this equation, then power method yields the smallest eigen values.

Q.5)Write down two properties of Eigen values.

Answer:1)The sum of eigen values of matrix A, is the sum of the elements of its principal diagonal.

2)Any similarity transformation applied to a matrix leaves its eigen values unchanged.

Program 10

OBJECTIVES: TO FIND NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS BY EULER'S METHOD.

(Code for Program of EULER'S METHOD)

```
#include<stdio.h>
#include <math.h>
#include<conio.h>
//dy/dx = xy#define F(x,y)  (x)*(y)
void main()
{
  double y1,y2,x1,a,n,h;
  int j;
  clrscr();
  printf("\nEnter the value of range: ");
  scanf("%lf %lf",&a,&n);
  printf("\nEnter the value of y1: ");
  scanf("%lf",&y1);
  printf("\n\nEnter the h: ");
  scanf("%lf",&h);
  printf("\n\n  y1 = %.3lf ",y1);
  for(x1=a,j=2; x1<=n+h; x1=x1+h,j++)
  {
   y2= y1 + h * F(x1,y1);
   printf("\n\n  x = %.3lf => y%d = %.3lf ",x1,j,y2);
   y1=y2;
  }
getch();
}

/*
 OUT PUT
---------


Enter the value of range: 1 1.5

Enter the value of y1: 5


Enter the h: 0.1


  y1 = 5.000

  x = 1.000 => y2 = 5.500

  x = 1.100 => y3 = 6.105

  x = 1.200 => y4 = 6.838

  x = 1.300 => y5 = 7.726
```

```
  x = 1.400 => y6 = 8.808

  x = 1.500 => y7 = 10.129


*/
```

QUIZ

Q.1)What is the use of Eulers method?
Answer: Euler's method is used to find out the solution of ordinary differential equation.

Q.2)What happens when h is small?
Answer:When the h is smaal the error is bound to be quite significant.

Q.3)What are the disadvantages of Eulers method?
Answer: This method is very slow.

Q.4)How the equation is solved using Eulers method?
Answer: In Eulers method we approximate the curve of the solution by the tangent in each interval.

Q.5)Write the equation for Eulers method for findind approximate solution?

Answer: $y_n = y_{n-1} + hf(x_0 + \bar{} n\text{-}1h, y_{n-1})$

Program 11

OBJECTIVES: To Find Numerical Solution Of Ordinary Differential Equations By Runge- Kutta Method.

```
/* Runge Kutta for a set of first order differential equations */

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

#define N 2 /* number of first order equations */
#define dist 0.1 /* stepsize in t*/
#define MAX 30.0 /* max for t */

FILE *output; /* internal filename */

void runge4(double x, double y[], double step); /* Runge-Kutta function */
double f(double x, double y[], int i); /* function for derivatives */

void main()
{
double t, y[N];
int j;

output=fopen("osc.dat", "w"); /* external filename */

y[0]=1.0; /* initial position */
y[1]=0.0; /* initial velocity */

fprintf(output, "0\t%f\n", y[0]);

for (j=1; j*dist<=MAX ;j++) /* time loop */
{
t=j*dist;
runge4(t, y, dist);
fprintf(output, "%f\t%f\n", t, y[0]);
}

fclose(output);
}

void runge4(double x, double y[], double step)
{
double h=step/2.0, /* the midpoint */
t1[N], t2[N], t3[N], /* temporary storage arrays */
```

```
k1[N], k2[N], k3[N],k4[N]; /* for Runge-Kutta */
int i;

for (i=0;i<N;i++)
{
t1[i]=y[i]+0.5*(k1[i]=step*f(x,y,i));
}

for (i=0;i<N;i++)
{
t2[i]=y[i]+0.5*(k2[i]=step*f(x+h, t1, i));
}

for (i=0;i<N;i++)
{
t3[i]=y[i]+ (k3[i]=step*f(x+h, t2, i));
}

for (i=0;i<N;i++)
{
k4[i]= step*f(x+step, t3, i);
}

for (i=0;i<N;i++)
{
y[i]+=(k1[i]+2*k2[i]+2*k3[i]+k4[i])/6.0;
}
}

double f(double x, double y[], int i)
{

if (i==0)
x=y[1]; /* derivative of first equation */
if (i==1)
x= -0.2*y[1]-y[0]; /* derivative of second equation */

return x;
}
```

QUIZ

Q.1)What is observation of RK method?

Answer: Here the operation is identical whether differential equation is linear or non-linear.

Q.2)Write down RK rule.

Answer: $k=1/6(k_1+2k_2+2k_3+k_4)$

Q.3)What is working rule of RK method?

Answer: Working rule for finding k of y corresponding to an increment h of x by RK method from

$dy/dx=f(x, y)$, $y(x_0)=y_0$

is as follows:

Calculate successively $k_1=hf(x_0, y_0)$, $k_2=hf(x_0+1/2h , y_0 +1/2k_1)$

$k_3= hf(x_0+1/2h , y_0 +1/2k_2)$ and $k_4= hf(x_0+h , y_0 +k_3)$

Finally compute $k=1/6(k_1+2k_2+2k_3+k_4)$

Q.4)What is the use of RK method?

Answer: RK method is used to solve differential equation.

Q.5)What are the advantages of RK method?

Answer:1)RK method do not require the calculation of higher order derivatives.

2) This method gives greater accuracy.

3)It requires only the functions values at some selected point.

Program 12

OBJECTIVES: To find the numerical solution of differential equation using mline method.

```
(Milne-Simpson Method)

 To approximate the solution of the initial value problem  y' = f(t,y)
 with  y(a) = y_0  over [a,b] by using the predictor

                          4h
 p_(k+1)  =  y_(k-3)  +   --- [2*f_(k-2) - f_(k-1) + 2*f_k]
                           3

 and the corrector :

                           h
 y_(k+1)  =  y_(k-1)  +   --- [f_(K-1) + 4*f_k + f_(K+1)].
                           3

 User has to supply  functions named : ffunction
 An example is included in this program.

-------------------------------------------------------------------------
----
*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define MAX 500


/*  define prototype for USER-SUPPLIED function f(x)  */

    double ffunction(double t, double y);

/*  EXAMPLE for "ffunction"   */

    double ffunction(double t, double y)

    {
        return ( (t - y) / 2.0 );
    }

/* ------------------------------------------------------- */

/*  Main program for algorithm 9.7  */

void main(void)

{
```

```
    int I, K;                  /* Loop counter                    */
    int N;                     /* Number of steps > 3             */
    double A, B, Y[MAX];       /* Endpoints and initial value     */
    double H;                  /* Compute the step size           */
    double T[MAX];
    double F1, F2, F3, F4;     /* Function values                 */
    double Hmin, Hmax;         /* Minimum and maximum step size   */
    double Pold, Yold;         /* Predictor and Corrector         */
    double Pmod, Pnew;         /* Modifier                        */


    printf("------------Milne-Simpson Method----------------\n");
    printf("--------- Example 9.13 on page 468 -------------\n");
    printf("-----------------------------------------------\n");
    printf("Please enter endpoints of the interval [A,B]:\n");
    printf("For used Example type  : 0, 3.0\n");
    scanf("%lf, %lf", &A, &B);
    printf("You entered [%lf, %lf]\n", A, B);
    printf("---------------------------------------\n");
    printf("Please enter number of steps: ( > 3 and < %d !)\n", MAX+1);
    scanf("%d",&N);
    if( (N > MAX) || (N < 4) )
    {
    printf(" Number of steps must be greater than 3 and less than
%d\n",MAX+1);
    printf(" Terminating. Sorry\n");
    exit(0);
    }
    printf("---------------------------------------\n");

    printf("You need all together FOUR initial values. You can\n");
    printf("use the Runge-Kutta method to compute the 2nd, 3rd and\n");
    printf("4th from the 1st one.\n");
    printf("Please enter initial values Y[0], Y[1], Y[2], Y[3] :\n");
    printf("Example 9.13 page 468:  1, 0.94323919, 0.89749071,
0.86208736\n");
    scanf("%lf, %lf, %lf, %lf", &Y[0], &Y[1], &Y[2], &Y[3]);
    printf("You entered Y[0] = %lf\n", Y[0]);
    printf("You entered Y[1] = %lf\n", Y[1]);
    printf("You entered Y[2] = %lf\n", Y[2]);
    printf("You entered Y[3] = %lf\n", Y[3]);

    /* Compute the step size and initialize */

    H    = (B - A) / N;
    T[0] = A;

    for( K = 1; K <= 3; K++) T[K] = A + K * H;

    F1 = ffunction(T[1],Y[1]);
    F2 = ffunction(T[2],Y[2]);
    F3 = ffunction(T[3],Y[3]);

    Pold = 0;
    Yold = 0;

    for( K = 3; K <= N-1; K++)
```

```
    {
        /* Milne Predictor */
        Pnew = Y[K-3] + 4.0 * H * (2.0*F1 - F2 + 2.0*F3) / 3.0;
        Pmod = Pnew + 28.0 * (Yold - Pold) / 29.0;
        /* Next mesh point */
        T[K+1] = A + H * (K+1);
        /* Evaluate f(t,y) */
        F4 = ffunction(T[K+1],Pmod);
        /* Corrector */
        Y[K+1] = Y[K-1] + H * (F2 + 4.0 * F3 + F4) / 3.0;
        /* Update the values */
        F1 = F2;
        F2 = F3;
        F3 = ffunction(T[K+1], Y[K+1]);
    }

    /* Output */

    for ( K = 0; K <= N; K++)
    {
       printf("K = %d, T[K] = %lf, Y[K] = %lf\n", K, T[K], Y[K]);
    }

}   /* End of main program */
```

QUIZ

Q.1)How to get greater accuracy in Mline method?

Answer:To insure greater accuracy, we must first improve the accuracy of the starting value and then subdivide the interval.

Q.2)What is the use of Mline method?

Answer:Mline method used to solve the differential equation over an interval.

Q.3)What is the equation of predicator in Mline method

Answer: $Y_4^{(p)} = y_0 + 4h/3(2\ f_1 - f_2 + 2f_3)$

Q.4) What is the equation of a corrector in Mline method?

Answer: $Y_4^{(c)} = y_2 + h/3(\ f_2 - 4f_3 + f_4)$

Q.5) What is the better approximation to the value of y_5?

Answer: $Y_5^{(c)} = y_3 + h/3(\ f_3 + 4f_4 + f_5)$

Program 13

OBJECTIVES: To find the numerical solution of differential equation using laplace equation.

```
(Dirichlet Method for Laplace's Equation)

                                                    2
 To approximate the solution of u_(xx)(x,y) + u_(yy) (x,y)  =  0

 over R = {(x,y): 0 <= x <= a,  0 <= y <= b} with

 u(x,0) = f_1(x),  u(x,b) = f_2(x)  for  0 <= x <= a  and

 u(0,y) = f_3(y),  u(a,y) = f_4(y)  for  0 <= y <= b.

 It is assumed that  Delta(x) = Delta(y) = h  and that integers

 n  and  m  exist so that  a = nh  and  b = mh.

 User has to supply functions  F1, F2, F3, F4  with gridpoints.
 as arguments. An example is given in this program.

-----------------------------------------------------------------------
---*/
```

Source code:

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

#define Max 50

/* Global Variables */

double H;               /* Step size  */

/* Prototypes */

double F1i(int i);
double F2i(int i);
double F3i(int i);
double F4i(int i);

/* Grid function for amplitude */

double F1i(int i)
{
    extern double H;
    double arg;
```

```
    arg = H * ( i - 1.0 );

    if( (arg >= 0) && (arg <= 4.0) )  return ( 180.0 );
    else
    {
        printf(" F1i() :Argument not in range ! Exiting !\n");
        printf(" arg  : %lf\n", arg);
        exit(0);
    }
}

double F2i(int i)
{
    extern double H;
    double arg;

    arg = H * ( i - 1.0 );

    if( (arg >= 0) && (arg <= 4) ) return ( 20.0 );
    else
    {
        printf(" F2i() :Argument not in range ! Exiting !\n");
        printf(" arg  : %lf\n", arg);
        exit(0);
    }
}

double F3i(int i)
{
    extern double H;
    double arg;

    arg = H * ( i - 1.0 );

    if( (arg >= 0) && (arg <= 4) ) return ( 80.0 );
    else
    {
        printf(" F3i() :Argument not in range ! Exiting !\n");
        printf(" arg  : %lf\n", arg);
        exit(0);
    }
}

double F4i(int i)
{
    extern double H;
    double arg;

    arg = H * ( i - 1.0 );

    if( (arg >= 0) && (arg <= 4) ) return ( 0.0 );
    else
    {
        printf(" F4i() :Argument not in range ! Exiting !\n");
        printf(" arg  : %lf\n", arg);
        exit(0);
    }
```

```
}


/* ------------------------------------------------------ */



void main(void)

{
    int i, j;               /* Loop Counters
*/
    double A, B;            /* INPUT : Width and height of Rectangle
*/
    double Ave;             /* INPUT : Initial approximation
*/
    int N, M;               /* dimensions of the grid
*/
    double U[Max][Max];     /* Grid-Amplitudes
*/
    int Count;              /* counter for while loop
*/
    double Tol;             /* stop criteria
*/
    double w, Pi;
    double H;               /* Grid spacing
*/
    double Relax, temp;

    printf("------------------------------------------------------
\n");
    printf("------- Dirichlet Method for Laplace's Equation -------
\n");
    printf("--------- Try Example 10.6 on page 528/529-------------
\n");
    printf("------------------------------------------------------
\n");

    printf("Please enter A (interval boundary of x)\n");
    printf("For present example type  4\n");
    scanf("%lf",&A);
    printf("Please enter B (interval boundary of y)\n");
    printf("For present example type  4\n");
    scanf("%lf",&B);
    printf("You entered A = %lf and B = %lf \n", A, B);
    printf("-----------------------------------------------\n");
    printf("Please enter Grid-Spacing H\n");
    printf("For present example type  0.5\n");
    scanf("%lf",&H);
    printf("You entered H = %lf\n", H);
    printf("-----------------------------------------------\n");
    printf("\n");
    printf("Please enter dimension of grid in x-direction :\n");
    printf("For present example type  9\n");
    scanf("%d",&N);
    printf("Please enter dimension of grid in y-direction :\n");
```

```
printf("For present example type  9\n");
scanf("%d",&M);
printf("You entered N = %d and M = %d\n", N, M);
printf("-----------------------------------------------\n");
printf("\n");
printf("Please enter the initial approximation :\n");
printf("For present example type  70\n");
scanf("%lf",&Ave);
printf("You entered = %lf\n", Ave);
printf("-----------------------------------------------\n");
printf("\n");

/* Compute step sizes */

Pi = 3.1415926535;

/* Only to make the 0-indexes save we initialize them to zero.
   This is only because in C fields begin with index 0.
   In this program we ignore the index 0. */

for ( i = 0; i < N; i++ ) U[i][0] = 0.0;
for ( i = 1; i < M; i++ ) U[0][i] = 0.0;

/* Initialize starting values at the interior points */

for ( i = 2; i <= N-1; i++ )
{
    for ( j = 2; j <= M-1; j++ )   U[i][j] = Ave;
}

/* Store boundary values in the solution matrix */

for ( j = 1; j <= M ; j++ )
{
    U[1][j] = F3i(j);
    U[N][j] = F4i(j);
}

for ( i = 1; i <= N ; i++ )
{
    U[i][1] = F1i(i);
    U[i][M] = F2i(i);
}

/* The SQR parameter */

temp =  cos( Pi/(N-1) ) + cos( Pi/(M-1) );

w = 4.0 / ( 2.0 + sqrt( 4.0 - temp * temp ) );

/* Initialize the loop control parameters */

Tol   = 1.0;
Count = 0.0;

while ( (Tol > 0.001) && (Count <= 140) )
{
```

```
       Tol = 0.0;
       for ( j = 2; j <= M - 1; j++ )
       {
           for ( i = 2; i <= N - 1; i++ )
           {
             Relax = w * ( U[i][j+1] + U[i][j-1] + U[i+1][j] +
                           U[i-1][j] - 4.0 * U[i][j] ) / 4.0;
             U[i][j] += Relax;
             if( fabs(Relax) > Tol ) Tol = fabs(Relax);
           }
           Count++;
       }
   }

   /* Output the solution */

   for ( j = 1; j <= M; j++ )
   {
       for ( i = 1; i <= N; i++ ) printf("%8.4lf ", U[i][j]);
       printf("\n");
   }

}   /* End of main program */
```

QUIZ

Q.1)What is the use of Laplace method?
Answer: Laplace method is used to find out the solution of partial differential equation.

Q.2)What is the boundary of Rectangular region R.

Answer: u(x,y) is the boundary of rectangular region R.

Q.3)What is the first step in Laplace equation?

Answer:Divide the given into network of square mesh of side h.

Q.4)What is the four mesh point?

Answer:The four mesh point is the average of its values at four neighbouring point to the left, right, above and below.

Q.5)How can we find the interior points?

Answer: Interior points are computed by the standard five point formula.

Program 14

OBJECTIVES: To find the numerical solution of differential equation using wave equation

```
 (Finite-Difference Solution for the Wave Equation)

                                                    2
 To approximate the solution of u_(tt)(x,t)  =  c u_(xx)(x,t)

 over  R = {(x,t): 0 <= x <= a, 0 <= t <= b}  with

 u(0,t) = 0,       u(a,t) = 0      for  0 <= t <= b and

 u(x,0) = f(x),  u_t(x,0) = g(x)   for  0 <= x <= a.

 User has to supply function  Fi(gridpoint)  and
 Gi(gridpoint):  boundary functions at the grid
 points. An examples is implemented in this program.
```

Souce code:

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

#define Max 50

/* Global Variables */

double H;                /* Step size  */

/* Prototypes */

double Fi(int i);
double Gi(int i);

/* Grid function for amplitude */

double Fi(int i)
{
    extern double H;
    double arg;

    arg = H * (i - 1);

    if( (arg >= 0) && (arg <= 1.0) )  return ( sin ( 3.1415926 * arg)
);
    else
    {
        printf(" Fi() :Argument not in range ! Exiting !\n");
```

```
        printf(" arg  : %lf\n", arg);
        exit(0);
    }
}

/* Grid function for velocity */

double Gi(int i)
{
    extern double H;
    double arg;

    arg = H * (i - 1);

    if( (arg >= 0) && (arg <= 1) ) return ( 0.0 );
    else
    {
        printf(" Gi() :Argument not in range ! Exiting !\n");
        printf(" arg  : %lf\n", arg);
        exit(0);
    }
}

/* ------------------------------------------------------ */

/*  Main program for algorithm 10-1  */

void main(void)

{
    int i, j;               /* Loop Counters
*/
    double A, B;            /* INPUT :Width and height of Rectangle
*/
    double C;               /* INPUT : Wave equation const.
*/
    int N, M;               /* Dimensions of the grid
*/
    double K, R, R2, R22, S1, S2;
    double U[Max][Max];     /* Grid-Amplitudes
*/

    printf("-- Finite-Difference Solution to the Wave Equation ----
\n");
    printf("--------- Example Problem No 4 on page 508 -----------
\n");
    printf("------------------------------------------------------
\n");

    printf("Please enter A (interval boundary of x)\n");
    printf("For present example type : 1.0\n");
    scanf("%lf",&A);
    printf("Please enter B (interval boundary of t)\n");
    printf("For present example type : 0.5\n");
    scanf("%lf",&B);
    printf("You entered A = %lf and B = %lf \n", A, B);
```

```
    printf("-----------------------------------------------------
\n");
    printf("\n");
    printf("Please enter dimension of grid in x-direction :\n");
    printf("For present example type : 6\n");
    scanf("%d",&N);
    printf("Please enter dimension of grid in y-direction :\n");
    printf("For present example type : 6\n");
    scanf("%d",&M);
    printf("You entered N = %d and M = %d\n", N, M);
    printf("-----------------------------------------------------
\n");
    printf("\n");
    printf("Please enter the wave equation constant C :\n");
    printf("For present example type : 2\n");
    scanf("%lf",&C);
    printf("You entered C = %lf\n", C);
    printf("-------------------------------------------------\n");
    printf("\n");

    /* Compute step sizes */

    H   = A / ( N - 1 );
    K   = B / ( M - 1 );
    R   = C * K / H;
    R2  = R * R;
    R22 = R * R / 2.0;
    S1  = 1.0 - R * R;
    S2  = 2.0 - 2.0 * R * R;

    /* Boundary conditions */

    for ( j = 1; j <= M; j++ )
    {
        U[1][j] = 0;
        U[N][j] = 0;
    }

    /* First and second rows */

    for ( i = 2; i <= N - 1 ; i++ )
    {
        U[i][1] = Fi(i);
        U[i][2] = S1 * Fi(i) + K * Gi(i) + R22 * ( Fi(i+1) + Fi(i-1) );
    }

    /* Generate new waves */

    for ( j = 3; j <= M; j++ )
    {
        for ( i = 2; i <= N - 1; i++ )
        {
            U[i][j] = S2 * U[i][j-1] + R2 * ( U[i-1][j-1] + U[i+1][j-1]
)
                        - U[i][j-2];
        }
    }
```

```
    /* Output the solution */

    for ( j = 1; j <= M; j++ )
    {
        printf("%8.6lf ", K * (j - 1));

        for ( i = 1; i <= N; i++ ) printf(" %8.6lf ", U[i][j]);

        printf("\n");
    }

}   /* End of main program */
```

Quiz

Q.1)What is the ue of wave equation?
Wave equation is used to find out thepartial solution od differential equation.

Q.2)What happen when$\alpha=1/c$?
Answer:The solution is stable.

Q.3) What happen when$\alpha<1/c$?
Answer: The solution is stable but inaccurate.

Q.4) What happen when$\alpha>1/c$?
Answer: The solution is unstable.

Q.5)Give an example of hyperbolic partial differential equation.
Answer: Wave equation is the simplest example.

Program 15

OBJECTIVES: To find the numerical solution of differential equation using heat equation.

Source code:

```
(Forward-Difference Method for the Heat Equation).
-------------------------------------------------------------------------
----
*/
/*
-------------------------------------------------------------------------
----

 (Forward-Difference Method for the Heat Equation)

                                                    2
 To approximate the solution of u_(t)(x,t)  =  c u_(x)(x,t)

 over R = {(x,t): 0 <= x <= a, 0 <= t <= b} with

 u(x,0) = f(x)  for  0 <= x <= a and

 u(0,t) = c_1,  u(a,t) = c_2  for  0 <= t <= b.

 User has to supply function Fi(gridpoint).
 An example is implemented in this program.

-------------------------------------------------------------------------
----
*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define Max 50

/* Global Variables */

double H;                /* Step size  */

/* Prototypes */

double Fi(int i);

/* Grid function for amplitude */

double Fi(int i)
{
```

```
    extern double H;
    double arg;

    arg = H * (i - 1);

    if( (arg >= 0) && (arg <= 1.0) ) return (4.0 * arg - 4.0 * arg *
arg);
    else
    {
        printf(" Fi() :Argument not in range ! Exiting !\n");
        printf(" arg  : %lf\n", arg);
        exit(0);
    }
}

/* ---------------------------------------------------------- */

/*  Main program for algorithm 10-2  */

/*  Remember : in C the fields start with index 0.
    In this program this field is ignored and when you
    change the program be aware of the index 0 in the U[Max][Max]
    array ! Here - as said before - it is ignored, so don't
    access it if you do not initialize it. */

void main(void)

{
    int i, j;               /* Loop Counters
*/
    double A, B;            /* INPUT :Width and height of Rectangle
*/
    double C;               /* INPUT : Wave equation const.
*/
    int N, M;               /* Dimensions of the grid
*/
    double K, S, C1, C2, R;
    double U[Max][Max];     /* Grid-Amplitudes
*/

    printf("-- Forward-Difference Method for the Heat Equation ----
\n");
    printf("--------- Example 10.3 on page 511 --------------------
\n");
    printf("-----------------------------------------------------
\n");

    printf("Please enter A (interval boundary of x)\n");
    printf("For present example type : 1.0\n");
    scanf("%lf",&A);
    printf("Please enter B (interval boundary of t)\n");
    printf("For present example type : 0.2\n");
    scanf("%lf",&B);
    printf("You entered A = %lf and B = %lf \n", A, B);
    printf("-----------------------------------------------------
\n");
    printf("\n");
```

```
    printf("Please enter dimension of grid in x-direction ( < 50
):\n");
    printf("For present example type : 6\n");
    scanf("%d",&N);
    if ( N >= 50 )
    {
       printf("MUST BE SMALLER THAN 50. ... terminating.\n");
       exit(0);
    }
    printf("Please enter dimension of grid in y-direction ( < 50
):\n");
    printf("For present example type : 11\n");
    scanf("%d",&M);
    if ( M >= 50 )
    {
       printf("MUST BE SMALLER THAN 50. ... terminating.\n");
       exit(0);
    }
    printf("You entered N = %d and M = %d\n", N, M);
    printf("-------------------------------------------------------
\n");
    printf("\n");
    printf("Please enter the heat-equation constant C :\n");
    printf("For present example type : 1\n");
    scanf("%lf",&C);
    printf("You entered C = %lf\n", C);
    printf("-------------------------------------------------\n");
    printf("\n");
    printf("Please enter constant C1 = u(0,t) :\n");
    printf("For present example type : 0\n");
    scanf("%lf",&C1);
    printf("You entered C1 = %lf\n", C1);
    printf("-------------------------------------------------\n");
    printf("\n");
    printf("Please enter constant C2 = u(A,t) :\n");
    printf("For present example type : 0\n");
    scanf("%lf",&C2);
    printf("You entered C2 = %lf\n", C2);
    printf("-------------------------------------------------\n");
    printf("\n");


    /* Compute step sizes */

    H   = A / ( N - 1 );
    K   = B / ( M - 1 );
    R   = C * C * K / ( H * H );
    S   = 1.0 - 2.0 * R;

    /* Boundary conditions */

    for ( j = 1; j <= M; j++ )
    {
        U[1][j] = C1;
        U[N][j] = C2;
    }
```

```
    /* First row */

    for ( i = 2; i <= N - 1 ; i++ ) U[i][1] = Fi(i);

    /* Generate new waves */

    for ( j = 2; j <= M; j++ )
    {
        for ( i = 2; i <= N - 1; i++ )
        {
            U[i][j] = S * U[i][j-1] + R * ( U[i-1][j-1] + U[i+1][j-1]
);
        }
    }

    /* Output the solution */

    printf("The grid amplitudes look like this :\n");
    printf("\n");

    for ( j = 1; j <= M; j++ )
    {
        printf("%8.6lf ", K * (j - 1));

        for ( i = 1; i <= N; i++ ) printf(" %8.6lf ", U[i][j]);

        printf("\n");
    }

}   /* End of main program */
```

QUIZ

Q.1)What is the use of heat method?
Answer: Heat method is used to find out solution of partial differential equation.

Q.2)What are the different method for heat equation?
Answer:

1)Schmidt method
2)Crank –Nicolson method
3)Iterative method of solution.

Q.3)Which method is used for non restrict α?
Answer: Crank –Nicolson method

Q.4)What is implicit scheme?
Answer: A method in which the calculation of an unknowm mesh value necessitates the solution of a set of simultaneous equation is known as an implicit scheme.

Q.5)Which is 3-level method?
Answer:Richardson scheme is 3-level method.

REFRENCES:

1. Numerical methods by B.S.Grewal

2. Numericalk method :E. Balagurusamy T.M.H