Finite Difference Method in MATLAB

For PDEs: Heat Equation, Advection-Diffiusion Equation,...

Tutor: Dr. Jeevan Kafle

Compiled by: Mr. Sandesh Thakuri

Centeral Department of Mathematics

TU, Nepal

Jan, 2024

Acknowledgement

First, I would like to thank Mr. **Hari Babu Khatri** for helping me with the figures and Ms. **Roshina Shrestha** for providing me the old codes.

Then I would like to thank Mr. **Bikram Bhandari** and Mr. **Nabin Niraula** for helping me out with the errors. Finally, I would like to thank my whole class for the positive feedback. Also, I am thankful for our teacher Dr. Jeevan Kafle.

Thank You.

Sandesh Thakuri
Jan, 2024

Contents

Part I

Finite Difference

One

Finite Difference

In this method we discretize a space into a finite number of points and use these points to find an approximate solution of a PDE.

1.1 Schemes for Discretization

Let $u = u(x,t)$ be a function in two variables.

Let the interval of the variable x is discretized into m number of points; $x_1, x_2, ..., x_m$ and of the variable t into n number of points; $t_1, t_2, ..., t_n$.

Forward Difference of 1^{st} order

$$\frac{\partial u(x_i,t_j)}{\partial x} = \frac{u(x_i+h,t_j) - u(x_i,t_j)}{h} = \frac{V_{i+1}^j - V_i^j}{h}$$

Likewise,

$$\frac{\partial u}{\partial t} = \frac{V_i^{j+1} - V_i^j}{k}$$

Backward Difference of 1^{st} order

$$\frac{\partial u(x_i,t_j)}{\partial x} = \frac{u(x_i,t_j) - u(x_i - h, t_j)}{h} = \frac{V_i^j - V_{i-1}^j}{h}$$

Likewise,

$$\frac{\partial u}{\partial t} = \frac{V_i^j - V_i^{j-1}}{k}$$

Central Difference of 1^{st} order

It is the average of forward difference and backward difference.

$$\frac{\partial u(x_i,t_j)}{\partial x} = \frac{V_{i+1}^j - V_i^j + V_i^j - V_{i-1}^j}{2h} = \frac{V_{i+1}^j - V_{i-1}^j}{2h}$$

Central Difference of 2^{nd} order

$$\frac{\partial^2 u}{\partial x^2} = \frac{V_{i+1}^j - 2V_i^j + V_{i-1}^j}{h^2}$$

$$\frac{\partial^2 u}{\partial t^2} = \frac{V_i^{j+1} - 2V_i^j + V_i^{j-1}}{k^2}$$

Code Conventions:

Variables	Axis	Loop var.	difference	Grid section
x	X	i	h	m
t/y	Y	j	k	n

Two

Heat Equation

Forward time Central space Scheme **(FTCS)** for the heat equation:
$u_t = \alpha u_{xx}$.
We have,

$$\begin{aligned}
\frac{\partial u}{\partial t} &= \alpha \frac{\partial^2 u}{\partial x^2} \\
\frac{V_i^{j+1} - V_i^j}{k} &= \frac{V_{i+1}^j - 2V_i^j + V_{i-1}^j}{h^2} \\
V_i^{j+1} - V_i^j &= \frac{\alpha k}{h^2}(V_{i+1}^j - 2V_i^j + V_{i-1}^j) \\
V_i^{j+1} &= V_i^j + c(V_{i+1}^j - 2V_i^j + V_{i-1}^j)
\end{aligned}$$

where $c = \frac{\alpha k}{h^2}$. This the required scheme.

Question

$$\begin{aligned}
u_t &= \alpha u_{xx} & 0 < x < 1, \quad t > 0, \quad \alpha = 0.05 \\
BCS: \quad u(0,t) &= u(1,t) = 0 & t > 0 \\
IC: \quad u(x,0) &= sin\pi x & 0 < x < 1
\end{aligned}$$

2.1 Matlab Code

```
%initialization
L=1;        m=10;    h=L/m;     x=0:h:L;
T=1;        n=10;    k=T/n;     t=0:k:T;

a=0.05;          c=a*k/(h*h);
v=zeros(m+1, n+1);

%checking
if (c<0|c>0.5)
  disp('The FTCS is unstable.');
else

%boundary equations
  v(1,:)=0;      v(m+1,:)=0;
  v(:,1)=sin(pi*x);

%scheme
  for j=1:n
    for i=2:m
      v(i,j+1)=v(i,j)+c*(v(i+1,j)-2*v(i,j)+v(i-1,j));
    end
  end
end

%graph
[p,q]=meshgrid(x,t);

surf(p,q,v);
xlabel('x');  ylabel('y');  zlabel('z');
title('Numerical solution of heat equation.');
```

Three

Advection Diffusion Equation

Forward Time Backward Space Central Space Scheme (FTBSCS) for the following advection diffiusion equation:

$$\frac{\partial C}{\partial t} + v\frac{\partial C}{\partial x} = D\frac{\partial^2 C}{\partial x^2}$$

where,
C is concentration of disolved sustances
v is velocity of the fluid
D is diffussion coefficient.

We have,

$$\begin{aligned}\frac{V_i^{j+1} - V_1^J}{k} + v\frac{V_i^j - V_{i-1}^j}{h} &= D\frac{V_{i+1}^j - 2V_i^j + V_{i-1}^j}{h^2}\\ V_i^{j+1} - V_1^j &= \frac{Dk}{h^2}(V_{i+1}^j - 2V_i^j + V_{i-1}^j) - \frac{vk}{h}(V_i^j - V_{i-1}^j)\\ V_i^{j+1} &= F(V_{i+1}^j - 2V_i^j + V_{i-1}^j) - G(V_i^j - V_{i-1}^j)\end{aligned}$$

where,
$F = \frac{Dk}{h^2} \qquad G = \frac{vk}{h}$

Question

$$C_t + vC_x = DC_{xx}$$

$$0 \le t \le 4000 * 24, \qquad D = 10^{-6} * 3600$$

$$0 \le x \le 100, \qquad v = 10^{-7} * 3600$$

$$IC: \quad C(x,0) = 100$$

3.1 Matlab Code

```
%Initialization
L=100;     m=20;    h=L/m;     x=0:h:L;
T=4000*24;  n=20;    k=T/n;   t=0:k:T;

D=(1E-6)*3600;       F=(D*k)/(h*h);
V=(1E-7)*3600;       G=(V*k)/h;

v=zeros(m+1,n+1);

%Initial conditon
v(:,1)=100;

%Scheme
for j=1:n
  for i=2:m
    v(i,j+1)=v(i,j)+F*(v(i+1,j)-2*v(i,j)+v(i-1,j))-G*(v(i,j)-v(i-1,j));
  end
end

%Graph
[p,q]=meshgrid(x,t);
surf(p,q,v);
xlabel('x';    ylabel('y');    zlabel('z');
title('Numerical solution of Advection-Diffusion equation.');
```

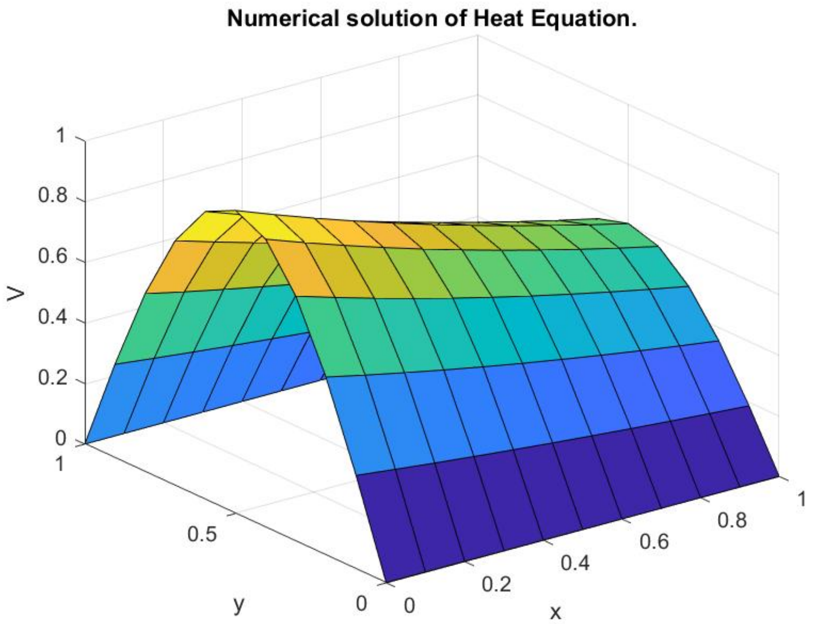


Figure 3.1: Heat Equation

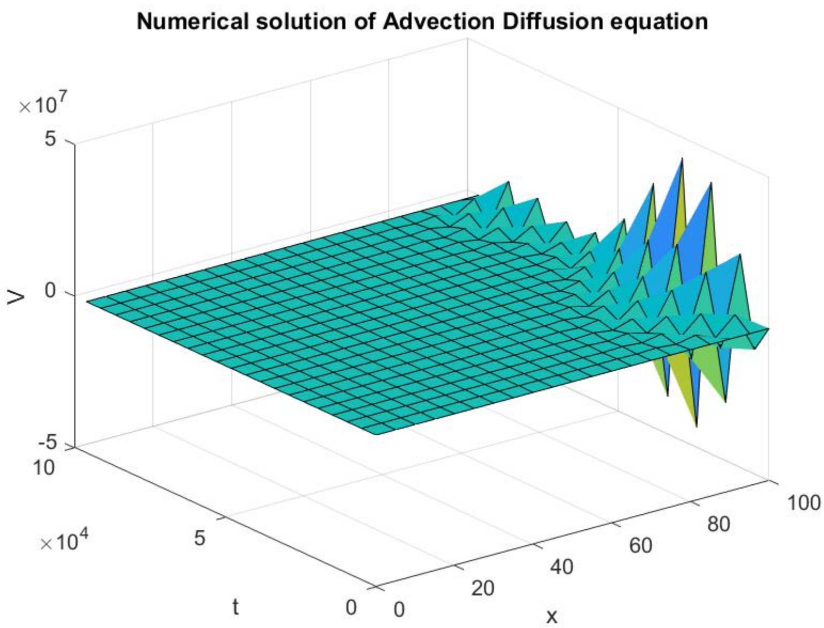


Figure 3.2: Advection Diffiusion equation

Four

Laplace Equation

The Laplace equation is

$$u_{xx} + u_{yy} = 0$$

The Central Space Central Space Scheme (**CSCSS**) for the above equation:

$$\begin{aligned}
\frac{V_{i+1}^{j} - 2V_i^j + V_{i-1}^{j}}{h^2} + \frac{V_i^{j+1} - 2V_i^j + V_i^{j-1}}{k^2} &= 0 \\
k^2(V_{i+1}^{j} - 2V_i^j + V_{i-1}^{j}) + h^2(V_i^{j+1} - 2V_i^j + V_i^{j-1}) &= 0 \\
k^2(V_{i+1}^{j} + V_{i-1}^{j}) + h^2(V_i^{j+1} + V_i^{j-1}) &= 2(h^2 + k^2)V_i^j
\end{aligned}$$

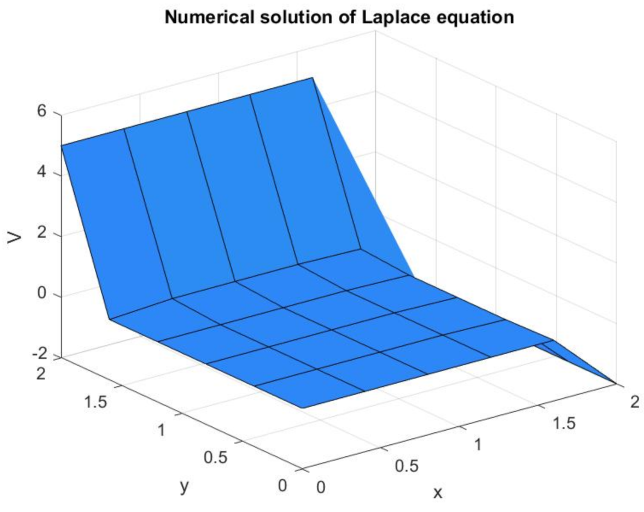
$$V_i^j = \frac{k^2(V_{i+1}^{j} + V_{i-1}^{j}) + h^2({}_i^{j+1} + V_i^{j-1})}{2(h^2 + k^2)} \tag{4.1}$$

Question

$$\begin{aligned}
& u_{xx} = u_{yy} && 0 \le x \le 2, \;\; 0 \le y \le 2 \\
BCS: \;\; & u(0, y) = 0, \;\; u(2, y) = 5 && 0 \le y \le 2 \\
& u(x, 0) = 0, \;\; u(x, 2) = -2 && 0 \le x \le 2
\end{aligned}$$

4.1 Matlab Code

```
%Initialization
L=2;        m=5;    h=L/m;     x=0:h:L;
B=2;        n=5;    k=B/n;     t=0:k:B;

c=1/(2*(h^2+k^2));
v=zeros(m+1, n+1);

%Boundary Conditions
v(1,:)=0;      v(m+1,:)=5;
v(:,1)=0;      v(:,n+1)=-2;

%Scheme
for j=2:n
  for i=2:m
    v(i,j)=c*((h^2*(v(i,j+1)+v(i,j-1)))+(k^2*(v(i+1,j)+v(i-1,j))));
  end
end

%Graph
[p,q]=meshgrid(x,t);

surf(p,q,v);
xlabel('x');  ylabel('y');  zlabel('z');
title('Numerical solution of Laplace equation.');
```



Five

Wave Equation

The wave equation is $\quad u_{tt} = au_{xx}$

The Central Space Central Space Scheme (**CTCSS**) for the above equation:

$$\frac{V_i^{j+1} - 2V_i^j + V_i^{j-1}}{k^2} = a\frac{V_{i+1}^j - 2V_i^j + V_{i-1}^j}{h^2}$$

$$V_i^{j+1} - 2V_i^j + V_i^{j-1} = \frac{ak^2}{h^2}(V_{i+1}^j - 2V_i^j + V_{i-1}^j)$$

$$V_i^{j+1} = 2(1-c)V_i^j + c(V_{i+1}^j + V_{i-1}^j) - V_i^{j-1} \tag{5.1}$$

where, $\quad c = \frac{ak^2}{h^2}$

Question

$$\begin{aligned} & u_{tt} = au_{xx} && 0 \le x \le 1, \ \ t \ge 0 \\ BCs: \ \ & u(0,t) = u(1,t) = 0 && t \ge 0 \\ ICs: \ \ & u(x,0) = 10sin\pi x \ \ u_t(x,0) = 0 && 0 \le x \le 1 \end{aligned}$$

5.1 Scheme for Initial Condition

The initial condition $u_t(x, 0) = 0$ also needs a separate scheme as it is in a derivative form.

The Central Time Scheme for u_t is :

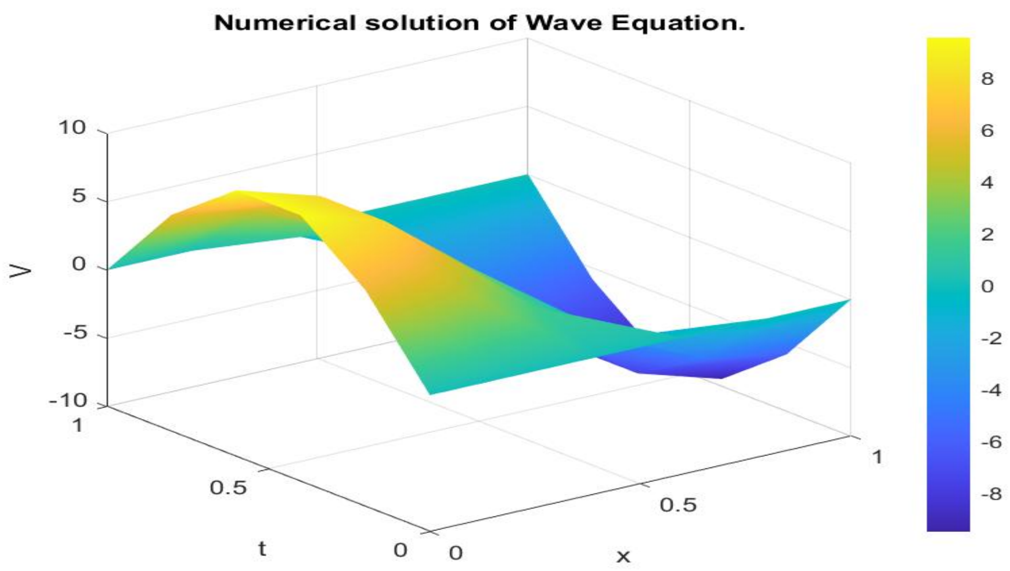
$$u_t = \frac{V_i^{j+1} - V_i^{j-1}}{2k}$$
$$\implies 2ku_t = V_i^{j+1} - V_i^{j-1}$$
$$For j = 0$$
$$2ku_t = V_i^1 - V_i^{-1}$$
$$\implies V_i^{-1} = V_i^1 - 2ku_t \tag{5.2}$$

Also, Putting $j = 0$ in equation (5.1)

$$V_i^1 = 2(1-c)V_i^0 + c(V_{i+1}^0 + V_{i-1}^0) - V_i^{-1} \tag{5.3}$$

From equation (5.2) and (5.3)

$$V_i^1 = (1-c)V_i^0 + 0.5c(V_{i+1}^0 + V_{i-1}^0) + ku_t \tag{5.4}$$



5.2 Matlab Code

```
%Initialization
L=1;        m=5;        h=L/m;        x=0:h:L;
T=1;        n=5;        k=T/n;        t=0:k:T;

a=1;                            c=a*(k^2/h^2);
V=zeros(m+1,n+1);

%Checking
if (c<=0|c>1)
    disp('The CTCSS is unstable')
else

%Boundary Conditions
    V(1,:)=0;        V(m+1,:)=0;

%Initial Conditions
    V(:,1)=10*sin(pi*x);

    for i=2:m
        V(i,2)=(1-c)*V(i,1)+0.5*c*(V(i+1,1)+V(i-1,1));
    end

%Main Scheme
    for j=2:n
        for i=2:m
            V(i,j+1)=2*(1-c)*V(i,j)+c*(V(i+1,j)+V(i-1,j))-V(i,j-1);
        end
    end

%Graph
    [p,q]=meshgrid(x,t);
    surf(p,q,V,'Edgecolor','none');
    xlabel('x');        ylabel('t');        zlabel('V');
    title('Numerical solution of Wave Equation.');
    shading interp;
end
```

Part II

Inital Part

Six

Numerical Integartion

6.1 Code for Trapezoidal Rule

```

% Initialization
f=@(x) x*sin(x);

a=0;      b=(pi/2);
n=5;      h=(b-a)/n;

S=0.5*(f(a)+f(b));
S1=0;

% Scheme
for i=1:n-1
  xi=a+i*h;
  S1=S1+f(xi);
end

I=h*(S+S1);

fprintf('The integral is %f\n',I)

```

$ans = 1.008265$

6.2 Code for Simpson's 1/3 rule

```

%Initialization
f=@(x) x*sin(x);

a=0;     b=(pi/2);
n = 12;  h=(b-a)/n;

S=f(a)+f(b);
S1=0;
S2=0;

% Scheme
for i=1:2:n-1
  xi=a+i*h;
  S1=S1+4*f(xi);
end

for i=2:2:n-2
  xi=a+i*h;
  S2=S2+2*f(xi);
end

% Output
I=(h/3)*(S+S1+S2);
fprintf('The integral value is %f.\n',I)

```

$ans = 0.999995$

6.3 Code for Simpson's 3/8 rule

```

% Initialization
f=@(x) x*sin(x);
a=0;           b=pi/2;     n=12;     h=(b-a)/n;

S1=f(a)+f(b);  S2=0;     S3=0;

% Scheme
for i=1:3:n-2
  x1=a+i*h;
  x2=a+(i+1)*h;
  S2=S2+f(x1)+f(x2);
end

for i=3:3:n-3
  S3=S3+f(a+i*h);
end

% Output
I=(3*h/8)*(S1+3*S2+2*S3);
fprintf('The integral value is %f.\n',I)

```

$$ans = 0.999989$$

Seven

Numerical Solution Polynomial equations

7.1 Bisection Method

```

%Initialization
f=@(x) x^2-26;
a=5;     b=6;     toll=0.001;

% Scheme
while abs(a-b)>=toll
  c=(a+b)/2;

  if f(a)*f(c)<=0
    b=c;
  else
    a=c;
  end
end

fprintf("The solution by the bisection method is %f.\n",c)
```

$$ans = 5.098633$$

7.2 Newton Raphson's Method

```
%Initialization
f=@(x) 3*x*sin(x)-exp(x);
df=@(x) 3*x*cos(x)+3*sin(x)-exp(x);

a0=1;       tollerance=0.0001;   diff=1;

% Scheme
while diff>=tollerance
  a1=a0-f(a0)/df(a0);

  diff=abs(a1-a0);
  a0=a1;
end

fprintf("The solution by Newton Raphson's method is %f.\n",a0)
```

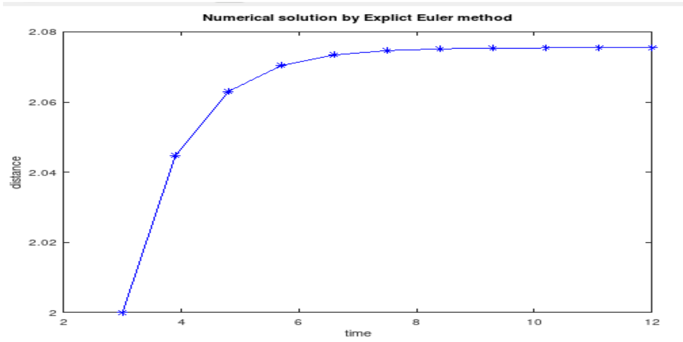
$ans = 1.159431$

Eight

Numerical Soultion of ODEs

8.1 Explicit Euler Method

```

% Initialization
f=@(t,y) exp(-t);

a=3;      b=12;
n=10;     h=(b-a)/n;    t=a:h:b;

y(1)=2;   % Inital guess

% Scheme
for i=1:n
  y(i+1)=y(i)+h*f(t(i),y(i));

  fprintf('Solution at t=%d is %f.\n',t(i+1),y(i+1));
end

% Graph
plot(t,y,'-b*');
xlabel('time');   ylabel('distance');
title('Numerical solution by Explict Euler method');
```



8.2 Implicit Euler Method

```

% Initialization
f=@(t,y) exp(-t);

a=3;       b=12;
n=10;      h=(b-a)/n;     t=a:h:b;

y(1)=2;    % Inital guess

% Scheme
for i=1:n
  k(i)=f(t(i+1),y(i)+h*f(t(i),y(i)));
  y(i+1)=y(i)+h*k(i);

  fprintf('Solution at t=%d is %f.\n',t(i+1),y(i+1));
end

% Graph
plot(t,y,'-b*');
xlabel('time');   ylabel('distance');
title('Numerical solution by Implict Euler method');
```

8.3 Second order Runge-Kutta method

```
% Initialization
f=@(t,y) exp(-t);

a=3;      b=12;
n=10;     h=(b-a)/n;     t=a:h:b;

y(1)=2;    % Inital guess

% Scheme
for i=1:n
  k(i)=f(t(i),y(i));
  k(i+1)=f(t(i+1),y(i)+h*k(i));

  y(i+1)=y(i)+0.5*h*(k(i)+k(i+1));

  fprintf('Solution at t=%d is %f.\n',t(i+1),y(i+1));
end

% Graph
plot(t,y,'-b*');
xlabel('time');   ylabel('distance');
title('Second order Runge-Kutta method');
```

8.4 Fourth order Runge-Kutta Method

```

% Initialization
f=@(t,y) exp(-t);

a=3;      b=12;
n=10;     h=(b-a)/n;    t=a:h:b;

y(1)=2;   % Inital guess

% Scheme
for i=1:n
  k(i)=f(t(i),y(i));
  k(i+1)=f(t(i)+0.5*h,y(i)+0.5*h*k(i));
  k(i+2)=f(t(i)+0.5*h,y(i)+0.5*h*k(i+1));
  k(i+3)=f(t(i)+h,y(i)+h*k(i+2));

  y(i+1)=y(i)+(h/6)*(k(i)+2*k(i+1)+2*k(i+2)+k(i+3));

  fprintf('Solution at t=%d is %f.\n',t(i+1),y(i+1));
end

% Graph
plot(t,y,'-b*');
xlabel('time');   ylabel('distance');
title('Fourth order Runge-Kutta method');
```

Nine

Solution of System of Equations

9.1 Gauss Elimination Method

```
% Initialization
A=[1,3,-2; 3,5,6; 2,4,3];
b=[5;7;8]
[n,n]=size(A);

% Non zero diagonal elements
for j=1:n-1                       %row1
  for i=j+1:n                     %row2
    if A(j,j)==0
      t=A(j,:);
      A(j,:)=A(i,:);
      A(i,:)=t;
    end
  end
end

%Forward elimination
for j=1:n-1                        %column jth
  for i=j+1:n                      %(j+1)th row
    m(i,j)=A(i,j)/A(j,j);
    A(i,:)=A(i,:)-m(i,j)*A(j,:);
    b(i,:)=b(i,:)-m(i,j)*b(j,:);
  end
end

```

```
%Backward substituion
x(n,:)=b(n,:)/A(n,n); %nth cordinate; i.e last row in 'x'

for i=n-1:-1:1
  x(i,:)=(b(i,:)-A(i,i+1:n)*x(i+1:n,:))/A(i,i);
end
x

```

$ans: \quad x = (-15, 8, 2)$

9.2 Gauss Seidel Method

```
% Initilaziion
A=[13,2,3; 2,15,1; 1,-1,10];     b=[46;33;25];
N=length(b);                     toll=0.0001;
x=zeros(N,1);                    y=zeros(N,1);

for j=1:100
  for i=1:N
    num=b(i)-A(i,1:i-1)*x(1:i-1)-A(i,i+1:N)*x(i+1:N);
    x(i)=num/A(i,i);
  end

  if abs(x-y)<toll
    fprintf('Iteration number is %d\n',j);
    break
  end
  y=x;
end
x

```

$ans: \quad x = 2.7279, 1.6766, 2.3949$

(same for Gauss-Jordan method aswell)

9.3 Gauss-Jordan Method

```
% Initilaziion
A=[13,2,3,46; 2,15,1,33; 1,-1,10,25];
[m,n]=size(A);

% Non zero diagonal elements
for j=1:m-1                      %row1
  for i=j+1:m                    %row2
    if A(j,j)==0
      t=A(j,:);
      A(j,:)=A(i,:);
      A(i,:)=t;
    end
  end
end

%Forward elimination
for j=1:n-2      %column jth; n-2 cause last column is b.
  for i=j+1:m                    %(j+1)th row
    m(i,j)=A(i,j)/A(j,j);
    A(i,:)=A(i,:)-m(i,j)*A(j,:);
    b(i,:)=b(i,:)-m(i,j)*b(j,:);
  end
end
% Backward elimination
for j=n-1:-1:2      %n-1 is the last column excluding b.
  for i=j-1:-1:1
    A(i,:)=A(i,:)-A(j,:)*(A(i,j)/A(j,j));
  end
end

% making pivot element 1
for i=1:m
  A(i,:)=A(i,:)/A(i,i);
  x(i)=A(i,n);
end
x
```

Ten

Grading System

```
% Internal Full Marks is 20 and final full marks is 30
I=input('Internal Marks Obtained:');
F=input('Final Marks Obtained:');

R=min(I,F/30*20*1.2);
fprintf('Revised Internal Marks is %2.2f.\n',R);

T=R+F;
fprintf('Total Marks Obtained is %2.2f.\n',T);

if I>20||F>30
  disp('Wrong Information.');
else
  if T>=45
    y='A';
  elseif T>=40
    y='A-';
  elseif T>=35
    y='B';
  elseif T>=30
    y='B-';
  else
    y='Fail';
  end
end
fprintf('The obtained grade is %s.\n',y)
```