RAMAIAH
Institute of Technology

MATLAB

Lab Manual

Numerical Techniques and Differential Equations – (MAC21)

BE II Semester, Computer Science Engineering Stream

Department of Mathematics, Ramaiah Institute of Technology, Bengaluru - 54

Lab 01

Taylor series for Single Variable:

Syntax

```
taylor(f,var)
taylor(f,var,a)
```

approximates `f` with the Taylor series expansion of `f` at the point `var = a`.

Program 01

```
syms x
T1=taylor(exp(x)) %Maculurian Series
T2=taylor(sin(x))
T3=taylor(log(sin(x)),x,2) % Taylor series at x=2
T4=taylor(log(x),x,1)
```

Output:

```
T1 =
 x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1

T2 =
 x^5/120 - x^3/6 + x

T3 =
log(sin(2)) - (x - 2)^4*(cos(2)^2/(12*sin(2)^2) + (cos(2)*(cos(2)/(12*sin(2))
+ (cos(2)*(cos(2)^2/(4*sin(2)^2) + 1/6))/sin(2)))/sin(2) + 1/12) + (x -
2)^3*(cos(2)/(12*sin(2)) + (cos(2)*(cos(2)^2/(3*sin(2)^2) + 1/4))/sin(2)) +
(x - 2)^5*((7*cos(2))/(240*sin(2)) + (cos(2)*((5*cos(2)^2)/(72*sin(2)^2) +
(cos(2)*((5*cos(2))/(72*sin(2)) + (cos(2)*(cos(2)^2/(5*sin(2)^2) +
1/8))/sin(2)))/sin(2) + 1/16))/sin(2) - (cos(2)*(cos(2)^2/(3*sin(2)^2) +
1/4))/(6*sin(2)) + (cos(2)*(cos(2)^2/(4*sin(2)^2) + 1/6))/(2*sin(2))) -
(cos(2)^2/(2*sin(2)^2) + 1/2)*(x - 2)^2 + (cos(2)*(x - 2))/sin(2)

T4 =
x - (x - 1)^2/2 + (x - 1)^3/3 - (x - 1)^4/4 + (x - 1)^5/5 - 1
```

Program 02

```
syms x
T1=taylor(exp(x)) %Maculurian Series
T2=taylor(sin(x))
T3=taylor(log(sin(x)),x,2) % Taylor series at x=2
T4=taylor(log(x),x,1)
```

Output:

```
T1 =
 x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1

T2 =
 x^5/120 - x^3/6 + x

T3 =
log(sin(2)) - (x - 2)^4*(cos(2)^2/(12*sin(2)^2) + (cos(2)*(cos(2)/(12*sin(2))
+ (cos(2)*(cos(2)^2/(4*sin(2)^2) + 1/6))/sin(2)))/sin(2) + 1/12) + (x -
2)^3*(cos(2)/(12*sin(2)) + (cos(2)*(cos(2)^2/(3*sin(2)^2) + 1/4))/sin(2)) +
(x - 2)^5*((7*cos(2))/(240*sin(2)) + (cos(2)*((5*cos(2)^2)/(72*sin(2)^2) +
```

```
(cos(2)*((5*cos(2))/(72*sin(2)) + (cos(2)*(cos(2)^2/(5*sin(2)^2) +
1/8))/sin(2)))/sin(2) + 1/16))/sin(2) - (cos(2)*(cos(2)^2/(3*sin(2)^2) +
1/4))/(6*sin(2)) + (cos(2)*(cos(2)^2/(4*sin(2)^2) + 1/6))/(2*sin(2))) -
(cos(2)^2/(2*sin(2)^2) + 1/2)*(x - 2)^2 + (cos(2)*(x - 2))/sin(2)

T4 =
x - (x - 1)^2/2 + (x - 1)^3/3 - (x - 1)^4/4 + (x - 1)^5/5 - 1
```

To represent term in ascending order:

Syntax

```
sympref('PolynomialDisplayStyle','ascend');
```

Program

```
sympref('PolynomialDisplayStyle','ascend');
T1
```

Output:

```
T1 =
1 + x + x^2/2 + x^3/6 + x^4/24 + x^5/120
```

Taylor series for Two-Variable:

Program

```
syms x y
f = y*exp(x - 1) - x*log(y);
T = taylor (f, [x y], [1 1],'Order',3) % Represent # of term in the series
```

Output

```
T =
x + (- 1 + x)^2/2 + (- 1 + y)^2/2
```

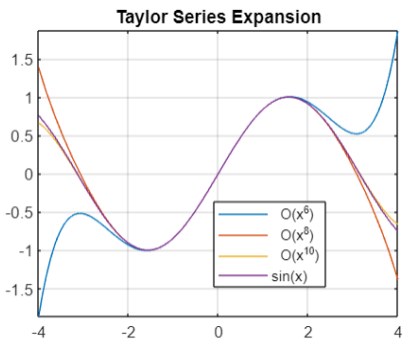
Taylor series for multvariable:

```
syms x y z
f = sin(x) + cos(y) + exp(z);
T = taylor(f)
```

Output

```
T =
x + cos(y) + exp(z) - x^3/6 + x^5/120
```

Plotting Taylor series for different order	Output

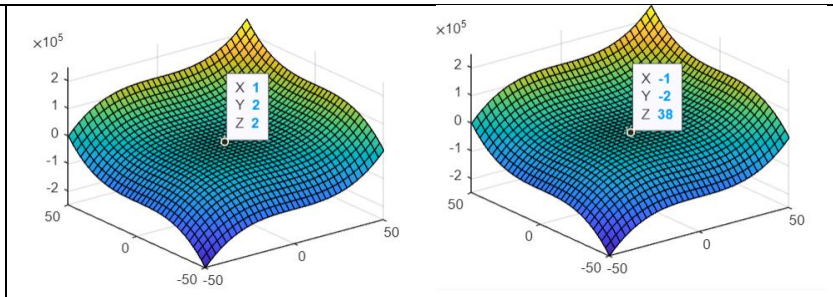
`syms x`
`f = sin(x);`
`T6 = taylor (f, x);`
`T8 = taylor (f, x,'Order',8);`
`T10 = taylor (f, x,'Order',10);`
`fplot ([T6 T8 T10 f])`
`xlim ([-4 4])`
`grid on`
`legend(O(x^6)', O(x^8)', O(x^{10})',`
`'sin(x)')`
`title('Taylor Series Expansion')`
	

Maxima and Minima for two variable functions:

```
syms x y real
f=input('Enter the function f(x,y):')

dfx=diff(f,x); %first derivative
dfy=diff(f,y);

eqns=[dfx==0,dfy==0];
S=solve(eqns,[x y], 'Real', true);% Gives stationary points
G=S.x;
H=S.y;

dfxx=diff(dfx,x); % f_xx %double derivative
dfyy=diff(dfy,y); % f_yy
dfxy=diff(dfy,x); %f_xy

A=subs(dfxx,S); %for AC-B^2
B=subs(dfxy,S);
C=subs(dfyy,S);
fun=subs(f,S); % Subsitutuion of values

for i=1: length(A)

if A(i)*C(i)-B(i)*B(i)>0 & A(i)<0
 fprintf('The Maximum point is %d %d',[G(i), H(i)]);
 fprintf('\n The maximum value is %d',fun(i));
figure;
fsurf(f,[-3 3 -3 3]);
 hold on
 plot3(G(i),H(i),fun,'*w','markersize',10)
hold off

 elseif A(i)*C(i)-B(i)*B(i)>0 & A(i)>0
 fprintf('Minimum point is %d %d',[G(i), H(i)]);
 fprintf('\n The minimum value is %d',fun(i));
figure;
fsurf(f,[-3 3 -3 3]);
 hold on
 plot3(G(i),H(i),fun,'*w','markersize',10)
hold off

elseif A(i)*C(i)-B(i)*B(i)<0
 fprintf('Saddle point is %d %d', [G(i), H(i)]);

else
 fprintf('No conclusion can be drawn for %d %d',[G(i), H(i)]);
end
end
```

Output:

```
Enter the function f(x,y):
x^3+y^3-3*x-12*y+20
The Maximum point is -1 -2
The maximum value is 38
Saddle point is 1 -2
Saddle point is -1 2
Minimum point is 1 2
The minimum value is 2
```



Exercise Problems

1. Using Maclaurin's series expand $\sqrt{1+\sin x}$ up to the term containing x^4.
2. Obtain the first four terms of Taylor's series of $\cos x$ about $x=\frac{\pi}{3}$.
3. Expand $\sin^{-1} x$ in powers of x up to second-degree term.
4. Expand a^x in powers of x up to the first three terms.
5. Expand $\tan^{-1} x$ in powers of $(x-1)$ up to the term containing $(x-1)^4$.
6. Expand $\log(1+\sin 2x)$ in powers of x up to the term containing x^4.
7. Expand the following functions in powers of x and y up to second-degree terms:

   i) $\sin x \sin y$ ii) $e^x \sin y$ iii) $e^{x^2-y^2}$ iv) $e^x \log(1+y)$.
8. Expand the following functions at the given point up to second-degree terms:

   i) $xy^2+\cos(xy)$ about $\left(1,\pi/2\right)$ ii) $x^2y+3y-2$ about (1,-2) iii) x^y about (1,1).
9. Discuss the maxima and minima of

   (i) $x^3y^2(1-x-y)$. (ii) x^3+y^3-3axy (iii) $f(x,y)=xy(1-x-y)$.

Lab 02

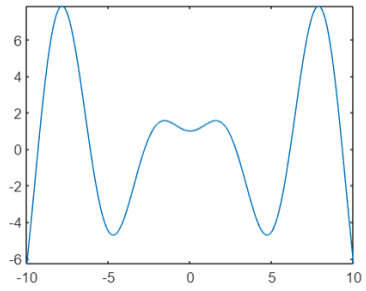
Newton-Raphson method:

Single Variable:

```
clc;
clear;
syms x
f(x)= x*sin(x)+cos(x);
df(x)= diff (f, x);
a=2; b=3;
if f(a)*f(b)<0
x0=(a+b)/2;
else
fprintf ("change interval value")
end
 %x0=input ("enter the initial approximation:");
for n=1: 5000
    xn= x0-(f(x0)/df(x0));
fprintf ('\n The required root at %d is: %f', n, xn);
if abs(f(xn)) <=0.000001
      break;
    else
        x0=xn;
    end
end
fplot(f);
xlim([-10 10])
```

Output:

```
The required root at 1 is: 2.847022
The required root at 2 is: 2.799175
The required root at 3 is: 2.798386
```



Multivariable:

```
clc;
syms x y h k
f (x, y) =x^2-y^2-4;
g (x, y) =y^2+x^2-16;

dfx = diff (f, x);%derivative
dfy = diff(f,y);
dgx = diff(g,x);
dgy = diff(g,y);

 x0=input ("Enter the initial value of x:");
 y0=input ("Enter the initial value of y:");

 for i=1:1000
            EQ1=dfx (x0, y0) *h+dfy (x0, y0) *k+f (x0, y0);
            EQ2=dgx (x0, y0) *h+dgy (x0, y0) *k+g (x0, y0);
```

```
            S=solve ([EQ1==0, EQ2==0],[h,k]);
            xn=x0+S.h;
            yn=y0+S.k;
fprintf ("The required root at %d is %f \n %f\n",i,xn,yn);
    if abs (f(xn, yn))<=0.00001
            break;
            else
                x0=xn;
                y0=yn;
    end
 end
```

Output:

```
Enter the initial value of x:1
Enter the initial value of y:1
The required root at 1 is 5.500000 3.500000
The required root at 2 is 3.659091 2.607143
The required root at 3 is 3.196005 2.454256
The required root at 4 is 3.162456 2.449494
The required root at 5 is 3.162278 2.449490
```

Exercise Problems

1. Using Newton – Raphson iterative method, find a real root of the following equations correct to 4 places of decimals.
   (i) $x^4 - 12x + 7 = 0$ (ii) $x + \log_{10} x = 3.375$ (iii) 2 tan x = 3x (iv) 3x – cosx − 1 = 0
   (v) x sinx + cosx = 0 between 2 and 3 (vi) $cosx = x^2$ *near* 1
   (vii) tanx + tanhx = 0 near x = 2.5 (viii) x^3 – 2x +0.5 = 0 (ix)$x \log_{10} x = 1.2$ near x = 2.5
2. Using Newton – Raphson iterative method, find a real root of the following equations correct to 4 places of decimals.
   i) $x^2 + y = 11; y^2 + x = 7$ $(x_0, y_0) = (3.5, -1.8)$
   ii) $x^3 = y + 100$; $y^3 = x + 150$ $(x_0, y_0) = (4.5,5.3)$
   iii) $x^2 = 3xy - 7;\ y = 2(x + 1)$ $(x_0, y_0) = (0.7,3.4)$
   iv) $x^2 + y^2 = 4;\ 4x^2 - y^2 = 4$ $(x_0, y_0) = (1,1)$

Lab 03

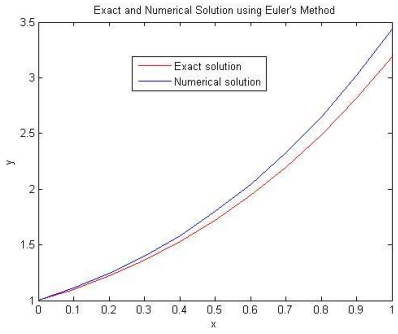
Numerical Solution to ODE

Euler's method

```
%Program to solve ordinary differential equations using Euler's method
clc;
close;
clear;
% Define the function f(x,y) that represents the differential equation
f = @(x,y) x + y;
% Define the initial conditions
x0 = 0; % initial x value
y0 = 1; % initial y value
xn = 1; % final x value
% Define the step size and the range of x values to approximate
h = 0.1; % step size
x = x0:h:xn; % range of x values
% Initialize the y vector with the initial value y0
y = zeros(size(x));
y(1) = y0;
% Use Euler's method to approximate the solution
for i = 2:length(x)
    y(i) = y(i-1) + h * f(x(i-1), y(i-1));
    disp(y(i))
end
%Analytical solution
syms s(t)
ode = diff(s)-t-s;
cond = s(0) == 1;
sSol(t) = dsolve(ode,cond)
t=[0:0.1:1];
%To  plot numerical solution
 plot(x, y, 'r');
 hold on
%To plot the approximate solution
plot(t, sSol(t),'b')
hold off
legend('Exact solution','Numerical solution')
xlabel('x');
ylabel('y');
title('Exact and Numerical Solution using Euler''s Method');
```

Output

```
    1.1000
    1.2200
    1.3620
    1.5282
    1.7210
    1.9431
    2.1974
    2.4872
    2.8159
    3.1875
sSol(t) =
2*exp(t) - t - 1
```



Modified Euler's method

```
%Program to solve ordinary differential equations using modified Euler's
method
clc;
clear;
close;
x0 = 0;
y0 = 1;
h = 0.1; % Step size
xn = 0.1; % Final value
% Define the function y' = f(x,y)
f = @(x,y) -y^2+ x ;
 % Implement Euler's modified method
x = x0:h:xn;
y = zeros(size(x));
y(1) = y0;
 for i = 2:length(x)
    % Predictor step
    y(i) = y(i-1) + h*f(x(i-1), y(i-1));
    fprintf('Required value by Euler''s method (Predictor method) %f \n :' ,
y(i));
    %disp(y(i))
    x(i)=x(i-1)+h;
    y(i,1)=y(i);
    % Corrector step
    for j=2:20
    y(i,j) = y(i-1) + (h/2)*(f(x(i-1), y(i-1)) + f(x(i), y(i,j-1)));
   disp(y(i,j));
    if abs(y(i,j)-y(i,j-1))<0.00001
        y(i)=y(i,j);
        fprintf('Required value by Euler''s modified method (Corrector
method) %f \n :' , y(i));
        break;
    end
    end
end
```

Output:

```
Required value by Euler's method (Predictor method) 0.900000 :
0.9000
0.9145
0.9132
0.9133
0.9133
0.9133
Required value by Euler's modified method (Corrector method) 0.913295
```

Lab 04

Taylor's method

```
%Program to solve ordinary differential equations using modified Taylor's
method
clc;
clear;
syms x y;
%Define function
f(x,y)=1/(x^2+y);
y0=4;
h=0.1;
x0=4;
xn=4.1;
% Define the order of the Taylor series expansion
n=1;
y(1)=y0;
y(2)=f(x,y);
% Compute the values of h^n/n! for n = 0, 1, 2, 3,...,
ht=h.^(0:n)./factorial(0:n);
for i=2:n
    % To compute the values of derivatives
            y(i+1)=diff(y(i),x)
end
for i=1:n
    x=x0+i*h;
    c=eval(y);
    % Compute the values of yb using the Taylor series expansion
    yb=sum(c.*ht);
    fprintf('The value of y(%f) is %f\n',x,yb);
end
```

Output

```
The value of y(4.100000) is 4.004805
```

Runge Kutta 4th order method

```
%Program to solve ordinary differential equations using RK 4th order method
clc;
clear;
f = @(x,y) x+y;
%Define the interval
x0=1;
y0=1;
h=0.2;
xn=2;
n=(xn-x0)/h;
x = x0:h:xn;
y = zeros(size(x));
y(1) = y0;
x(1)=x0;
 for i = 2:length(x)
    k1=h*f(x(i-1),y(i-1));
k2=h*f(x(i-1)+h/2,y(i-1)+k1/2);
k3=h*f(x(i-1)+h/2,y(i-1)+k2/2);
k4=h*f(x(i-1)+h,y(i-1)+k3);
      y(i)=y(i-1)+(k1+2*k2+2*k3+k4)/6;
      x(i)=x(i-1)+h;
    fprintf("The value of y(%f) is %f\n", x(i), y(i));
end
```

Output

```
The value of y(1.200000) is 1.464200
The value of y(1.400000) is 2.075454
The value of y(1.600000) is 2.866319
The value of y(1.800000) is 3.876562
The value of y(2.000000) is 5.154753
```

Exercise Problems

1. Solve the following initial value problems by Euler's method.
   a) $\frac{dy}{dx} = \frac{y-x}{y+x}$; $y(0) = 1$ at $x = 0.4$ by taking $h = 0.1$
   b) $\frac{dy}{dx} = x \log y - y \log x$; $y(1) = 1$ at $x = 1.4$ by taking $h = 0.1$
2. Solve the following initial value problems using Modified Euler's method
   (Carry out three iterations at each stage)
   a) $\frac{dy}{dx} = \frac{2y}{x} + x^3$; $y(1) = 0.5$ at $x = 1.4$ by taking $h = 0.2$
   b) $\frac{dy}{dx} = x + y^2$; $y(0) = 1$ at $x = 0.2$ by taking $h = 0.1$
3. Solve the following initial value problems using Runge - Kutta method of fourth order
   a) $\frac{dy}{dx} = y - x^2$; $y(0.6) = 1.7379$ at $x = 0.8$ by taking $h = 0.1$
   b) $\frac{dy}{dx} = xy + y^2$; $y(0) = 1$ at $x = 0.2$ by taking $h = 0.2$
4. Solve the following initial value problems by Taylor's series method by considering terms up to fourth degree
   a) $\frac{dy}{dx} = 1 - 2xy$; $y(0) = 0$ at $x = 0.2$
   b) $\frac{dy}{dx} = xy^2 - 1$; $y(0) = 1$ at $x = 0.1$

Lab 5

Solving Higher Order Ordinary Differential Equation

`dsolve(eqn)` solves the differential equation eqn, where eqn is a symbolic equation.
Use `diff` and == to represent differential equations. For example, diff(y,x) == y represents the equation $dy/dx = y$. Solve a system of differential equations by specifying eqn as a vector of those equations.

```
clc;
clear all;
% Define the differential equation
syms y(x);
eqn=input(' Enter the differential equation')
yg= simplify( dsolve(eqn))
```

Problem 1:

$$\frac{d^3y}{dx^3} - 2\frac{d^2y}{dx^2} + 4\frac{dy}{dx} - 8y = 0$$

Output 1:

```
Enter the differential equation
diff(y,x,3) - 2*diff(y,x,2) + 4*diff(y,x)- 8*y == 0

eqn(x) =
4*diff(y(x), x) - 8*y(x) - 2*diff(y(x), x, x) + diff(y(x), x, x, x) == 0

yg =
C2*cos(2*x) + C1*exp(2*x) - C3*sin(2*x)
```

Problem 2:

$$\frac{d^2y}{dt^2} - \frac{dy}{dt} + 9y = 5e^{-2t}$$

Output 2:

```
Enter the differential equation
diff(y,x,2) - 6* diff(y,x)+ 9*y == 5*exp(-2*x)

eqn(x) =
9*y(x) - 6*diff(y(x), x) + diff(y(x), x, x) == 5*exp(-2*x)

yg =
exp(-2*x)/5 + C1*exp(3*x) + C2*x*exp(3*x)
```

Problem 3:

$$x^2\frac{d^2y}{dx^2} - 2x\frac{dy}{dx} - 12y = 6x + 5$$

Output 2:

```
Enter the differential equation
x^2*diff(y,x,2)-2*x*diff(y,x,1) -12*y == 6*x+5

eqn(x) =
```

```
x^2*diff(y(x), x, x) - 12*y(x) - 2*x*diff(y(x), x) == 6*x + 5

yg =
C1*x^(57^(1/2)/2 + 3/2) - (3*x)/7 + C2*x^(3/2 - 57^(1/2)/2) - 5/12
```

Exercise Problems

1. $(D^4 + 2D^3 + D)y = 0$
2. $(D^3 + 2D^2 + D)y = x^2e^{2x} + \sin^2 x$
3. $\frac{d^2y}{dx^2} + \frac{1}{x}\frac{dy}{dx} = \frac{12\log x}{x^2}$
4. $(3x+2)^2 y'' + 3(3x+2)y' - 36y = 8x^2 + 4x + 1$

Solving higher order ordinary differential Equation with initial condition

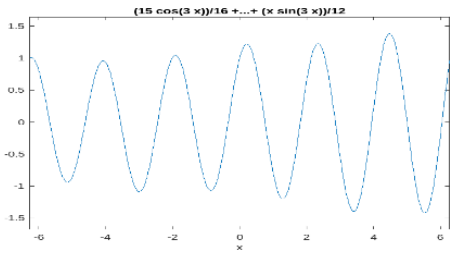
```
clc;
clear all;
close all;
% Define the differential equation
syms y(x);
eqn=input('enter the differential equation')
Dy = diff(y,x);
cond=input('Enter the Initial Condition')
yg= simplify( dsolve(eqn))
yp = simplify(dsolve(eqn,cond))
ezplot(yp)
```

Problem 1:

$y'' + 9y = \cos 2x.\cos x$ and given that $y(0) = 1$ and $y'(0) = 2$ and also plot the graph

Output 1:

```
enter the differential equation
diff(y,x,2) + 9*y == cos(2*x)*cos(x)

eqn(x) =
9*y(x) + diff(y(x), x, x) == cos(2*x)*cos(x)

Enter the Initial Condition
 [y(0)==1, Dy(0)==2]
cond =
[y(0) == 1, subs(diff(y(x), x), x, 0) == 2]
yg =
(7*cos(3*x))/144 + cos(x)/16 + (x*sin(3*x))/12 + C1*cos(3*x) - C2*sin(3*x)
yp =
(15*cos(3*x))/16 + (2*sin(3*x))/3 + cos(x)/16 + (x*sin(3*x))/12
```

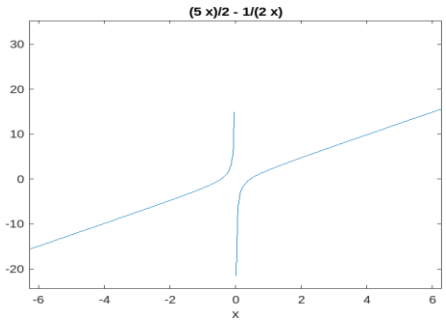


Problem 2:

$x^2y'' + xy' - y = 0$ and given that $y(1) = 2$ and $y'(1) = 3$ and also plot the graph

Output 2:

```
enter the differential equation
x^2*diff(y,x,2)+x*diff(y,x,1)-y == 0

eqn(x) =
x^2*diff(y(x), x, x) - y(x) + x*diff(y(x), x) == 0

Enter the Initial Condition
 [y(1)==2, Dy(1)==3]

cond =
[y(1) == 2, subs(diff(y(x), x), x, 1) == 3]

yg =
C2*x + C1/(2*x)

yp =
(5*x)/2 - 1/(2*x)
```



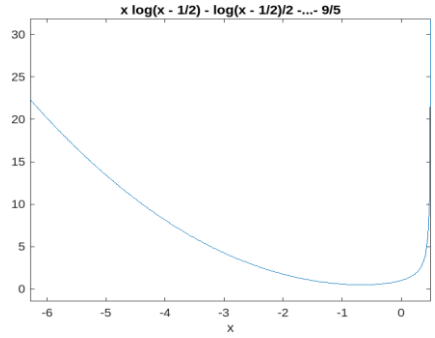
Problem 3:

$(2t-1)^2 y'' + (2t-1)y' - 2y = 8t^2 - 2t + 3$ and given that $y(0) = 1$ and $y'(0) = 2$ and also plot the graph

Output 3:

```
enter the differential equation
(2*x-1)^2*diff(y,x,2)+(2*x-1)*diff(y,x,1)-2*y == 8*x^2-2*x+3

eqn(x) =
(2*x - 1)^2*diff(y(x), x, x) - 2*y(x) + (2*x - 1)*diff(y(x), x) == 8*x^2 -
2*x + 3

Enter the Initial Condition
[y(0)==1, Dy(0)==2]

cond =
[y(0) == 1, subs(diff(y(x), x), x, 0) == 2]


yg =
x*log(x - 1/2) - log(x - 1/2)/2 - (4*x)/5 + C1*(x - 1/2) + (4*x^2)/5 -
(2*2^(1/2)*C2)/(3*(2*x - 1)^(1/2)) - 9/5


yp =
```

```
x*log(x - 1/2) - log(x - 1/2)/2 - (x - 1/2)*(2/3 - log(2) + pi*1i) - (4*x)/5
+ 37i/(15*(2*x - 1)^(1/2)) + (4*x^2)/5 - 9/5
```



Exercise Problems:

1. $(D^2+D)y = 2+2x+x^2,\ y(0)=8, y'(0)=-1$
2. $(D^3+2D^2+D)y = x^2e^{2x}+\sin^2 x$ and $y(1)=0, y'(2)=-1$
3. $x^3\frac{d^3y}{dx^3}+3x^2\frac{d^2y}{dx^2}+x\frac{dy}{dx}+y = x\log x$ and $y(2)=0, y'(2)=-1$
4. $(3t+1)^2y''+(3t+1)y'-2y = 8t^2-2t+3$ and $y(1)=0, y'(1)=-1$

Solving higher order ordinary differential Equation with boundary condition

```
clc;
clear all;
close all;
% Define the differential equation
syms y(x);
eqn=input('enter the differential equation')
cond=input('Enter the Boundary Condition')
yg= simplify( dsolve(eqn))
yp = simplify(dsolve(eqn,cond))
ezplot(yp)
```

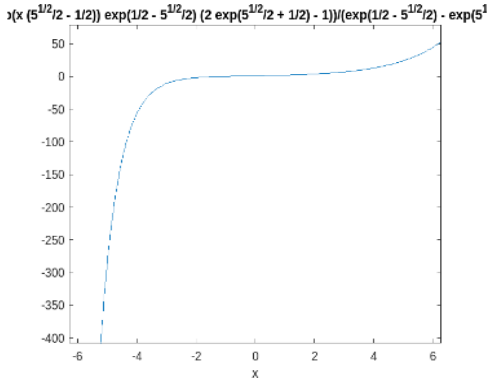
Problem 1:

$y'' = y$ with boundary conditions $y(0) = 1$ *and* $y(1) = 2$ and also plot the graph

Output 1:

```
enter the DE
diff(y(x), x) - y(x) + diff(y(x), x, x) == 0

eqn =
diff(y(x), x) - y(x) + diff(y(x), x, x) == 0

Enter the BC
 [y(0)==1, y(1)==2 ]

cond =
[y(0) == 1, y(1) == 2]

yg =
exp(-(x*(5^(1/2) + 1))/2)*(C1 + C2*exp(5^(1/2)*x))
```

```
yp =
- (exp(-x*(5^(1/2)/2 + 1/2))*(exp(5^(1/2)/2 + 1/2) - 2*exp(1/2 -
5^(1/2)/2)*exp(5^(1/2)/2 + 1/2)))/(exp(1/2 - 5^(1/2)/2) - exp(5^(1/2)/2 +
1/2)) - (exp(x*(5^(1/2)/2 - 1/2))*exp(1/2 - 5^(1/2)/2)*(2*exp(5^(1/2)/2 +
1/2) - 1))/(exp(1/2 - 5^(1/2)/2) - exp(5^(1/2)/2 + 1/2))
```

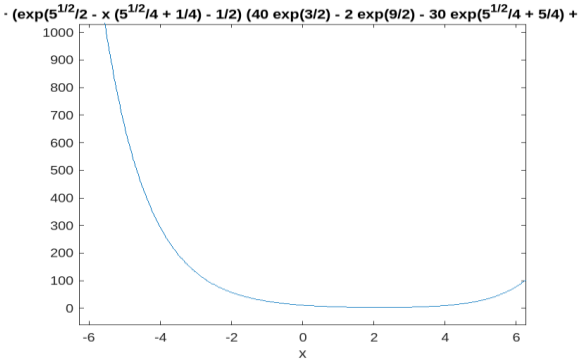


Problem 2:

$4y'' + 2y' - y = e^x$ with boundary conditions $y(2) = 3$ *and* $y(3) = 4$ and also plot the graph

Output 2:

```
enter the differential equation
 4*diff(y,x,2)+2*diff(y,x,1)-y == exp(x)

eqn(x) =
2*diff(y(x), x) - y(x) + 4*diff(y(x), x, x) == exp(x)

Enter the Boundary Condition
[y(2)==3, y(3)==4]

cond =
[y(2) == 3, y(3) == 4]

yg =
exp(x)/5 + C1*exp(-(x*(5^(1/2) + 1))/4) + C2*exp((x*(5^(1/2) - 1))/4)

yp =
exp((5*x)/4 + (5^(1/2)*x)/4 - x*(5^(1/2)/4 + 1/4))*(5^(1/2)/10 + 1/10) -
exp((5*x)/4 + (5^(1/2)*x)/4 - x*(5^(1/2)/4 + 1/4))*(5^(1/2)/10 - 1/10) -
(exp(x*(5^(1/2)/4 - 1/4) - 5^(1/2)/2 - 1/2)*(40*exp(3/2) - 2*exp(9/2) -
30*exp(5/4 - 5^(1/2)/4) + 2*exp(13/4 - 5^(1/2)/4)))/(10*(exp(1/4 - 5^(1/2)/4)
- exp(5^(1/2)/4 + 1/4))) + (exp(5^(1/2)/2 - x*(5^(1/2)/4 + 1/4) -
1/2)*(40*exp(3/2) - 2*exp(9/2) - 30*exp(5^(1/2)/4 + 5/4) + 2*exp(5^(1/2)/4 +
13/4)))/(10*(exp(1/4 - 5^(1/2)/4) - exp(5^(1/2)/4 + 1/4)))
```

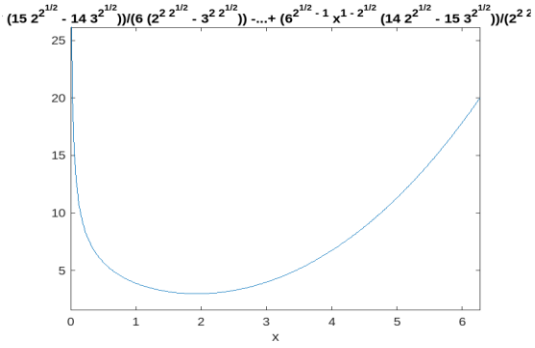


Problem 3:

$x^2y'' - xy' - y = 2x$ with boundary conditions $y(2) = 3$ *and* $y(3) = 4$ and also plot the graph

Output 3:

```
enter the differential equation
x^2*diff(y,x,2)-x*diff(y,x,1)-y == 2*x

eqn(x) =
x^2*diff(y(x), x, x) - y(x) - x*diff(y(x), x) == 2*x

Enter the Boundary Condition
[y(2)==3, y(3)==4]

cond =
[y(2) == 3, y(3) == 4]

yg =
C1*x^(2^(1/2) + 1) - x + C2*x^(1 - 2^(1/2))

yp =
(x^(2^(1/2) + 1)*(15*2^(2^(1/2)) - 14*3^(2^(1/2))))/(6*(2^(2*2^(1/2)) -
3^(2*2^(1/2)))) - x + (6^(2^(1/2) - 1)*x^(1 - 2^(1/2))*(14*2^(2^(1/2)) -
15*3^(2^(1/2))))/(2^(2*2^(1/2)) - 3^(2*2^(1/2)))
```



Exercise Problems

1. $(D^2 + D)y = 2 - 2x - x^2,\ y(0) = 8, y(2) = -1$
2. $(D^3 + 2D^2 - D)y = x^2e^{2x} - \sin^2 x$ and $y(1) = 0, y(2) = -1$
3. $x^3\frac{d^3y}{dx^3} - 3x^2\frac{d^2y}{dx^2} - x\frac{dy}{dx} + y = x + \log x$ and $y(2) = 0, y(2) = -1$
4. $(3t-1)^2 y'' + (3t-1)y' + 2y = 8t^2 + 2t + 3$ and $y(1) = 0, y(1) = -1$

Lab 6

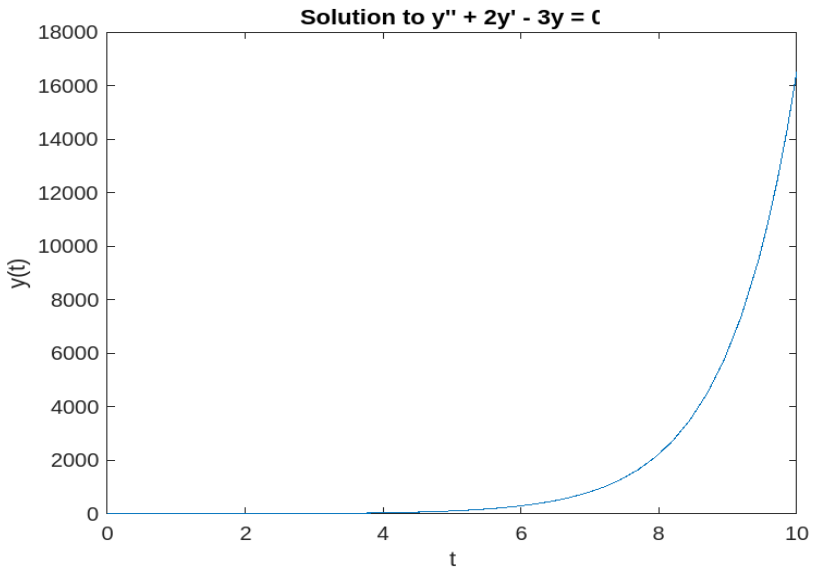
Solving Higher Order Ordinary Differential Equation Using ODE45 code

`[t,y] = ode45(odefun,tspan,y0)`, where `tspan = [t0 tf]`, integrates the system of differential equations `y'=f(t,y)` from `t0` to `tf` with initial conditions `y0`. Each row in the solution array y corresponds to a value returned in column vector `t`.

Program 1:

```
%Write a mat lab code for solving the differential equation  y'' + 2y' - 3y =
0
with initial conditions y(0) = 1 and y'(0) = 0 and also plot the graph using
ODE45 solver

clc;
eqn = @(t,y) [y(2); -2*y(2) + 3*y(1)];

% Define the initial conditions
t0 = 0;
y0 = [1; 0];

% Define the time interval for the solution
tspan = [0 10];

% Use the ODE45 solver to solve the differential equation
[t,y] = ode45(eqn, tspan, y0);

% Plot the solution
plot(t, y(:,1))
xlabel('t')
ylabel('y(t)')
title('Solution to y'''' + 2y'' - 3y = 0')
```

Output:

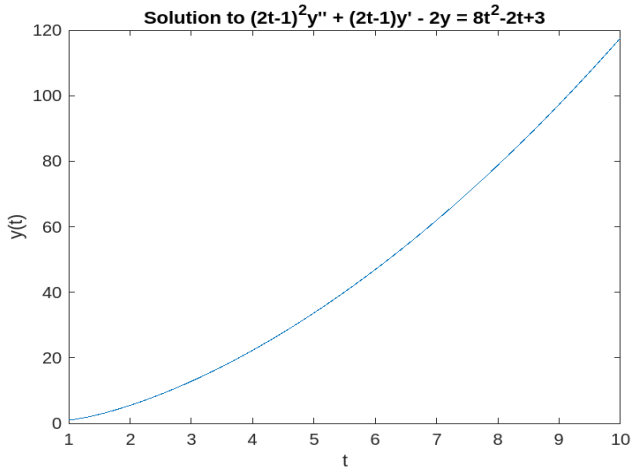


Program 2:

Write a mat lab code for solving the differential equation $(2t-1)^2 y'' + (2t-1)y' - 2y = 8t^2 - 2t + 3$ with initial conditions y(0) = 1 and y'(0) = 2 and also plot the graph using ODE45 solver

```
clc;
       clear all;
       close all;
eqn = @(t,y) [y(2); (8*t^2-2*t+3-(2*t-1)*y(2)+2*y(1))/(2*t-1)^2];

% Define the initial conditions
t0 = 0;
y0 = [1; 2];

% Define the time interval for the solution
tspan = [1 10];

% Use the ODE45 solver to solve the differential equation
[t,y] = ode45(eqn, tspan, y0);

% Plot the solution
plot(t, y(:,1))
xlabel('t')
ylabel('y(t)')

title('Solution to (2t-1)^2y'''' + (2t-1)y'' - 2y = 8t^2-2t+3')
```

Output:



Exercise Problems

1. $(D^3 + 2D^2 + D)y = x^2 e^{2x} + \sin^2 x \qquad y(2) = 0, y'(2) = -1$
2. $(D^2 - 2D + 4)y = e^x \cos x, \qquad y(1) = 0, y'(1) = -1$
3. $x^3 \frac{d^3y}{dx^3} + 3x^2 \frac{d^2y}{dx^2} + x\frac{dy}{dx} + y = x\log x \ , \ y(2) = 0, y'(2) = -1$

Lab 7

Interpolation

Newton's Forward Difference

Problem 01: Use Newton's Forward Difference formula and find the value of y at x = 1.5

X	1	2	3	4	5
Y	2	5	10	17	26

Program:

```
clc;
clear all;

syms p

%Defining x and y arrays
x = input('Input x values as an array : ');
y = input('Input y values as an array : ');
x0 = input('Enter the x for which y is to be found : ');

%Finding the length of x
n = length(x);

D = zeros(n,n);
D(:,1) = y;

for j = 2:n
    for i = 1:n-j+1
        D(i,j) = D(i+1,j-1)-D(i,j-1);
    end
end

h = x(2) - x(1);
u = (p-x(1))/h;
fx = @(p) y(1);
G = u;

for k = 1:n-1
    fx = fx+G*D(1,k+1);
    G = G*(u-k)/(k+1);
end

fx = simplify(fx)
fprintf('Value of f(x0) using Newtons forward interpolation formula is f(%f)
= %f',x0,subs(fx,p,x0))
```

Output:

```
Input x values as an array : [1,2,3,4,5]
Input y values as an array : [2,5,10,17,26]
Enter the x for which y is to be found : 1.5
fx =
p^2 + 1
Value of f(x0) using Newtons forward interpolation formula is f(1.500000) =
3.250000
```

Problem 02: Use Newton's Forward Difference formula and find the value of y at x = 0.3

X	0	1	2	3
Y	1	2	1	10

Newton's Backward Difference

Problem 01: Use Newton's Backward Difference formula and find the value of y at x = 4.5

X	1	2	3	4	5
Y	1	7	25	61	121

Program:

```
clc;
clear all;

syms p

%Defining x and y arrays
x = input('Input x values as an array : ');
y = input('Input y values as an array : ');
x0 = input('Enter the x for which y is to be found : ');


%Finding the length of x
n = length(x);

D = zeros(n,n);
D(:,1) = y;

for j = 2:n
    for i = n:-1:j
        D(i,j) = D(i,j-1)-D(i-1,j-1);
    end
end

h = x(2) - x(1);

u = (p-x(n))/h;
fx = @(p) y(n);
G = u;

for k = 1:n-1
    fx = fx+G*D(n,k+1);
    G = G*(u+k)/(k+1);

end

fx = simplify(fx)
fprintf('Value of f(x0) using Newtons Backward interpolation formula is f(%f) =
%f',x0,subs(fx,p,x0))
```

Output:

```
Input x values as an array : [1,2,3,4,5]
Input y values as an array : [1,7,25,61,121]
Enter the x for which y is to be found : 4.5
fx =
p^3 - p + 1
Value of f(x0) using Newtons Backward interpolation formula is f(4.500000) =
87.625000
```

Problem 02: Use Newton's Backward Difference formula and find the value of y at x = 84

X	40	50	60	70	80	90
Y	180	204	226	250	276	304

Newton's Divided difference

Problem 01: Use Newton's Divided difference formula and find the value of y at x = 1.6

X	1.2	2	2.5	3
y	1.36	0.58	0.34	0.20

Program:

```
clc;
clear all;

syms p
x = input('Input x values as an array : ');
y = input('Input y values as an array : ');
x0 = input('Enter the x for which y is to be found : ');

%Finding the length of x
n = length(x);

%Divided difference formula
for i = 1:n
   D(i,1) = y(i);
end

for i = 2:n
   for j = 2:i
      D(i,j)=(D(i,j-1)-D(i-1,j-1))/(x(i)-x(i-j+1));
   end
end
fx = @(p) D(n,n);

for i = n-1:-1:1
   fx = fx*(p-x(i)) + D(i,i);
end

fx = simplifyFraction(fx)
fprintf('Newtons iterated value using divided difference : f(%f) = %f \n',x0,
subs(fx,p,x0));
```

Output:

```
Input x values as an array : [1.2,2,2.5,3]
Input y values as an array : [1.36,0.58,0.34,0.20]
Enter the x for which y is to be found : 1.6
fx =
- 0.1004*p^3 + 0.9532*p^2 - 3.2379*p + 4.0464
Newtons iterated value using divided difference : f(1.600000) = 0.894600
```

Problem 02: Use Newton's Divided difference formula and find the value of y at x = 8

X	4	5	7	10	11
y	48	100	294	900	1210

Lagrange Interpolation

Problem 01: Use Lagrange Interpolation formula to find the value of y at x = 3.5

X	1	2	3	4	5
Y	4	9	16	25	36

Program:

```
clc;
clear all;
syms p
x = input('Input x values as an array : ');
y = input('Input y values as an array : ');
x0 = input('Enter the x for which y is to be found : ');

n = length(x);
fx = 0;
terms = [];
for k = 1 : n
    t=1;
    for i = 1 : n
        if i ~= k
            t = t * (p - x(i))/(x(k)-x(i));
        end
    end
    terms = [terms,t];
end

for i = 1 : n
    fx = fx + terms(i) * y(i);
end
disp('f(p)=');
disp(simplify(fx));

fprintf('Value of f(x0) using Lagrange interpolation is f(%f) =
%f',x0,subs(fx,p,x0))
```

Output:

```
Input x values as an array : [1,2,3,4,5]
Input y values as an array : [4,9,16,25,36]
Enter the x for which y is to be found : 3.5
f(p)=
(p + 1)^2
Value of f(x0) using Lagrange interpolation is f(3.500000) = 20.250000
```

Problem 02: Use Lagrange Interpolation to determine the value of y at x = 5 given that

X	1	3	7	11
Y	45	71	100	151

Lab 8

Numerical Integration

Simpson's 1/3rd Rule

Problem 01: Evaluate the integral $\int_0^{\pi} \sin(x)dx$ using Simpson's 1/3rd rule with n = 6.

Program:

```
syms x

% Lower Limit
a = 0;
% Upper Limit
b = pi;
% Number of Segments
n = 6;
% Declare the function
f =@(x) sin(x);
% h is the segment size
h = (b - a)/n;
% sums stores the summation of first and last segment
sums = f(a)+f(b);
% variables Odd and Even to store
% summation of odd and even terms respectively
Odd = 0;
Even = 0;
for i = 1:n-1
    xi = a+(i*h);
    if rem(i,2)==0
        Even = Even+f(xi);
    else
        Odd = Odd + f(xi);
    end
end

% Simpsons 1/3 Rule formula
I = (h/3)*(sums+4*Odd+2*Even);

disp('The approximation of above integral is: ');
disp(I);
```

Output:

```
The approximation of above integral is:
2.0009
```

Problem 02: Evaluate using Simpson's 1/3rd rule for the following data

x	0	1	2	3	4	5	6
y	1	4	9	16	25	36	49

Program:

```
x = [0,1,2,3,4,5,6];
y = [1,4,9,16,25,36,49];

% Number of Segments
p = length(x); %number of element in list is (no% of segments + 1)
n = p - 1;

% Lower Limit
a = x(1);

% Upper Limit
b = x(p);

% h is the segment size
h = x(2)-x(1);

% sums stores the summation of first and last segment
sums = y(1)+y(p);

% variables Odd and Even to store
% summation of odd and even terms respectively
Odd = 0;
Even = 0;
y(1) = []; %removing first and last element as they've already been used
y(n) = [];

for i = 1:n-1
    if rem(i,2)==0
        Even = Even +y(i)
    else
        Odd = Odd + y(i);
    end
end

% Simpsons 1/3 Rule formula
I = (h/3)*(sums+4*Odd+2*Even);

disp('The approximation of above integral is: ');
disp(I);
```

Output:

```
The approximation of above integral is:
114
```

Exercise Problems

1. **Evaluate the integral $\int_0^1 \frac{x}{1+x^2}\,dx$ using Simpson's 1/3rd rule**
2. **Evaluate Using Simpson's 1/3rd rule for the following data.**

X	0	1	2	3	4	5	6
Y	1	0.7071	0.4472	0.3162	0.2425	0.1961	0.1643

Simpson's 3/8th Rule

Problem 01: Evaluate the integral $\int_0^1 \frac{1}{1+x^2} dx$ with using Simpson's 3/8th Rule using n = 9

Program:

```
syms x

% Lower Limit
a = 0;

% Upper Limit
b = 1;

% Declare the function
f =@(x) 1/(1+x^2);

% Number of Segments in multiples of 3
n = 9;

if rem(n,3)~=0
    disp('number of segments should be a multiple of 3')
else


    % h is the segment size
    h = (b - a)/n;

    % sums stores the summation of first and last segment
    sums = f(a)+f(b);
    % variables Odd and Even to store
    % summation of odd and even terms respectively
    m = 0;
    k = 0;
    for i = 1:n-1
        xi=a+(i*h);
        if rem(i,3)==0
            m = m+f(xi);
        else
            k = k+f(xi);
        end
    end

    % Formula to calculate numerical integration
    % using Simpsons 1/3 Rule
    I = (3*h/8)*(sums+3*k+2*m);

    disp('The approximation of above integral is: ');
    disp(I);
end
```

Output:

```
The approximation of given integral is:
    0.7854
```

Problem 02: Evaluate using Simpson's 3/8th rule for the following data

X	1	1.5	2	2.5	3	3.5	4
Y	3	-2	1	0	2	3	-3

Program:

```
x = [1,1.5,2,2.5,3,3.5,4];
y = [3,-2,1,0,2,3,-3];

% Number of Segments
p = length(x);
n = p - 1;

% Lower Limit
a = x(1);

% Upper Limit
b = x(p);

if rem(n,3)~=0
    disp('number of segments should be a multiple of 3')
else
    % h is the segment size
    h = x(2)-x(1);

    % sums stores the summation of first and last segment
    sums = y(1)+y(p);

    sm3 = 0;
    smo = 0;

    y(1) = []; %removing first and last element as they've already been used
    y(n) = [];
    for i = 1:n-1
        if rem(i,3)==0
            sm3=sm3+y(i);
        else
            smo=smo+y(i);
        end
    end
    I = (3*h/8)*(sums+3*smo+2*sm3);
    disp('The value of the above integral is')
    disp(I)
end
```

Output:

```
The value of the given integral is
2.2500
```

Exercise Problems

1. **Evaluate the integral $\int_0^1 \frac{1}{x+1} dx$ with using Simpson's 3/8th Rule using n = 9.**

2. Evaluate using Simpson's 3/8th rule for the following data:

X	0	1/6	2/6	3/6	4/6	5/6	1
Y	2	1.99	1.92	1.78	1.54	1.26	1

Lab 9

Gauss-Seidel method

Program:

```
clc;
clear all
n = input("enter the number of varibles:");
a = input("enter the co-efficient matrix:");
b = input("enter the constant matrix:");
oldx = input("initial guesses:");
x = oldx;
eps = input("enter the error tolerance:");
maxitr = input("enter the maximum number of iterations:");
for k = 1:maxitr
    for i = 1:n
        sum = 0;
        for j = 1:n
            if i~=j then
                sum = sum+a(i,j) * x(j);
            end
        end
        x(i) = (b(i)-sum)/a(i,i);
        disp(x);
    end
    if abs(max(x)-max(oldx))<=eps
        fprintf("the solutions are :");
        disp(x);
        fprintf("the number of iterations involved:")
        disp(k);
        break;
    else
        oldx = x;
    end
end
```

Output:

```
enter the number of varibles:3
enter the co-efficient matrix:[5 2 1;1 4 2;1 2 5]
enter the constant matrix:[12;15;20]
initial guesses:[1;0;3]
enter the error tolerance:0.00001
enter the maximum number of iterations:1000
the solutions are:
      1.0000
    2.0000
    3.0000
the number of iterations involved:     8
```

Exercise Problems

Use Gauss Siedel method to solve the following system of equations:

1. $5x+y+2z=4$; $3x+5y+z=7$; $x+y+5z=3$.
2. $10x+y+z=12$; $x+10y+z=12$; $x+y+10z=12$.

3. $x+y+54z=110$; $27x+6y-z=85$; $6x+15y+2z=72$.
4. $5x-y=9$; $x-5y+z=4$; $y-5z=6$.
5. $28x+4y-z=32$; $2x+17y+4z=35$; $x+3y+10z=24$.
6. $x+y+54z=110$; $27x+6y-z=85$; $6x+15y+2z=72$.
7. $x+8y+2z=-4$; $10x+2y+z=59$; $2x-y+20z=74$.

Rayleigh power method

Program:

```
clc;
clear all
n = input("enter the number of varibles:");
a = input("enter the  matrix:");
oldx = input("initial guesses:");
%x = oldx;
eps = input("enter the error tolerance:");
maxitr = input("enter the maximum number of iterations:");

for k = 1:maxitr

    C=a*oldx;
        m=max(abs(C));
        x=(C./m);

    if abs((x)-(oldx))<=eps
        fprintf("The Eigen vector:");
        disp(x);
        fprintf("The largest Eigen value:");
        disp(m);
        fprintf("the number of iterations involved:")
        disp(k);
        break;
    else
        oldx = x;
    end

end
```

Output:

```
enter the number of varibles:3
enter the matrix: [2 0 1;0 2 0;1 0 2]
initial guesses:[1;0;0]
enter the error tolerance:0.00001
enter the maximum number of iterations:1000
The Eigen vector:
1.0000
 0
1.0000
The largest Eigen value:    3.0000
the number of iterations involved:    12
```

Exercise Problems

a) $\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$ by taking $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$ as the initial approximation.

b) $\begin{bmatrix} 6 & -2 & 2 \\ -2 & 3 & -1 \\ 2 & -1 & 3 \end{bmatrix}$ by taking $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$ as the initial approximation.

Lab 10

Solution of System of ODEs by matrix method

Program:

```
clc;
clear all
A=input("Enter matrix:");
n=input("Enter dimension of equation:");
[V,D]= eig(A);
disp(V);
disp(D);
I=inv(V);
X0=[0;1];
Y0=I*X0;
syms y(t)
for i=1:n
    for j=1:n
        if i==j
            ode=diff(y,t)==D(i,j)*y
            cond = y(0)==Y0(j);
            Y(j)=dsolve(ode,cond);
        end
    end
end
disp(Y);
X=V*Y';
disp(X')
```

Output:

```
Enter matrix: [4 2; -1 1]
Enter dimension of equation:2
    0.8944   -0.7071
   -0.4472    0.7071

     3     0
     0     2

Y=[C1*exp(3*t), C1*exp(2*t)]

X=[(2*5^(1/2)*C1*exp(3*t))/5 - (2^(1/2)*C1*exp(2*t))/2,
(2^(1/2)*C1*exp(2*t))/2 - (5^(1/2)*C1*exp(3*t))/5]
```

Exercise Problems

Solve the following system of first order linear differential equations using matrix method.

a) $x_1' = 3x_1, x_2' = 5x_1 - 4x_2; x_1(0) = 1, x_2(1) = -1.$

b) $x_1' = x_1 + x_2, x_2' = 3x_1 + 3x_2; x_1(0) = 0, x_2(0) = 1.$

c) $x_1' = 3x_1, x_2' = x_1 + x_2.$