# Chapter-IV

## Introduction:

MATLAB is basically a numerical system, but the addition of a symbolic toolbox has transformed MATLAB to a more powerful tool in engineering problem solving. When doing symbolic mathematics, the result of evaluating an expression is generally another expression. By keeping the variables unknown throughout consecutive steps of calculations, the toolbox yields exact answers with more accuracy than numerical approximation methods. You can tell MATLAB to manipulate expressions that let you compute with mathematical symbols rather than numbers. The symbolic toolbox is a symbolic-math package with extensive computational capabilities such as integration, differentiation, series expansion, solution of algebraic and differential equations, to name just a few. The symbolic toolbox is based on the MAPLE kernel as an engine to handle symbolic mathematics.

## Declaring symbolic variables:

All symbolic variables in MATLAB must be defined with the **syms** or **sym** commands before they are used. Once the symbolic variables are defined, they can be used in expressions in the same manner that numeric variables are used. For example,

```
>>syms x;                        %create a symbolic object x
>>x=sym('x');                    %alternative way of creating a symbolic object
>>syms x y z;                    %create several symbolic objects at once
```

As a result expressions with these variables will be treated as symbolic expressions.

We can assign an expression to a variable using the assignment operator (=). As an example, let us assign the expression $x*\sin\left(x\right)+e^{-\frac{x}{2}}$ to the variable f. This is accomplished as follows:

```
>>syms x;
>>f=x*sin(x)+exp(-x/2);
```

While we're at it, let us go ahead and define another variable g as $x*\cos\left(x\right)-e^{-\frac{x}{2}}$. We can now add the two functions in the usual way,

```
>>syms x;                        %define the symbolic variable x
>>f=x*sin(x)+exp(-x/2);          %define the variable f
>>g=x*cos(x)-exp(-x/2);          %define the variable g
>>h=f+g                          %add the variables f and g
```

## Plotting:

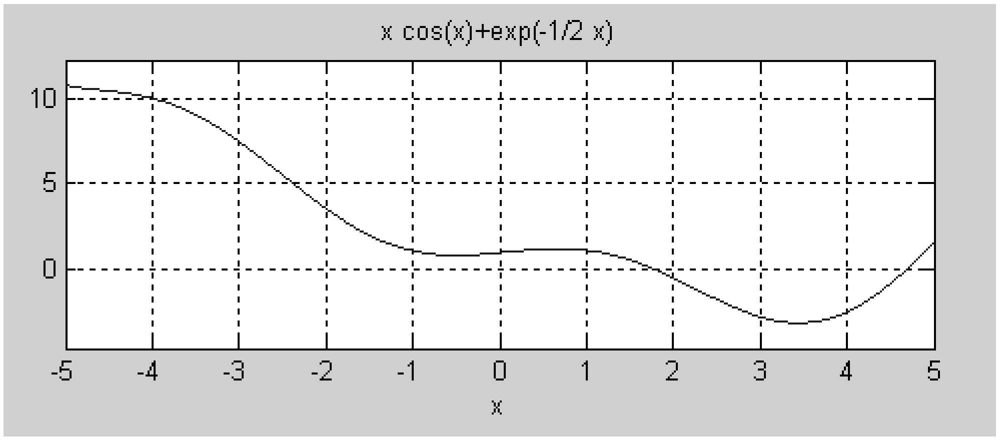
MATLAB has a built-in plot command called **ezplot** that will plot symbolic functions with a single variable over some specified range.

**Syntax:**

```
>>ezplot(y, [ymin, ymax])          %plot the expression in y on the specified interval
```

*Practice:*

```
>>syms x;
>>f=x*cos(x)+exp(-x/2);
>>ezplot(f, [-5,5]);grid
```



The function **ezsurf** provides plotting of 3-D colored surface over a specified domain.

**Syntax:**

```
>>ezsurf(f, domain)
```

*Practice:*

```
>>syms x y;
>>f=(1/(2*pi))*exp(-(x^2+y^2)/2);
>>domain=[-3 3 –3 3];
>>ezsurf(f,domain);
>>xlabel(‘x’)
>>ylabel(‘y’)
>>title(‘Bivariate gaussian density function’)
```

The function **subs** allows you to substitute a number or a symbol to a symbolic expression.

*Practice:*

Evaluate the function $x * \sin\left(x\right) + e^{-\frac{x}{2}}$ at x = 2 and x=v.

```
>>syms x;                      %define the symbolic variable x
>>f=x*sin(x)+exp(-x/2);        %define the function f
>>subs(f,x,2);                 %evaluate f at x=2
>>subs(f,x,v);                 %evaluate f at x=v
```

## Partial Fraction Expansion:

The **extended** symbolic toolbox provides access to the **MAPLE** kernel, which has a built-in command to perform partial fraction expansion.

*Practice:*

Find the partial fraction decomposition of

$$f(x)=\frac{x}{x^2-3x+2}$$

```
>>maple convert(x/(x^2-3*x+2),parfrac,x)
```

*ans*: $-\frac{1}{x-1}+\frac{2}{x-2}$

## Differentiation:

To differentiate an expression with respect to an independent variable, we use the function **diff**.

**Syntax:**

```
>>diff(f,x);                %return the derivative of f with respect to x
>>diff(f,x,n);              %return the nth derivative of f with respect to x
```

*Practice:*

```
>>syms x;                   %define the symbolic variable x
>>f=x*sin(x)+exp(-x/2);     %define the symbolic expression f
>>g=diff(f,x)               %differentiate f with respect to x
>>h=diff(f,x,3)             %differentiate f three times with respect to x
```

*Practice:*
Find the first and second derivative of the function $f(x)=\cos(ax)$

```
>>syms a x;                 %create symbolic variables
>>f=cos(a*x);               %define a symbolic function
>>fprime=diff(f,x)          %differentiate f(x) with respect to x


fprime =

-sin(a*x)*a

>>diff(f,a);                %differentiate the function f with respect to a
>>fdoubleprime=diff(f,x,2)  %calculate the second derivative with respect to x

fdoubleprime =
-cos(a*x)*a^2
```

*Practice*

Find the derivative of the following function and then evaluate it at x=7

$$f(x) = \frac{2x^2 - 3x + 1}{x^3 + 2x^2 - 9x - 18}$$

```
>>syms x;
>>f=(2*x^2-3*x+1)/(x^3+2*x^2-9*x-18);
>>de=diff(f,x)

de =

(4*x-3)/(x^3+2*x^2-9*x-18)-(2*x^2-3*x+1)/(x^3+2*x^2-9*x-18)^2*(3*x^2+4*x-9)

>>simplify(de)

ans =

-(2*x^4-6*x^3+15*x^2+76*x-63)/(x^3+2*x^2-9*x-18)^2

>>x=7; eval(df)

ans =

   -0.0305
```

*Practice:*

Find the first and second derivatives of $f(x) = x^4 + x^3 + x^2 + x + 1$

```
>>syms x;
>>f=x^4+x^3+x^2+x+1;
>>fprime=diff(f,x)

fprime =

4*x^3+3*x^2+2*x+1

>>fdoubleprime=diff(f,x,2)

fdoubleprime =

12*x^2+6*x+2
```

*Practice:*

Calculate the derivative of the function $f(x)=\sqrt{1-x^2}$ and evaluate the value of the derivative at x=2.

```
>>syms x;
>>f=sqrt(1-x^2);
>>fprime=diff(f,x)

fprime =

-1/(1-x^2)^(1/2)*x
```

This result can be expressed in a more readable form using the **pretty** command, as follows:

```
>>pretty(fprime)
>>val=subs(fprime,x,2)

val =

  -0.0000 + 1.1547i
```

**Integration:**

If f is a symbolic expression, then int(f, var) returns another symbolic expression, representing the indefinite integral of exp with respect to var. Definite integration can also be carried out by specifying the interval over which the integral is to be taken.

**Syntax:**

```
>>syms x;                   %declare the symbolic variable x
>>int(f,x);                 %indefinite integral of f
>>int(f,a,b);               %definite integral of f
```

*Practice:*

```
>>clear                     %clear variables
>>syms x;                   %declare the symbolic variable x
>>int(sin(x),0,pi/2)        %evaluate the integral of a sine over [0,pi/2]
```

*Practice:*

```
>>syms x;                   %define the symbolic variable x
>>f=x*cos(x^2);             %define a symbolic expression f
>>int(f,x)                  %compute the indefinite integral of f
```

Practice:

```
>>syms x;                   %define the symbolic variable x
>>int(x*cos(x),x)           %compute the indefinite integral of x*cos(x)
```

*Practice:*

Expand the following function in partial fraction expansion

```
>>syms x;
>>f=x/(x^2+5*x+6);
>>pfe=diff(int(f))

pfe =

3/(x+3)-2/(x+2)
```

*Practice:*

Evaluate the following integrals

1. $\int_0^1 xe^x dx$
2. $\int_0^1 xe^x \sin\left(x\right) dx$

```
>>syms x;
>>f1=x*exp(x);
>>int(f1,0,1)

ans =

1

>>f2=x*exp(x)*sin(x);
>>int(f2,0,1)

ans =

1/2*exp(1)*sin(1)-1/2

>>numeric(ans)

ans =

   0.6437
```

**Limits:**

**Syntax:**

```
>>limit(expr, var, a);
```

Computes the limiting value of **exp** as **var** approaches **a**.

*Practice:*

Find the limit of the function $f(x)=5abx+3e^{ax+b}$

1. $x \to 0$
2. $x \to 1$ from the left
3. $x \to 1$ from the right

```
>>syms x a b;
>>f=5*a*b*x+3*exp(a*x+b);
>>limit(f,x,0)

ans =

3*exp(b)

>>limit(f,x,1,'left')

ans =

5*a*b+3*exp(a+b)

>>limit(f,x,1,'right')

ans =

5*a*b+3*exp(a+b)
```

**Symbolic algebra:**

Certain commands from the symbolic toolbox allow the manipulation of algebraic expressions. You can simplify, expand, and factor expressions, find the coefficients of a polynomial, expand an expression into a series.

The function **expand** multiplies out products and powers. The function **factor** does essentially the inverse of expand, and the function **simplify** attempts to find the form of an expression with the smallest number of parts. The function **collect** combines terms of a polynomial.

*Practice:*

```
>>syms x;                  %define a symbolic variable
>>f=(x-2)^2+(x-3)^3;       %define a symbolic expression f
>>expand(f)                %expand the symbolic expression f

f =

-8*x^2+23*x-23+x^3
```

*Practice:*

```
>>syms x;                  %define a symbolic variable
```

```
>>f=x^4-1;                    %define a symbolic expression f
>>factor(f)                   %factor out the symbolic expression f

f =

(x-1)*(x+1)*(x^2+1)
```

*Practice:*

Factor the following 8-th order polynomial

$$f(x) = x^8 - 41x^4 + 400$$

```
>>syms x;
>>f=x^8-41*x^4+400;
>>factor(f)

ans =

(x-2)*(x+2)*(x^2-5)*(x^2+5)*(x^2+4)
```

*Practice:*

```
>>syms x;                     %define a symbolic variable
>>f=(x-1)/(x^2-1);            %define a symbolic expression f
>>f=simplify(f)               %simplify the symbolic expression f

f =

1/(x+1)
```

*Practice:*

```
>>syms x;                     %define a symbolic variable
>>f=(x-3)*(x-5)*(x-1);        %define a symbolic expression f
>>f=collect(f)                %combine terms of f

f =

x^3-9*x^2+23*x-15
```

## Solving algebraic equations:

The **solve** command provides the symbolic solution of algebraic equations. It sets the symbolic expression equal to zero before solving it.

**Syntax:**

```
>>solve(equ, x)               %solve equ with respect to the variable x
```

```
>>solve(equ1,equ2, x,y)            %solve system of equations with respect to x and y
```

*Practice:*

Find the roots of the quadratic equation $x^2 - 5x + 4 = 0$

```
>>syms x;                            %define a symbolic variable
>>equ=x^2-5*x+4;                     %define the quadratic equation
>>sol=solve(equ)                     %compute the roots of the equation

sol =

[ 1]
[ 4]
```

*Practice:*

Find the general solution of the quadratic equation $a * x^2 + b * x + c$ .

```
>>syms a b c x;                %define symbolic objects
>>solve(a*x^2+b*x+c,x)         %solve the quadratic equation with respect to x

ans =

[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

The solve command can also be used to solve several equations, as follows:

```
>>syms x y;
>>[x  y]=solve(x^2+x*y+y-3,x^2-4*x+3,x,y)

x =

[ 1]
[ 3]


y =

[    1]
[ -3/2]
```

*Practice:*

Find the absolute maximum and minimum values for the function

$f(x) = x^4 - 4x^3 + 2x^2 + 4x + 2$ on the interval [0,4]

```
>>syms x;
```

```
>>f=x^4-4*x^3+2*x^2+4*x+2;
>>sol=solve(diff(f,x))

sol =

[         1]
[ 2^(1/2)+1]
[ 1-2^(1/2)]

>>numeric(sol)

ans =

    1.0000
    2.4142
   -0.4142
```

Of these three numbers only two are in the interval [0,4]. We compute the value of the function at these numbers as well as the endpoints of the interval [0,4].

```
>>subs(f,[0 1 2.4142 4])

ans =

    2.0000    5.0000    1.0000   50.0000
```

Clearly, the absolute maximum of the function f(x) is 50 and the absolute minimum is 1.

*Practice:*

Solve the following system of linear equations:

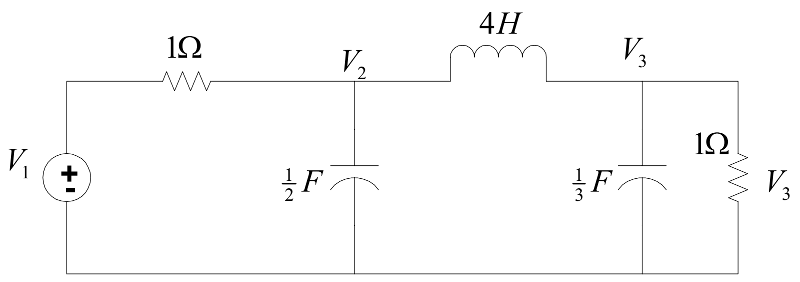
$$\begin{cases} 3x+2y=5 \\ x-5y=3 \end{cases}$$

```
>>syms x y;                          %define symbolic variables
>>equ1=3*x+2*y-5;                    %define the first equation
>>equ2=x-5*y-3;                      %define the second equation
>>sol=solve(equ1,equ2,x,y)           %solve for x and y

sol =

    x: [1x1 sym]
    y: [1x1 sym]

>>sol.x                              %return the value of x

ans =

31/17
```

```
>>sol.y                                    %return the value of y

ans =

-4/17
```

*Practice:*

For the given circuit, determine the transfer function.



*Solution:*

We shall use nodal analysis to derive the transfer function of the given circuit. By inspection, we obtain,

$$\frac{V_2 - V_1}{1} + \tfrac{1}{2} s V_2 + \frac{V_2 - V_3}{4s} = 0$$

$$\frac{V_3 - V_2}{4s} + \tfrac{1}{3} s V_3 + \frac{V_3}{1} = 0$$

Now, we are ready to use MATLAB to solve for V2 and V3:

```
>>syms V1  V2  V3  s;
>>equ_1=(V2-V1)+1/2*s*V2+(V2-V3)/(4*s);
>>equ_2=(V3-V2)/(4*s)+1/3*s*V2+V3;
>>[V2,V3]=solve(equ_1,equ_2,V2,V3)

V2 =

2*V1*(3+4*s^2+12*s)/(12+20*s^2+29*s+4*s^3)


V3 =

6*V1/(12+20*s^2+29*s+4*s^3)
```

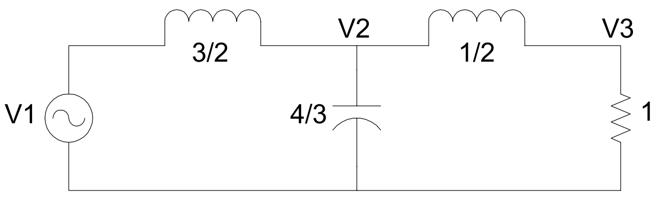
From the expression of V3 we deduce the transfer function of the given circuit:

$$H(s)=\frac{V_3(s)}{V_1(s)}=\frac{6}{4s^3+20s^2+29s+12}$$

*Practice:*

For the circuit shown below,

1. Find the transfer function
2. Sketch the magnitude and phase responses



Using nodal analysis, we obtain the following equations:

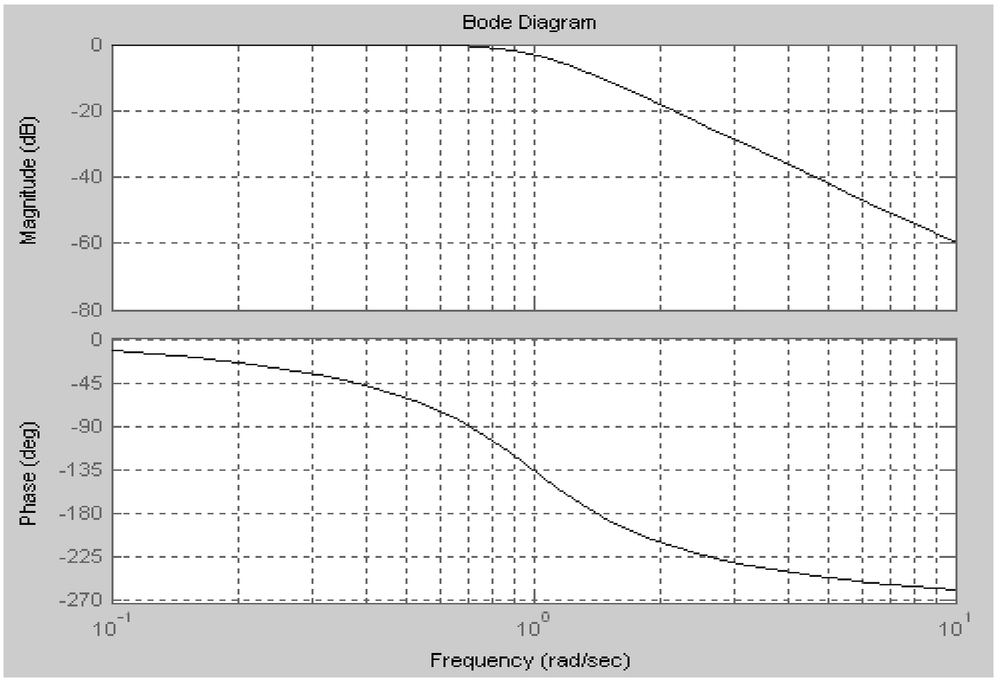
Node-2: $-\frac{2}{3s}V_1-\frac{2}{s}V_3+\left(\frac{2}{3s}+\frac{4s}{3}+\frac{2}{s}\right)V_2=0$

Node-3: $-\frac{2}{s}V_2+\left(1+\frac{2}{s}\right)V_3=0$

```
>>syms V1 V2 V3 s;
>>f1=-2/(3*s)*V1-2/s*V3+V2*(4*s/2+2/(3*s)+2/s);
>>f2=V3*(1+2/s)-2/s*V2;
>>sol=solve(f1,f2,V2,V3)

sol =

   v2: [1x1 sym]
   v3: [1x1 sym]

>>sol.V3

ans =

v1/(1+s^3+2*s^2+2*s)

>>H=sol.V3/V1

H =

1/(1+s^3+2*s^2+2*s)
```

The transfer function is still a symbolic expression. Before plotting the frequency response we need to convert it in numerical form.

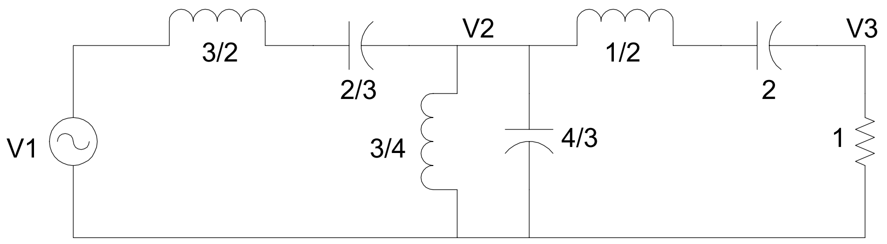
```
>>num=[1];
>>den=[1 2 2 1];
>>T=tf(num,den)

Transfer function:
         1
---------------------
s^3 + 2 s^2 + 2 s + 1

>>bode(T)
```



*Practice:*

For the circuit shown below,

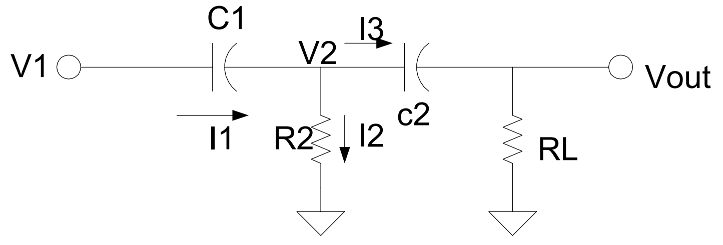
1. Obtain the transfer function
2. Sketch the magnitude and phase responses



```
>>syms V1 V2 V3 s;
>>f1=V3*(1+2*s/(s^2+1))-V2*2*s/(s^2+1);
>>f2=V2*(2*s/(s^2+1)+2*s/(3*(s^2+1))+4*(s^2+1)/(3*s)-V1*2*s/(3*(s^2+1))-V3*2*s/(s^2+1);
>>sol=solve(f1,f2,V2,V3)
>>H=sol.V3

sol =

   v2: [1x1 sym]
   v3: [1x1 sym]


ans =

v1*s^3/(5*s^4+5*s^2+5*s^3+s^6+2*s^5+1+2*s)

>>H=sol.V3/V1

H =

s^3/(5*s^4+5*s^2+5*s^3+s^6+2*s^5+1+2*s)

>>num=[1 0 0 0]; den=[1 2 5 5 5 2 1];
>>T=tf(num,den)

Transfer function:
                   s^3
---------------------------------------------
s^6 + 2 s^5 + 5 s^4 + 5 s^3 + 5 s^2 + 2 s + 1

>>bode(T)
```

*Practice:*

Find the transfer function of an RC-RC ladder circuit depicted below



We assume the output voltage Vout=1 v, and work back to find the input voltage V1.

```
>>syms V1 V2 Vout I1 I2 I3 R2 RL C1 C2 s w H;
>>Vout=1;
>>I3=Vout/RL;
>>V2=Vout+I3/(s*C2);
>>I2=V2/R2;
>>I1=I2+I3;
>>V1=V2+I1/(s*C1);
>>H=Vout/V1;


H =

1/(1+1/RL/s/C2+(1/RL+(1+1/RL/s/C2)/R2)/s/C1)
```

To simplify the expression further we separate the numerator and denominator using the **numden** command, and then collect like terms in the denominator.

```
>>[N,D]=numden(H)

N =

RL*s^2*C2*R2*C1


D =

RL*s^2*C2*R2*C1+s*R2*C1+s*C2*R2+RL*s*C2+1

>>D=collect(D,s)

D =

RL*s^2*C2*R2*C1+(R2*C1+C2*R2+RL*C2)*s+1

>>pretty(N), pretty(D)
```

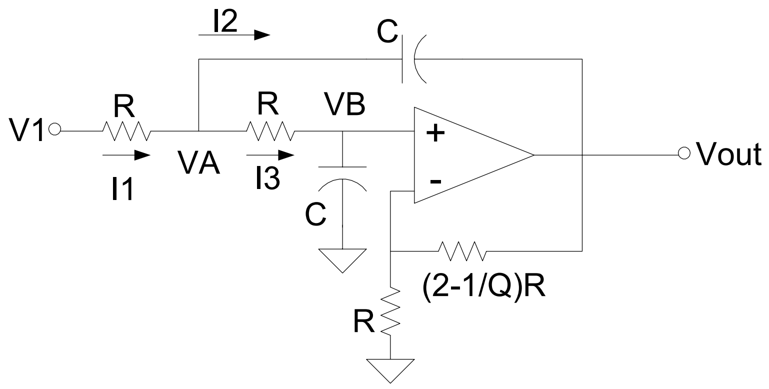
Lastly, we evaluate the frequency response H(jw)

```
>>pretty(subs(N,s,j*w)), pretty(subs(D,s,j*w))
```

_answer_: $H(j\omega)=\dfrac{-R1C1R2C2\omega^2}{-R1C1R2C2\omega^2+j\left(R2C2+C1R2+R1C1\right)\omega+1}$

_Practice:_

Find the transfer function of the active lowpass filter depicted below.



```
>>syms V1 VB VA Vout I1 I2 I3 R C Q s w H
>>Vout=1;
>>VB=Vout*R/(R+(2-1/Q)*R);
>>VB=simplify(VB);
>>I3=VB*s*C;
>>VA=VB+R*I3;
>>I2=(VA-Vout)*s*C;
>>I1=I2+I3;
>>V1=VA+I1*R;
>>H=Vout/V1

H =

(3*Q-1)/(Q+s^2*C^2*R^2*Q+s*C*R)

>>[N,D]=numden(H)


N =

3*Q-1


D =

Q+s^2*C^2*R^2*Q+s*C*R
```

```
>>pretty(N), pretty(collect(D,s))
>>pretty(N), pretty(subs(collect(D,s),s,j*w))
```

answer: $H(j\omega) = \dfrac{3Q-1}{-\omega^2C^2R^2 + j\omega CR + Q}$

**Fourier transformation:**

The function **fourier(f,t,w)** returns the Fourier transform of an expression f of the variable t, resulting in an expression of the variable w.

*Practice:*

```
>>syms  t w;
>>f=exp(-4*pi*t^2);
>>F=fourier(f,t,w)

F =

1/4*4^(1/2)*exp(-1/16*w^2/pi)
```

The function **ifourier(F,w,t)** returns the inverse Fourier transform of F of the variable w.  The resulting expression is given in terms of the variable t.

**Laplace transformation:**

The function **laplace(f, t, s)** returns the Laplace transform of f of the variable t.  The resulting expression is given in terms of  the variable s.

*Practice:*

```
>>syms t s;                    %specify the symbolic variables
>>F=laplace(exp(-2*t),t,s)     %compute Laplace transform
```

*Practice:*

Find the Laplace transform of $t^3\sin(2t)$

```
>>syms s t;                    %define symbolic variables
>>Y=laplace(t^3*sin(2*t),t,s)  %compute the Laplace transform

Y =

6/(s^2+4)^2*sin(4*atan(2/s))
```

The function **ilaplace(F,s,t)** returns the inverse Laplace transform of F of the variable s.  The resulting expression is given in terms of  the variable t.

*Practice:*

```
>>syms t s;                    %specify symbolic variables
>>f=ilaplace(1/(s+2),s,t)      %compute the inverse Laplace transform
```

You make the Laplace transform look better by using the **pretty** function.

```
>>pretty(Y)                    %make Y look better
```

**<u>Z transform:</u>**

The function **ztrans(f, n, z)** returns the z transform of the sequence f[n] of the variable n. The resulting expression is given in terms of the variable z.

*<u>Practice:</u>*

```
>>syms n  z ;                  %specify symbolic variables
>>F=ztrans(n,n,z)              %compute the z transform of f[n]=n
```

The function **iztrans(F,z,n)** returns the inverse z transform of F of the variable z. The resulting expression is given in terms of the variable n.

**<u>Taylor series:</u>**

The function **taylor(f, a,n)** returns a truncated Taylor series expansion of the function f about the value x=a, up to order n. Only Taylor functions of one variable are possible.

*<u>Practice:</u>*

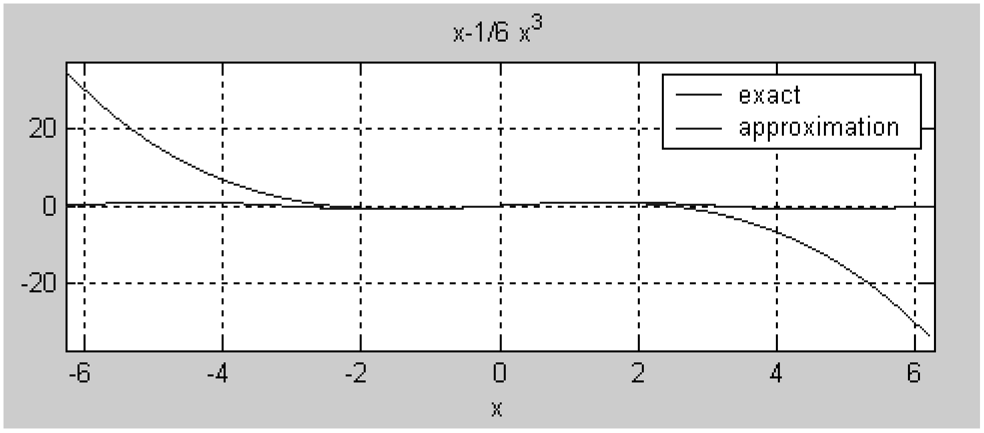
```
>>syms x;                      %define the symbolic variable x
>>f=sqrt(x+4);                 %define the symbolic function f(x)
>>y=taylor(f,0,4)              %give the first four terms of Taylor series

y =

2+1/4*x-1/64*x^2+1/512*x^3

>>%compare the functions f(x) and y(x)
>>ezplot(f, [-6,6])            %plot the function f  versus x
>>hold on                      %hold the first plot
>>ezplot(y,[-6,6])             %plot the function y versus x
>>grid                         %add grid to plot
>>hold off
```

*<u>Practice:</u>*

Obtain the Maclaurin series approximation to $f(x) = \sin(x)$

```
>>syms x;                      %define a symbolic variable
>>f=sin(x);                    %define the function f(x)
>>explot(f,[-2*pi,2*pi]);      %plot the given function f(x)
```

```
>>y=taylor(f,4,0);              %calculate the Maclaurin approximation
>>hold on                       %hold previous plot
>>ezplot(y,[-2*pi,2*pi])        %plot Maclaurin series
>>legend('exact','approximation')
>>grid
>>hold off
```



```
y =

x-1/6*x^3
```

*Practice:*

Obtain the Maclaurin polynomial of order 6 for the function $f\left(x\right)=e^{x}$

```
>>syms x;
>>f=exp(x);
>>y=taylor(f,0,6)

y =

1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5

>>pretty(y)
```

**MATRICES:**

Symbolic matrices and vectors are arrays whose elements are symbolic expressions.

*Practice:*

```
>>syms a  b  c;                 %define symbolic variables
>>A=[a  b  c;  b  c  a;  c  a  b];  %specify the symbolic matrix A
>>h=size(A)                     %provide the size of matrix A
```

The determinant and inverse of symbolic matrices are computed via the functions **det** and **inv**, respectively.

*Practice:*

```
>>syms a  b  c  d;              %define symbolic objects
>>A=[a  b;  c  d];              %specify the symbolic matrix A
>>d=det(A)                      %compute the determinant of A

d =

a*d-b*c

>>C=inv(A)                      %compute the inverse of A

C =

[  d/(a*d-b*c), -b/(a*d-b*c)]
[ -c/(a*d-b*c),  a/(a*d-b*c)]
```

The eigenvalues and eigenvectors of symbolic matrices can be found using function **eig.**

**Syntax:**

```
>>E=eig(A)                      %provide the eigenvalues of matrix A
>>[V, D]=eig(A)                 %compute eigenvalues and eigenvectors of A
```

To generate a symbolic diagonal matrix use the **diag** command.

```
>>syms a b c;                   %define symbolic objects
>>D=diag([a b c])               %create a diagonal matrix D

D =

[ a, 0, 0]
[ 0, b, 0]
[ 0, 0, c]
```

**Symbolic differential equations:**

MATLAB can be used to find symbolic solutions to ordinary differential equations, with or without initial conditions.  The function **dsolve** returns the symbolic solutions to ordinary differential equations.  The letter D is used to indicate the operation of differentiation

$$Dy = \frac{dy}{dt}$$

$$D2y = \frac{d^2y}{dt^2}$$

$$D3y = \frac{d^3y}{dt^3}$$

Initial conditions can be specified by additional equations.

*Practice:*

$$\begin{cases} \dfrac{dy}{dt} + y = 3 \\ y(0) = 2 \end{cases}$$

```
>>syms y;                       %define the symbolic variable y
>>dsolve('Dy= -y+3,y(0)=2')    %solve the differential equation
```

*Practice:*

$$\begin{cases} \dfrac{d^2y}{dt^2} - 2\dfrac{dy}{dt} + 3y = 5 \\ y(0) = 1, \qquad y'(0) = 1 \end{cases}$$

```
>>syms y;                                        %define symbolic variable y
>>y=dsolve('D2y-2*Dy+3*y=5,Dy(0)=-2, y(0)=1')    %solve differential equation
>>ezplot(y,[-6,6])                               %plot the solution
```

*Practice:*

$$\begin{cases} \dfrac{d^2y}{dx^2} + y = \cos(3x) \\ y(0) = 1, \qquad y'(0) = 0 \end{cases}$$

```
>>syms x y;                                      %define symbolic variables
>>y=dsolve('D2y=cos(3*x)-y', 'y(0)=1','Dy(0)=0','x'); %solve differential equation
>>y=simplify(y)                                  %simplify the solution

y =

-1/2*cos(x)^3+3/2*cos(x)
```

**Systems of linear differential equations:**

Systems of linear differential equations occur in many problems of science and engineering. In this section we shall illustrate the use of **dsolve** function to deal with systems of linear differential equations.

*Practice:*

Use **dsolve** to find the solution of the given system of differential equations.

$$\begin{cases} \frac{dx_1}{dt} = x_1 + 3x_2 \\ \frac{dx_2}{dt} = 5x_1 + x_2 \end{cases}$$

```
>>syms t x1 x2 Dx1 Dx2;
>>equ='Dx1=x1+3*x2,Dx2=5*x1+x2';
>>[x1,x2]=dsolve(equ,'t')

x1 =

1/2*C1*exp((1+15^(1/2))*t)+1/2*C1*exp(-(-1+15^(1/2))*t)-1/10*C2*15^(1/2)*exp(-(-
1+15^(1/2))*t)+1/10*C2*15^(1/2)*exp((1+15^(1/2))*t)


x2 =

-1/6*C1*15^(1/2)*exp(-(-
1+15^(1/2))*t)+1/6*C1*15^(1/2)*exp((1+15^(1/2))*t)+1/2*C2*exp((1+15^(1/2))*t)+1/2*C2*ex
p(-(-1+15^(1/2))*t)
```

*Practice:*

Use **dsolve** to find the solution of the system of differential equations satisfying the given initial value conditions.

$$\begin{cases} \frac{dx_1}{dt} = x_1 + 3x_2 \\ \frac{dx_2}{dt} = 5x_1 + x_2 \\ x_1(0) = 2, \qquad x_2(0) = 3 \end{cases}$$

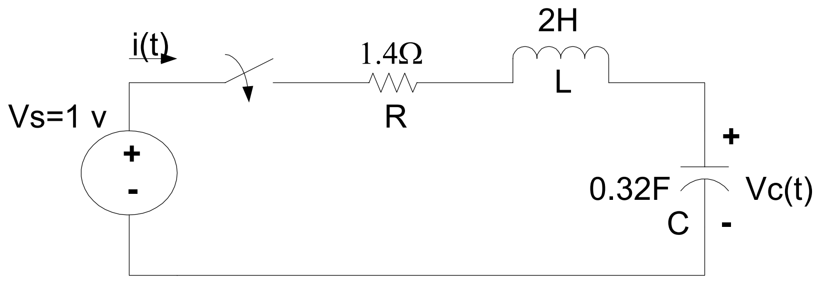
```
>>syms t x1 x2;
>>equ='Dx1=x1+3*x2, Dx2=5*x1+x2, x1(0)=2, x2(0)=3';
>>[x1,x2]=dsolve(equ,'t')



x1 =

exp((1+15^(1/2))*t)+exp(-(-1+15^(1/2))*t)-3/10*15^(1/2)*exp(-(-
1+15^(1/2))*t)+3/10*15^(1/2)*exp((1+15^(1/2))*t)
```

```
x2 =

-1/3*15^(1/2)*exp(-(-
1+15^(1/2))*t)+1/3*15^(1/2)*exp((1+15^(1/2))*t)+3/2*exp((1+15^(1/2))*t)+3/2*exp(-(-
1+15^(1/2))*t)
```

*Practice:*

Consider the RLC circuit depicted below. The initial inductor current is zero, and the initial capacitor voltage is 0.5 volts. A step voltage of 1 volt is applied at time t=0.

1. Determine i(t) and v(t) over the range 0<t<15 sec.
2. Obtain a plot of current versus capacitor voltage



Applying KVL, we obtain the differential equation model of the circuit.

$$L\frac{di}{dt}+Ri+v_c=Vs$$

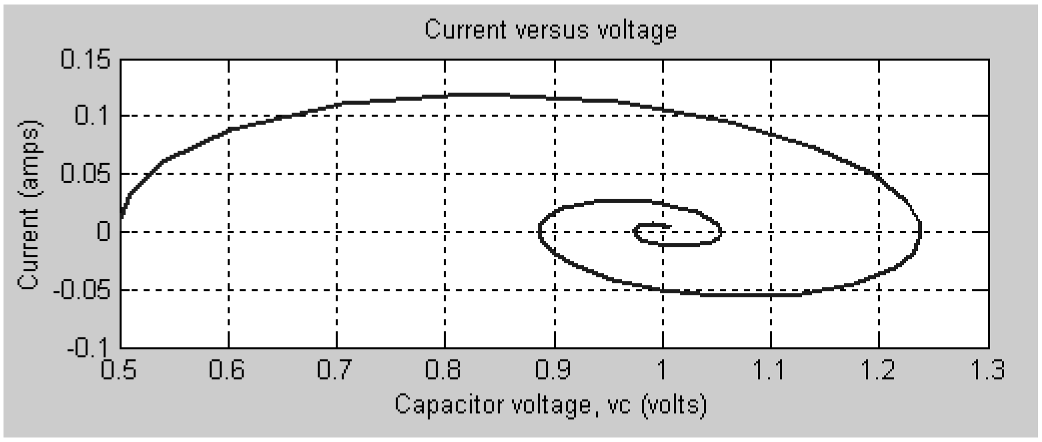
$$i=c\frac{dv_c}{dt}$$

Let

$$x_1=v_c$$
$$x_2=i$$
$$\therefore \dot{x}_1=\frac{x_2}{C}$$
$$\therefore \dot{x}_2=\frac{1}{L}\left(V_s-x_1-Rx_2\right)$$

```
function  xdot=circuit(t,x)
Vs=1;
R=1.4; L=2; C=0.32;
xdot= [x(2)/C ; 1/L*(Vs-x(1)-R*x(2))];

>>x0=[0.5; 0];
```

```
>>tspan=[0 15];
>>[t,x]=ode23('circuit',tspan,x0);
>>plot(x(:,1), x(:,2), 'LineWidth',2);grid
>>ylabel('Current (amps)')
>>xlabel('Capacitor voltage, vc(volts)')
```


Current versus voltage
Current (amps)
Capacitor voltage, vc (volts)