



KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

Semester – I

18MMP111

NUMERICAL ANALYSIS - PRACTICAL

4H – 2C

Instruction Hours / week: L: 0 T: 0 P: 2

Marks: Internal: 40

External: 60 Total: 100

End Semester Exam: 3 Hours

Course Objectives

This course enables the students to learn

- In-depth understanding of functional, logic, and programming paradigms.
- How to implement several programs in languages other than the one emphasized in the core curriculum.

Course Outcomes (COs)

On successful completion of this course, the student will be able to

1. Know the concepts for problem solving.
2. Acquire new knowledge in computing, including the ability to learn about new ideas and advances, techniques, tools, and languages, and to use them effectively; and to be motivated to engage in life-long learning
3. Comprehend important issues related to the development of computer-based systems in a professional context using a well-defined process.

List of Practical

1. Solution of non-linear equation-Bairstow's method for quadratic factors.
2. Solution of simultaneous equations-Gauss Elimination.
3. Solution of simultaneous equations-Gauss Jordan.
4. Solution of simultaneous equations-Gauss Jacobi.
5. Solution of simultaneous equations-Gauss Seidal.
6. Solution of simultaneous equations-Triangularisation.
7. Numerical integration-Trapezoidal rule.
8. Numerical integration-Simpson's rules.
9. Solution for ordinary differential equation-Euler method.

10. Solution for ordinary differential equation- Runge Kutta Second order.

11. Solution for parabolic equation - Explicit method.

12. Solution for parabolic equation - The Crank Nicolson method.

EX:NO: 1	BAIRSTOW'S METHOD FOR QUADRATIC FACTORS
----------	--

AIM:

To find the quadratic factor of a biquadratic expression by using Bairstow's method using C program.

ALGORITHM:

STEP 1: Start the process.

STEP 2: Declare the variable i,n as int data type and a[10],
b[10],c[10],p,q,d,dp,dq as float data type.

STEP 3: Get the coefficient in array by for loop using scanf statement.

STEP 4: Get the values of p and q using scanf statement.

STEP 5: By using Read method initialize the value,
b[0]=a[0];c[0]=b[0];b[1]=a[1]-p*b[0];c[1]=b[1]-p*c[0];

STEP 6: Using for loop (i=2; i<n; i++)and using the formula as
b[i]=a[i]-p*b[i-1]-q*b[i-2]

c[i]=b[i]-p*c[i-1]-q*c[i-2];

STEP 7: To get the p and q values compute the following formula as

b[n]=a[n]-p*b[n-1]-q*b[n-2];

d=c[n-2]*c[n-2]-c[n-3]*(c[n-1]-b[n-1]);

dp=(b[n-1]*c[n-2]-b[n]*c[n-3])/d;

dq=(b[n]*c[n-2]-b[n-2]*(c[n-1]-b[n-1]))/d;

p=p+dp,q=q+dq.

STEP 8: using if loop check the condition.

`((fabs(dp)<pow(10,-6))&&(fabs(dq)<pow(10,-6))),`

If this condition is true print p and q values, otherwise goto read.

STEP 9: Again check with it condition with,

`(p>0)&&(q>0),(p<0)&&(q>0)`

`(p>0)&&(q<0),(p>0)&&(q>0)`

If any one of these condition is true print p and q value.

STEP 10: stop the process.

*/*Bairstow's method*/*

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
Void main()
{
int i,n;
float a[10],b[10],c[10],p,q,d,dp,dq;
clrscr();
n=4;
printf("\n Enter the coefficients:\n");
for(i=0;i<n+1;i++)
scanf("%f",&a[i]);
printf("\n Enter the values of p and q:\n");
scanf("\n%f%f",&p,&q);
read:
b[0]=a[0];
c[0]=b[0];
b[0]=a[1]-p*b[0];
c[1]=b[1]-p*c[0];
for(i=2;i<n;i++)
{
b[i]=a[i]-p*b[i-1]-q*b[i-2];
```

```

c[0]= b[i]-p*c[i-1]-q*c[i-2];
}
b[n]=a[n]-p*b[n-1]-q*b[n-2];
d=c[n-2]*c[n-2]-c[n-3]*(c[n-1]-b[n-1]);
dp=(b[n-1]*c[n-2]-b[n]*c[n-3])/d;
dq=(b[n]*c[n-2]-b[n-1]*(c[n-1]-b[n-1]))/d;
p=q+dp;
q=q+dp;
if((fabs(dp)<pow(10,-6))&&(fabs(dq)<pow(10,-6)))
printf("\n the values of p and q are:\n%f\t%f\n",p,q);
else
goto read;
printf("\n the quadratic factor is:\n");
if(p>0)&&(q<0)
printf("x^2+%fx+%f",p,q);
else if (p<0)&&(q>0)
printf("x^2+%fx+%f",p,q);
else if(p>0)&&(q<0)
printf("x^2+%fx+%f",p,q);
else
printf("x^2+%fx+%f",p,q);
getch();
}

```

OUTPUT:

Enter the coefficients

1 -3 -4 -2 8

Enter the value of p and q

0.95 1.05

The value of p and q are:

2.000000 2.000000

The quadratic factor is:

$X^2 + 2.000000x + 2.000000$.

RESULT:

The above program has been executed successfully and the output is verified.

EX.NO:2	GAUSS ELIMINATION METHOD
---------	--------------------------

AIM:

To solve the system of linear algebraic equation by Gauss elimination method by using c program.

ALGORITHM:

STEP 1: Start the process.

STEP 2: Declare the variable in a float data type.

STEP 3: Using scanf statement obtain the value of n.

STEP 4: Read the RHS constant.

STEP 5: Read the coefficient row wise.

STEP 6: Calculate the value of unknown variable using

$$X[i]=c[i]/a[i][i];$$

STEP 7: Obtain the result.

STEP 8: Stop the process.

```
/* Gauss elimination Method*/
```

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

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

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

```
void main()
```

```
{
```

```
int i,j,k,n;
```

```
float a[10][10],x[10],c[10];
```

```
clrscr();
```

```
printf("\n\t\t GAUSS ELIMINATION METHOD\n");
```

```
printf("Enter the value of n:\n");
```

```
scanf("%d", &n);
```

```
printf("Enter the right hand side constant:\n");
```

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

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

```
printf("Enter the coefficient of row wise:\n");
```

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

```
{
```

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

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

```
}
```

```
for(k=0;k<n-1;k++)
```

```

{
for(i=k+1;i<n;i++)
{
for(j=k+1;j<n;j++)
a[i][j]=a[i][j]-(a[i][k]/a[k][k]*a[k][j]);
c[i]=c[i]-(a[i][k]/a[k][k]*c[k]);
}
}

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

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

printf("\n x[%d]=%7.3f",n-1,x[n-1]);

for(k=0;k<n-1;k++)
{
i=n-k-2;

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

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

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

printf("\n x[%d]=%7.3f",I,x[i]);

}

getch();

}

```

Output:

GAUSS ELIMINATION METHOD

Enter the value of n:

3

Enter the right hand side constants:

3 10 13

Enter the coefficient of row wise:

1 2 1

2 3 3

3 -1 2

The solution is:

$X[2]=3.000$

$X[1]=-1.000$

$X[0]=2.000$

RESULT:

The above program has been executed successfully and the output is verified.

EX.NO:3	GAUSS JORDAN METHOD
---------	---------------------

AIM:

To solve the system of linear algebraic equation by Gauss Jordan method using C program.

ALGORITHM:

STEP1: Start the process.

STEP2: Declare the variable i,j,k,n as integer data type and x[10],a[10][10] as float data type.

STEP3: Get the 'n' value using scanf statement.

STEP4: Get the right hand side constant in array in for loop using scanf statement.

STEP5: Get the coefficient row wise value using for loop (i=0;i<n;i++)(j=0;j<n;j++) and in array a[i][j].

STEP6: Calculate $a[i][j] = a[i][j] - (a[i][k]/a[k][k]*a[k][j])$;

STEP7: Get the result using for loop and by using

$$x[i] = a[i][n]/a[i][i]$$

and print the result.

STEP8: Stop the process

/* Gauss Jordan Method*/

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

int i,j,k,n;

float x[10],a[10][10];

clrscr();

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

scanf("%d",&n);

printf("\n Enter the right hand side constants:\n");

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

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

printf("\n Enter the coefficient row wise:");

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

{

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

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

printf("\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]);  
}  
}  
}
```

```
printf("\n The solution is : \n");  
for(i=0;i<n;i++)  
{  
x[i]=(a[i][n]/a[i][i]);  
printf("x[%d]=%f\n",i,x[i]);  
}  
getch();  
}
```

OUTPUT:

Enter the values for n:

3

Enter the right hand side constants:

4 9 2

Enter the coefficient row wise:

1 1 2

2 -1 3

3 -1 -1

The solution is:

$X[0] = 1.000000$

$X[1] = -1.000000$

$X[2] = 2.000000$

RESULT:

The above program has been executed successfully and the output is verified.

EX.NO:4	GAUSS JACOBI METHOD
---------	---------------------

AIM:

To solve the system of linear algebraic equation by Gauss Jacobi method using C program.

ALGORITHM:

STEP1: Start the process.

STEP2: Declare the variable i,j,m,n,l as integer data type and

x[10],a[10][10],b[10],c[10] as float data type.

STEP3: Get the 'n' value and 'l' value using scanf statement.

STEP4: Get the right hand side constant in array in the for loop using scanf statement.

STEP5: Get the coefficient row wise value using for loop

(i=0;i<n;i++),(j=0;j<m;j++) and in array a[i][j].

STEP6: Calculate $c[i] = c[i] - a[i][j] * x[j]$;

STEP7: Get the result using for loop and by using

$x[i] = c[i] / a[i][i]$

and print the result.

STEP8: Stop the process.

GAUSS JACOBI'S METHOD\

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int i,j,m,n,l;
float x[10],a[10][10],b[10],c[10];
clrscr();
printf("Enter the value of n:\n");
scanf("%d",&n);
printf("Enter the no. of iterations:\n");
scanf("%d",&l);
printf("Enter the right hand side constants:\n");
for(i=0;i<n;i++)
scanf("%f",&b[i]);
printf("Enter the coefficients row wise:\n");
for(i=0;i<n;i++)
{
x[i]=0;
for(j=0;j<n;j++)
scanf("%f",&a[i][j]);
}
m=1;
line:
for(i=0;i<n;i++)
{
c[i]=b[i];
for(j=0;j<n;j++)
{
if(i!=j)
c[i]=c[i]-a[i][j]*x[j];
}
}
for(i=0;i<n;i++)
x[i]=c[i]/a[i][i];
m=m+1;
```

```
if(m<=l)
goto line;
else
{
printf("The solution is:\n");
for(i=0;i<n;i++)
printf("x[%d]=%f\n",i,x[i]);
}
getch();
}
```

OUTPUT:

Enter the values of n:

3

Enter the number of iteration:

10

Enter the right hand side constants:

9 -22 22

Enter the coefficient row wise:

10 2 1

1 10 -1

-2 3 10

The solution is:

$X[0] = 1.000000$

$X[1] = -2.000000$

$X[2] = 3.000000$

RESULT:

The above program has been executed successfully and the output is verified.

EX.NO:5	GAUSS SEIDAL METHOD
---------	---------------------

AIM:

To solve the system of linear algebraic equation by Gauss Jacobi method using C program.

ALGORITHM:

STEP1: Start the process.

STEP2: Declare the variable i,j,m,n,l as integer data type and

x[10],a[10][10],b[10],c as float data type.

STEP3: Get the 'n' value and 'l' value using scanf statement.

STEP4: Get the right hand side constant in array in the for loop using scanf statement.

STEP5: Get the coefficient row wise value using for loop

(i=0;i<n;i++),(j=0;j<n;j++) and in array a[i][j].

STEP6: Calculate $c = c - a[i][j] * x[j]$;

STEP7: Get the result using for loop and by using

$x[i] = c / a[i][i]$

and print the result.

STEP8: Stop the process.

GAUSS SEIDAL METHOD\

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int i,j,m,n,l;
float x[10],a[10][10],b[10],c;
clrscr();
printf("\n Enter the value of n:\n");
scanf("%d",&n);
printf("\n Enter the no.of of iteration:\n");
scanf("%d",&l);
printf("Enter the right side constants:\n");
for(i=0;i<n;i++)
scanf("%f",&b[i]);
printf("Enter the coefficient row wise:\n");
for(i=0;i<n;i++)
{
x[i]=0;
for(j=0;j<n;j++)
scanf("%f",&a[i][j]);
}
m=1;
line:
for(i=0;i<n;i++)
{
c=b[i];
for(j=0;j<n;j++)
{
if(i!=j)
c=c-(a[i][j]*x[j]);
}
x[i]=c/a[i][i];
}
```

```
m=m+1;
if(m<=l)
goto line;
else
{
printf("The solution is:\n");
for(i=0;i<n;i++)
printf("x[%d]=%f\n",i,x[i]);
}
getch();
}
```

OUTPUT:

Enter the values of n:

3

Enter the no. of iteration:

10

Enter the right hand side constants:

-6 -7 -8

Enter the coefficient row wise:

10 -2 -2

-1 10 -1

-1 -1 10

The solution is:

$X[0] = -0.976744$

$X[1] = -0.896406$

$X[2] = -0.987315$

RESULT:

The above program has been executed successfully and the output is verified.

EX.NO:6	TRIANGULARIZATION METHOD
---------	--------------------------

AIM:

To solve a set of simultaneous linear equation by Triangularization method using C-program.

ALGORITHM:

STEP 1: Start the process.

STEP 2: Declare the variables l,I,m,n,k as int data type and

x[10],a[10][10],b[10][10],c[10],u[10][10] as float data type.

STEP 3: Get the 'n' values using scanf statement.

STEP 4: Get the right hand side constant in array in for loop using scanf statement.

STEP 5: Get the coefficient row wise values using for loop

(i=0;i<n;i++),(j=0;j<n;j++) and in array a[i][j].

TRIANGULARIZATION METHOD\

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
    Int i,j,k,m,n;
    float x[10],a[10][10],b[10][10],u[10][10],c[10];
    clrscr();
    printf("Enter the value of n:\n");
    scanf("%d",&n);
    printf("Enter the right hand side constants:\n");
    for(i=0;i<n;i++)
        scanf("%f",a[i][n]);
    printf("Enter the coefficents row wise:\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            scanf("%f",&a[i][j]);
    }
    for(i=0;i<n;i++)
        b[i][0]=a[i][0];
    for(j=1;j<n+1;j++)
        u[0][j]=(a[0][j]/b[0][0]);
    for(m=1;m<n;m++)
    {
        for(i=m;i<n;i++)
        {
            for(k=0;k<m;k++)
                a[i][m]=a[i][m]-b[i][k]*u[k][m];
            b[i][m]=a[i][m];
        }
        for(j=m+1;j<n+1;j++)
        {
            for(k=0;k<m;k++)
                a[m][j]=a[m][j]-b[m][k]*u[k][j];
            u[m][j]=a[m][j]/b[m][m];
        }
    }
}
```

```
}  
}  
for(k=0;k<n;k++)  
{  
    i=(n-k-1);  
    x[i]=u[i][n];  
    for(j=i+1;j<n;j++)  
        x[i]=x[i]-u[i][j]*x[j];  
}  
printf("The solution is:\n");  
for(i=0;i<n;i++)  
    printf("x[%d]=%f\n",i,x[i]);  
getch();  
}
```

OUTPUT:

Enter the values of n:

3

Enter the right hand side constants:

12 20 33

Enter the coefficient row wise:

2 1 4

8 -3 2

4 11 -1

The solution is:

$X[0] = 3.000000$

$X[1] = 2.000000$

$X[2] = 1.000000$

RESULT:

The above program has been executed successfully and the output is verified.

EX.NO:7	TRAPEZOIDAL RULE
---------	------------------

AIM:

To evaluate the integral by trapezoidal rule using c program.

ALGORITHM:

STEP1: Start the process.

STEP2: Declare the variables in as integer data type .

STEP3: Declare the variables a,b,x [30],y[30],n in float data type and double as h,
i1.

STEP4: Get the upper and lower limits using scanf statement .

STEP5: Make sure that the sub intervals should not exceed 30.

STEP6: Calculate h value

$$h=(b-a)/n;$$

STEP7: Calculate i value by using for loop and initialize i1=0.

STEP8: Get the result by using the formula

$$i1=i1+(h/2)*(x[i]+y[i+1]);$$

STEP9: Stop the process.

TRAPIZOIDAL RULE\

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x)(log(x))
void main()
{
int i,n;
float a,b,x[30],y[30];
double h,i1;
clrscr();
printf("Enter the lower and upper limits:\n");
scanf("%f%f",&a,&b);
printf("Enter the no. of subintervals (multiple of 6 not exceeding 30):\n");
scanf("%d",&n);
h=(b-a)/n;
x[0]=a;
for(i=0;i<n+1;i++)
{
y[i]=f(x[i]);
if(i==n)
goto line;
x[i+1]=x[i]+h;
}
line:
i1=0;
for(i=0;i<n;i++)
i1=i1+(h/2)*(y[i]+y[i+1]);
printf("By trapezoidal rule the solution is:%f",i1);
getch();
}
```

OUTPUT:

Enter the lower and upper limits:

4 5.2

Enter the no.of sub intervals(multiple of 6 notexceeding 30):

6

By Trapezoidal Rule the solution is: 1.827655

RESULT:

The above program has been executed successfully and the output is verified.

EX.NO:8	SIMPSON'S RULE
---------	----------------

AIM:

To using evaluate the integral by simpson's 1/3 rule using c program.

ALGORITHM:

STEP1: Start the process.

STEP2: Declare the variables i,n as integer data type.

STEP3: Declare the variables a,b,x[30],y[30] in float data type and double as h,i1.

STEP4: Get the upper and lower limits using scanf statement.

STEP5: Make sure that the sub intervals should not exceed 30.

STEP6: Calculate h value

$$h=(b-a)/n;$$

STEP7: Calculate i value by using for loop and initialize i1=0.

STEP8: Get the result by using the formula.

$$i1=i1+(h/3)*(y[i]+4*y[i+1]+y[i+2]);$$

STEP9: Stop the process.

SIMPSON'S RULE\

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x)(log(x))
void main()
{
    int i,n;
    float a,b,x[30],y[30];
    double h,i1;
    clrscr();
    printf("Enter the lower and upper limits:\n");
    scanf("%f%f",&a,&b);
    printf("Enter the no. of subintervals (multiple of 6 not exceeding 30):\n");
    scanf("%d",&n);
    h=(b-a)/n;
    x[0]=a;
    for(i=0;i<n+1;i++)
    {
        y[i]=f(x[i]);
        if(i==n)
            goto line;
        x[i+1]=x[i]+h;
    }
    line:
    i1=0;
    for(i=0;i<n-1;i+=2)
        i1=i1+(h/3)*(y[i]+4*y[i+1]+y[i+2]);
    printf("By simpson's 1/3 rule the solution is:\t%f",i1);
    getch();
}
```

OUTPUT:

Enter the lower and upper limits:

4 5.2

Enter the no.of sub intervals(multiple of 6 not exceeding 30):

6

By Simpsons1/3 Rule the solution is: 1.827647

RESULT:

The above program has been executed successfully and the output is verified.

EX.NO:9	EULER METHOD
---------	--------------

AIM:

To solve an ordinary differential equation by Euler Method using c program.

ALGORITHM:

STEP1: Start the process.

STEP2: Define a function as $f(x,y)$.

STEP3: Declare the variables h, x_n, x, y_s are float data type i as int.

STEP4: Get the value for x and y_s and h using scanf statement.

STEP5: Calculate the values for y_s using the formulas

$y_s = y_s + h * f(x, y_s);$

$x_n = x + h;$

STEP6: Printf y_s values using for loop and print the result.

STEP7: Stop the process.

EULER METHOD\

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x,y)(x+y)
void main()
{
    int i;
    float h,xn,x,ys;
    clrscr();
    printf("Enter the values of x and y:\n");
    scanf("%f%f",&x,&ys);
    printf("\nEnter the step size:\n");
    scanf("%f",&h);
    for(i=1;i<11;i++)
    {
        ys=ys+h*f(x,ys);
        xn=x+h;
        x=xn;
        printf("\n%f",ys);
    }
    getch();
}
```

OUTPUT:

Enter the values of x and y :

0.0

1.0

Enter the step size:

0.2

1.200000

1.480000

1.856000

2.347200

2.976640

3.771968

4.766362

5.999635

7.519562

9.383474

RESULT:

The above program has been executed successfully and the output is verified

EX.NO:10	RUNGE-KUTTA SECOND ORDER METHOD
----------	---------------------------------

AIM:

To solve a set of ordinary differential equation by Runge-Kutta Second Order Method by using c program.

ALGORITHM:

STEP1: Start the process.

STEP2: Declare the variables h,k1,k2,x,y as float data type.

STEP3: Define a function as f(x,y)(-y).

STEP4: Get the values for x and y and h using scanf statement.

STEP5: Calculates the values x, y,k1,k2 by using for loop and using formula.

$k1 = h * f(x, y);$

$k2 = h * f(x + h/2, (y + k1/2));$

$y = y + k2;$

STEP6: Print the y values for using for loop.

STEP7: Stop the process.

RUNGE-KUTTA SECOND ORDER METHOD\

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x,y)(-y)
void main()
{
int i;
float h,k1,k2,x,y;
clrscr();
printf("Enter the initial values of x & y:");
scanf("%f%f",&x,&y);
printf("Enter the step size h:");
scanf("%f",&h);
printf("\n The values are:");
for(i=1;i<11;i++)
{
k1=h*f(x,y);
k2=h*f((x+h/2),(y+k1/2));
y=y+k2;
x=x+h;
printf("\n%d\t%f\t%f",i,x,y);
}
getch();
}
```

OUTPUT:

Enter the initial values of x and y:

0.0

1.0

Enter the step size h:

0.1

The values are:

1	0.100000	0.905000
2	0.200000	0.819025
3	0.300000	0.741218
4	0.400000	0.670802
5	0.500000	0.607076
6	0.600000	0.549404
7	0.700000	0.497210
8	0.800000	0.449975
9	0.900000	0.407228
10	1.000000	0.368541

RESULT:

The above program has been executed successfully and the output is verified.

EX.NO:11	EXPLICIT METHOD
----------	-----------------

AIM:

To solve parabolic equation by explicit method using C program.

ALGORITHM:

STEP 1: Start the process.

STEP 2: Define the function $f(x)(x*(10-x)/2)$.

STEP 3: Declare the variables i,m,j,n as int data type and h,k,a,u,c as float data type.

STEP 4: Using scanf statement get the values.

STEP 5: Using for loop $j=0$ to $m+1$ assign the values $u[0][j]=0; u[n][j]=0$

STEP 6: Using for loop $i=0$ to $n-1$ calculate the values for u and c

$u[i][0]=f(i*h);$

$c=((a*k)/(h*h));$

STEP 7: Using the for loop $j=0$ to $m-1$ and $i=0$ to $n-1$ calculate the values for u

$u[i][j+1]=c*u[i-1][i]+(1.0-2.0*c)*u[i][j];$

STEP 8: Using for loop $i=0$ to $n-1$ and print the value of $u(i,j)$.

STEP 9: Using for loop $j=0$ to m and $p=1$ to $m-1$ and print the values of $u[i][j]$.

STEP10: Stop the process

EXPLICIT METHOD\

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x)(x*(10-x)/2)
void main()
{
int i,n,j,m;
float h,k,a,u[20][20],c;
clrscr();
printf("Enter the values of h,k,a,n,m:\n");
scanf("%f%f%f%d%d",&h,&k,&a,&n,&m);
for(j=0;j<m+1;j++)
{
u[0][j]=0;
u[n][j]=0;
}
for(i=1;i<n;i++)
u[i][0]=f(i*h);
c=((a*k)/(h*h));
for(j=0;j<m;j++)
for(i=0;i<n;i++)
u[i][j+1]=c*(u[i-1][j]+u[i+1][j])+(1.0-2.0*c)*u[i][j];
printf("\n");
printf("Solution of parabolic equation with u(x,0)=0.5x(10-x)\n");
printf("\n");
printf("(i,j)\t");
for(i=1;i<n;i++)
printf("u(%d,j)\t",i);
```

```
printf("\n");
for(j=0;j<m+1;j++)
{
printf("(i,%d)\t",j);
{
for(i=1;i<n;i++)
printf("%6.2f\t",u[i][j]);
}
printf("\n");
}
getch();
}
```

OUTPUT:

Enter the values of h ,k , a ,n ,m :

1 0.1 5 10 5

Solution of parabolic equation with $u(x,0)=0.5x(10-x)$:

(i,j)	u(1,j)	u(2,j)	u(3,j)	u(4,j)	u(5,j)	u(6,j)	u(7,j)	u(8,j)	u(9,j)
(i,0)	4.50	8.00	10.50	12.00	12.50	12.00	10.50	8.00	4.50
(i,1)	4.00	7.50	10.00	11.50	12.00	11.50	10.00	7.50	4.00
(i,2)	3.75	7.00	9.50	11.00	11.50	11.00	9.50	7.00	3.75
(i,3)	3.50	6.62	9.00	10.50	11.00	10.50	9.00	6.62	3.50
(i,4)	3.31	6.25	8.56	10.00	10.50	10.00	8.56	6.25	3.31
(i,5)	3.12	5.94	8.12	9.53	10.00	9.53	8.12	5.94	3.12

RESULT:

The above program has been executed successfully and the output is verified.

EX.NO:12	CRANK NICOLSON METHOD
----------	-----------------------

AIM:

To solve the parabolic equation by using the Crank Nicolson Method using c program.

ALGORITHM:

STEP1: Start the process.

STEP2: Define the function $f(x)(\sin(3.14159*x))$.

STEP3: Declare the variables i,j,k,m,n as int data type and

h,a[20][20],u[50],b[20][20] as float data type.

STEP4: Get the input values n,h,m using scanf statement.

STEP5: Using the loop i=1 to n=1 and the print the value u(i,j) using for loop j=0

to m and assign the value

U[0][j]=0;

U[n][j]=0;

STEP6: Using for loop i-1 to n-1 and the calculate

u[i][0]=f(i*h);

STEP7: Using for loop i=1 to n=1 and j=1 to n-1 and using the conditions.

i) (i==j-1||i==j+1) assign a[i][j]=-1.

ii) (i==j), assign a[i][j]=4 else a[i][j]=0.

iii) (i==j),assign a[i][n+j-1]=1 else a[i][n+j-1]=0.

STEP8: Using for loop k=1 to n-1 and i=1 to n-1 and using if condition (i!=k)

again using for loop $j=k+1$ to $(2*n-1)$ calculate

$$A[i][j]=a[i][j]-(a[i][k]/a[k][k])*a[k][j];$$

STEP9: Using for loop $i=1$ to $n-1$ and $j=n$ to $(2*n-1)$ calculate

$$B[i][j-n+1]=a[i][j]/a[i][i];$$

STEP10: Using for loop $j=1$ to m print j , using for loop $i=1$ to $n-1$ assign $u[i][j]=0$

and for loop $k=1$ to $n-1$ calculate

$$U[i][j]=u[i][j]+b[i][k]*(u[k-1][j-1]+u[k+1][j-1]);$$

STEP11: Get the value $u[i][j]$ and print the result.

STEP12: Stop the process.

CRANK NICOLSON'S METHOD\

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x)(sin(3.14159*x))
void main()
{
int i,j,k,m,n;
float h,a[20][20],u[50][50],b[20][20];
clrscr();
printf("Enter the no. of sub intervals on the x axis:\n");
scanf("%d",&n);
printf("Enter the no. of time for which solution is required:\n");
scanf("%d",&m);
printf("Enter the step size h for x:\n");
scanf("%f",&h);
printf("PARABOLIC EQUATION-SIMPLE CRANK-NICOLSON
METHOD:\n");
printf("solution for f(x)=sin((22/7)*x)and zero boundary values\n\n");
printf("j\t");
for(i=1;i<n;i++)
printf("\tu(%d,j)\t",i);
printf("\n");
for(j=0;j<m+1;j++)
{
u[0][j]=0;
u[n][j]=0;
}
for(i=1;i<n;i++)
u[i][0]=f(i*h);
for(i=1;i<n;i++)
{
for(j=1;j<n;j++)
if((i==j-1)||(i==j+1))
a[i][j]=-1;
else if(i==j)
```

```

a[i][j]=4;
else
a[i][j]=0;
}
for(i=1;i<n;i++)
{
for(j=1;j<n;j++)
if(i==j)
a[i][n+j-1]=1;
else
a[i][n+j-1]=0;
}
for(k=1;k<n;k++)
{
for(i=1;i<n;i++)
if(i!=k)
for(j=k+1;j<(2*n-1);j++)
a[i][j]=a[i][j]-(a[i][k]/a[k][k])*a[k][j];
}
for(i=1;i<n;i++)
for(j=n;j<(2*n-1);j++)
b[i][j-n+1]=a[i][j]/a[i][i];
for(j=1;j<m+1;j++)
{
printf("%d\t",j);
for(i=1;i<n;i++)
{
u[i][j]=0;
for(k=1;k<n;k++)
u[i][j]=u[i][j]+b[i][k]*(u[k-1][j-1]+u[k+1][j-1]);
printf("\t%7.4f\t",u[i][j]);
}
printf("\n");
}
getch();
}

```

OUTPUT:

Enter the no. of sub intervals on the X axis :

5

Enter the no. of time for which solution is required:

5

Enter the step size h for x:

0.2

PARABOLIC EQUATION-SIMPLE CRANK-NICOLSON'S METHOD solution

For $f(x) = \sin((22/7)x)$ and zero boundary values:

j	u(1,j)	u(2,j)	u(3,j)	u(4,j)
1	0.3993	0.6460	0.6460	0.3993
2	0.2712	0.4388	0.4388	0.2712
3	0.1842	0.2981	0.2981	0.1842
4	0.1251	0.2025	0.2025	0.1251
5	0.0850	0.1371	0.1376	0.0850

RESULT:

The above program has been executed successfully and the output is verified.