Plotting with MATLAB

One of the outstanding strengths of MATLAB is its capability of graphical visualization of computational results. For this, MATLAB has available very simple but powerful graphical functions. MATLAB is thus able to represent ordinary functions in ***two-dimensional plots*** (xy plots) and functions of two variables in ***three-dimensional plots*** with perspective (xyz plots).

SS

Two-dimensional Plot

The *plot* function is by far the most commonly used graphical function. A plot can be made from

1. a data set or
2. an equation.

A typical xy plot and its various parts is illustrated in Fig. 6. Some commands used for plotting are as follows:

Plot: The MATLAB basic *xy* plotting function is written as *plot*(*x*,*y*). With only one argument, say, *plot*(*y*), the plot function will plot the values in the vector *y* versus their indices 1, 2, 3, . . . , and so on.

xlabel: To label the abscissa, the syntax is *xlabel('text')*

ylabel: To label the ordinate, the syntax is *ylabel('text')*

title: To put a title at the top of the plot, the syntax is *title*('*text*')

Note that the order of *xlabel*, *ylabel*, and *title* commands does not matter but they must be placed *after* the plot command, either on separate lines using ellipses or on the same line separated by commas.

data symbol: When data are plotted, each data point is plotted with a *data symbol*, or *point marker*, such as the small circles shown in Fig. 6. Specifiers for data markers, line types, and colors are illustrated in Table 19

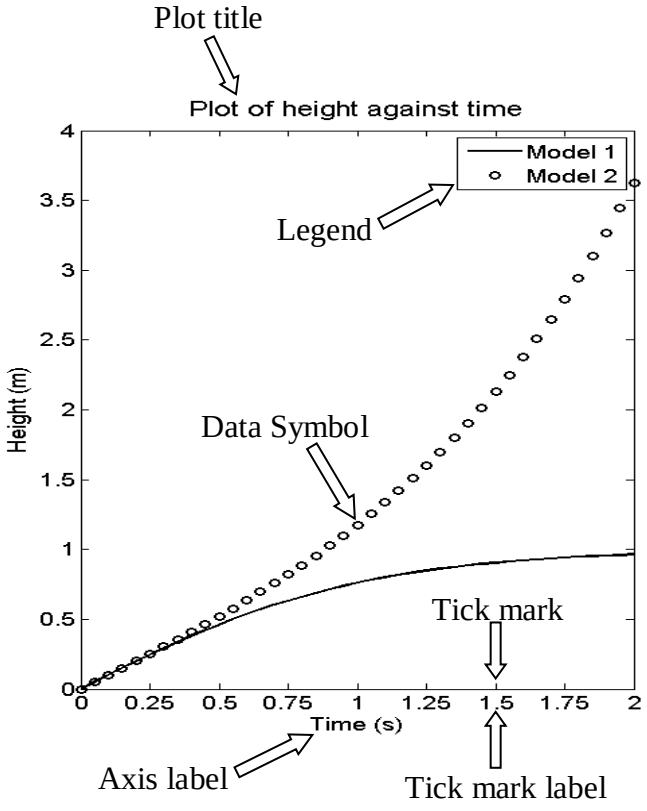


Fig. 6 A typical *xy* plot

Table 19 Specifiers for data markers, line types, and colors

Data markers		Line types		Colors	
Dot (.)	.	Solid line	-	Black	k
Asterisk (*)	*	Dashed line	- -	Blue	b
Cross (x)	x	Dash-dotted line	- .	Cyan	c
Circle (o)	o	Dotted line	:	Green	g
Plus sign (+)	+			Magenta	m
Square (□)	s			Red	r
Diamond (◊)	d			White	w
Five-pointed star (★)	p			Yellow	y

For example, to plot a graph of y against x using the following specifiers

Data marker = □, line type = - -, and color = yellow

the command is : *plot*(x,y, 's -- y')

The *plot*(*x*,*y*) function in MATLAB automatically selects a tick-mark spacing for each axis and places appropriate tick labels. This feature is called *autoscaling*

axis: the axis command can be used to override the MATLAB selections for the axis limits. The syntax is *axis*([*xmin xmax ymin ymax*]). The axis command has the following variants:

Subplot: the *subplot* command is used to plot several smaller "subplots" in the same figure. The syntax is *subplot*(*m*,*n*,*p*). This command divides the Figure window into an array of rectangular panes with *m* rows and *n* columns. The variable p tells MATLAB to place the output of the plot command following the subplot command into the *p*th pane. For example, *subplot*(3,2,5) creates an array of six panes, three panes deep and two panes across, and directs the next plot to appear in the fifth pane (in the bottom left corner).

For example, the following script file is used to construct Fig. 7, which shows the plot of the functions

$y = e^{-1.2x} \sin(10x + 5)$ for $0 \le x \le 5$ and

$y = |x^3 - 100|$ for $-6 \le x \le 6$.

```
x = 0:0.01:5;
y = exp(-1.2*x).*sin(10*x+5);
subplot(1,2,1)
plot(x,y),xlabel('x'),ylabel('y'),axis([0  5  -1  1])
x = -6:0.01:6;
y = abs(x.^3-100);
subplot(1,2,2)
plot(x,y),xlabel('x'),ylabel('y'),axis([-6  6  0  350])
```

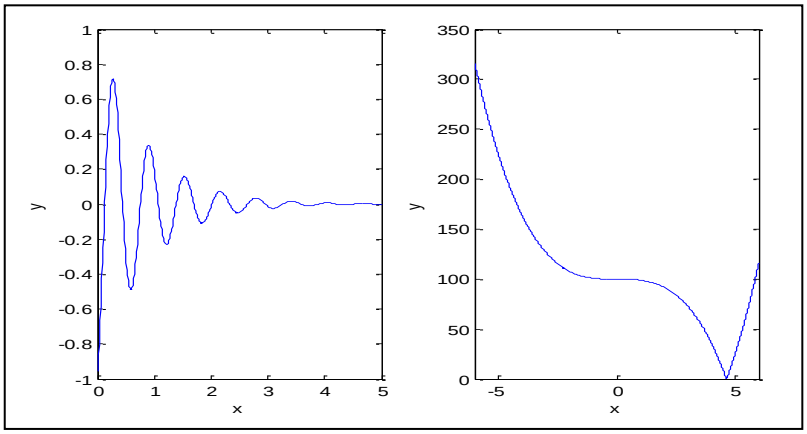


Fig. 7 Illustration of *subplot* command

Overlay Plots: Assuming a matrix A has *m* rows and *n* columns, the following variants of the MATLAB basic plotting functions plot(x,y) and plot(y) can be used to create *overlay plots*:

- *plot*(A) plots the columns of A versus their indices and generates *n* curves.

- *plot*(x,A) plots the matrix A versus the vector x, where x is either a row vector or a column vector and A is a matrix with *m* rows and *n* columns. If the length of x is *m*, then each *column* of A is plotted versus the vector x. There will be as many curves as there are columns of A. If x has length *n*, then each *row* of A is plotted versus the vector x. There will be as many curves as there are rows of A.
- *plot*(A,x) plots the vector x versus the matrix A. If the length of x is *m*, then x is plotted versus the *columns* of A. There will be as many curves as there are columns of A. If the length of x is *n*, then x is plotted versus the *rows* of A. There will be as many curves as there are rows of A.
- *plot*(A,B) plots the columns of the matrix B versus the columns of the matrix A.

For example, if matrices A and B are given as

$$A = \begin{pmatrix} 1 & 7 \\ 2 & 8 \\ 3 & 9 \\ 4 & 10 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \\ 4 & 7 \end{pmatrix}$$

The following script plots the columns in matrix B against the columns in matrix A which is illustrated in Fig. 8

```
A = [1,7;2,8;3,9;4,10];
B = [1,4;2,5;3,6;4,7];
plot(A,B)
xlabel('x'), ylabel('y')
```

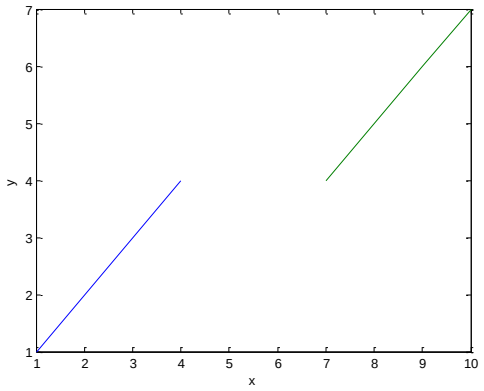


Fig. 8 Plotting columns in matrix B against the columns in matrix A

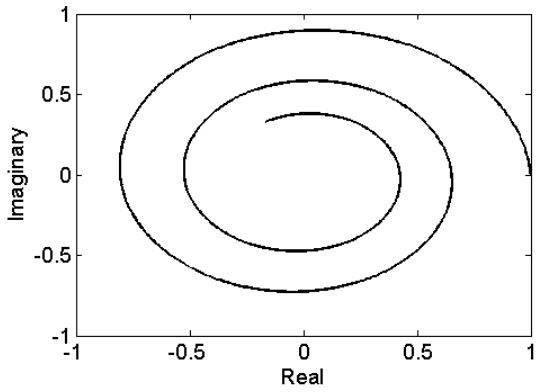


Fig. 9 Plotting a complex number

Plots of complex numbers: if y is complex, plot(y) plots the imaginary parts versus the real parts. Thus plot(y) in this case is equivalent to plot(real(y),imag(y)). This situation is the only time when the plot function handles the imaginary parts; in all other variants of the plot function, it ignores the imaginary parts. For example, the script

```
z = 0.1 + 0.9i;
n = 0:0.01:10;
plot(z.^n), xlabel('Real'), ylabel('Imaginary')
```

generates a spiral plot shown in Fig. 9

The hold command:

The hold command creates a plot that needs two or more plot commands.

Assuming that

$0 \leq x \leq 10$

$y1 = x^2$
$y2 = x^3$
$y3 = 100 + x^2$
$y4 = 100 + x^3$

Suppose it is required to plot y2 versus y1 on the same plot with y4 versus y4, the following script can be used and the result is illustrated in Fig. 8.

```
x = 0:10;
y1 = x.^2;
y2 = x.^3;
y3 = 100 + x.^2;
y4 = 100 + x.^3;
plot(y1,y2)
hold
plot(y3,y4)
gtext('y2 versus y1'), gtext('y4 versus y3')
```

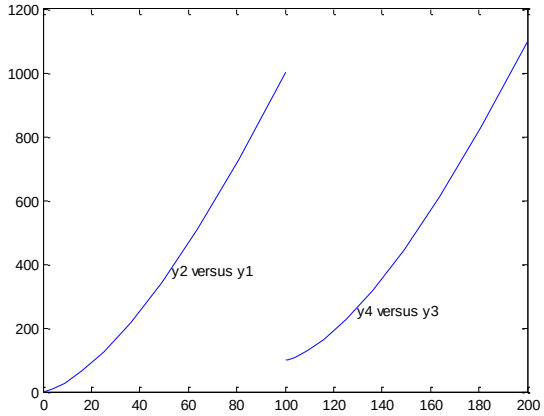


Fig. 8. Demonstration of the "hold" command

Note that the "gtext" command places a string text in a figure window at a point specified by the mouse. For example, the string texts *"y2 versus y1"* and *'y4 versus y3'* are placed in Fig. 8 when the mouse button is clicked within Fig. 8. However, when more than one plot command is used, the "gtext" command should not be placed before any of the plot commands.

Logarithmic Plots

Logarithmic scales abbreviated as log scales are widely used to
(1) represent a data set that covers a wide range of values and
(2) identify certain trends in data.

It is important to remember the following points when using log scales:

1. Negative numbers can not be plotted on a log scale because the logarithm of a negative number is not defined as a real number.
2. **N**umber 0 can not be plotted on a log scale because $\log_{10}0 = \ln 0 = -\infty$.

MATLAB has three commands for generating plots having log scales. The appropriate command depends on which axis must have a log scale. The commands are:

1. **loglog(x,y)** : both x and y scales are logarithmic.
2. **semilogx(x,y):** the *x* scale is logarithmic and the *y* scale is rectilinear.
3. **semilogy(x,y):** the *y* scale is logarithmic and the *x* scale is rectilinear.

The following script demonstrates the use of the logarithmic plotting functions and Fig. 9 illustrates the corresponding graph plotted

```
x1 = 0:0.01:3; y1 = 25*exp(0.5*x1);
y2 = 40*(1.7.^x1);
x2 = logspace(-1,1,500); y3 = 15*x2.^(0.37);
subplot(1,2,1),semilogy(x1,y1,x1,y2, '--'),...
legend ('y = 25e^{0.5x}', 'y = 40(1.7) ^x'),...
xlabel('x'),ylabel('y'),grid,...
subplot(1,2,2),loglog(x2,y3),legend('y = 15x^{0.37}'),...
xlabel('x'),ylabel('y'),grid
```

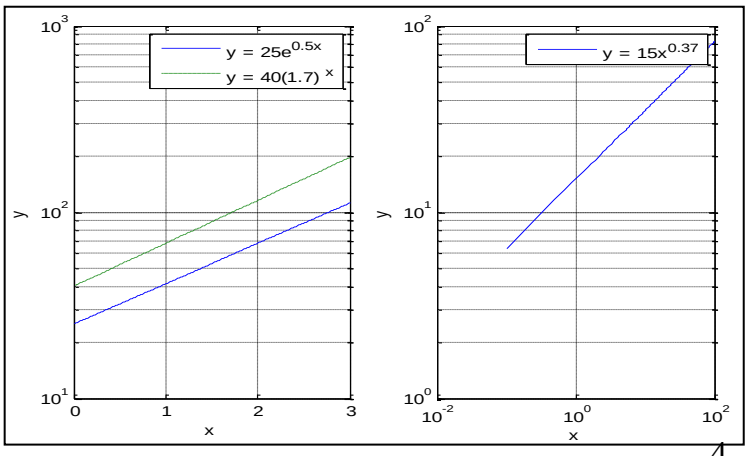


Fig. 9 Demonstration of the logarithmic plotting functions

Table 20 Some other 2D plotting functions

Command	Description
bar(x,y)	Creates a bar chart of y versus x.
plotyy(x1,y1,x2,y2)	Produces a plot with two *y* axes, y1 on the left and y2 on the right.
polar(theta,r,'type')	Produces a polar plot from the polar coordinates theta and r, using the line type, data marker, and colours specified in the string type.
stairs(x,y)	Produces a stairs plot of y versus x.
stem(x,y)	Produces a stem plot of y versus x.

Table 20 contains some other 2D plotting functions. The following script demonstrates the use of some of these special plotting commands while Fig. 10 illustrates the corresponding graphs generated.

```
x=linspace(0,2*pi,14)
y=sin(x);
subplot(1,3,1)
bar(x,y)
title('Bar plot'), xlabel('x'),
ylabel('y')
subplot(1,3,2)
stairs(x,y)
title('Stairs plot'), xlabel('x'),
ylabel('y')
subplot(1,3,3)
stem(x,y)
title('Stem plot'), xlabel('x'),
ylabel('y')
```

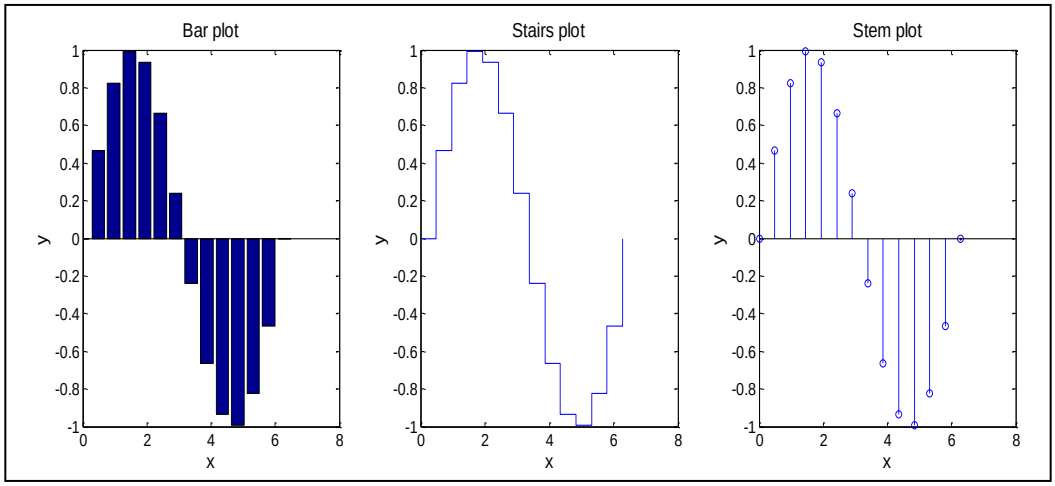


Fig. 10 Demonstration of the bar, stairs, and stem plotting commands

Consider the expression

$$r = \frac{2}{1 - 0.5\cos\theta}$$

where r denotes distance and θ denotes angle. The following script illustrates how the polar plot of the expression can be realised while Fig. 11 illustrates the corresponding graph generated.

```
theta = 0:pi/90:2*pi;
r = 2./(1-0.5*cos(theta));
polar(theta,r),title('Polar plot')
```

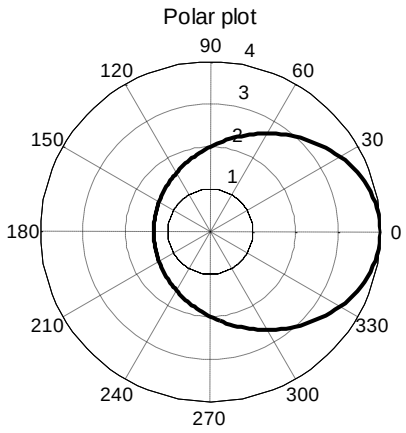


Fig. 11 Demonstration of polar plot

Example 7

Assuming that the Cartesian coordinates (x,y) can be converted to an equivalent polar coordinates (r,θ) by applying the following expressions:

$$r = \sqrt{x^2 + y^2} \text{ and}$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

Write a MATLAB script that converts the Cartesian coordinates in Table 22 into polar coordinates and execute the polar plot.

Table 22 Cartesian coordinates

x	y
1	2
4	3
1	1
4	0
9	1

Solution

```
points = [1 2; 4 3; 1 1; 4 0; 9 1];
r = sqrt(points(:,1).^2 + points(:,2).^2);
thetha = atan(points(:,2)./points(:,1));
polar(theta,r)
```

The result of the plot is illustrated in Fig. 12

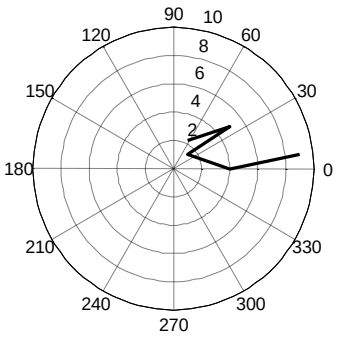


Fig. 12 Polar plot for Example 7

Three-dimensional Plot

MATLAB provides many functions for creating three-dimensional plots which includes
- line plots
- surface plots, and
- contour plots.

Line plot

Lines in three-dimensional space can be plotted with the *plot3* function. Its syntax is:

plot3(x,y,z)

For example, consider the following equations:

$x = e^{-0.005t}\sin t$

$y = e^{-0.005t}\cos t$

$z = t$

$0 \leq t \leq 10\pi$

The following script can be used to realize the 3D-line plot of the equations and the corresponding graph generated is illustrated in Fig. 13

```
t = 0:pi/50:10*pi;
plot3(exp(-0.05*t).*sin(t),exp(-0.05*t).*cos(t),t),...
```

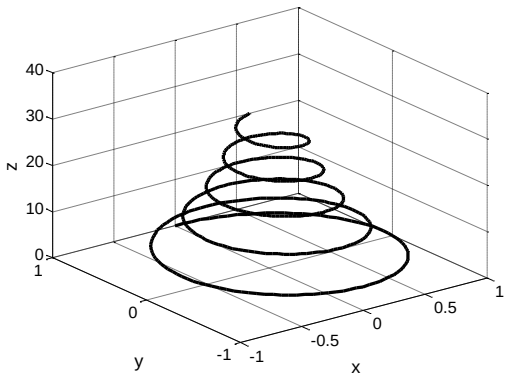


Fig. 13 Demonstration of 3D-line plot

xlabel('x'),ylabel('y'),zlabel('z'),grid

Surface Mesh Plot

The function $z = f(x, y)$ represents a surface when plotted on *xyz* axes, and the mesh function provides the means to generate a surface plot.

Before this function can be used, a grid of points must be generated in the *xy* plane and then the function $f(x, y)$ can be evaluated at these points. The meshgrid function generates the grid and its syntax is

[X,Y] = meshgrid(x,y)

Or

[X,Y] = meshgrid(min:spacing:max)

After the grid is computed, the surface plot can be generated with the mesh function. Its syntax is:

mesh(x,y,z)

Consider the expression

$$z = xe^{-\left[\left(x-y^2\right)^2+y^2\right]},$$

for $-2 \le x \le 2$ and $-2 \le y \le 2$

The following script can be used to realize the surface mesh plot of the expression and the corresponding graph generated is illustrated in Fig. 14

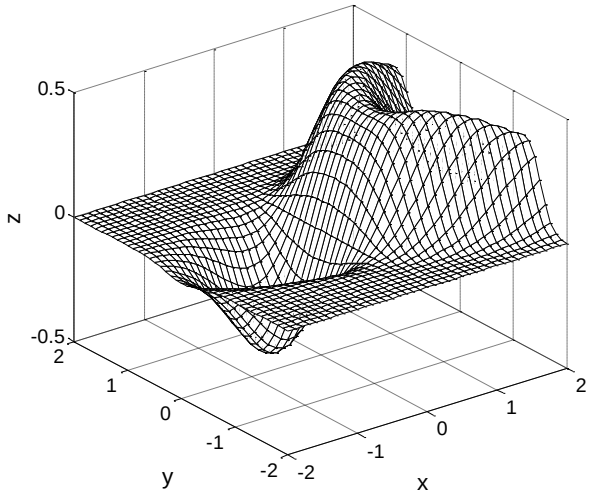


Fig. 14 Demonstration of surface mesh plot i.e., *mesh* command

```
[X,Y] = meshgrid(-2:0.1:2);
Z = X.*exp(-((X-Y.^2).^2+Y.^2));
mesh(X,Y,Z),xlabel('x'),ylabel('y'),zlabel('z')
```

The surf and surfc functions are similar to mesh and meshc except that the former create a shaded surface plot. Consider the expression

$$z = \frac{1}{1+\left(x+0.5y\right)^2},$$

for $-5 \le x \le 5$ and $0 \le y \le 10$

The following script can be used to realize the surface mesh plot of the expression and the corresponding graph generated is illustrated in Fig. 15

```
x=linspace(-5,5,41)';
y=linspace(0,10,41)';
[X,Y]=meshgrid(x,y);
z=1./(1+(X+0.5*Y).^2);
surf(x,y,z)
xlabel('x'), ylabel('y') ,zlabel('z')
```

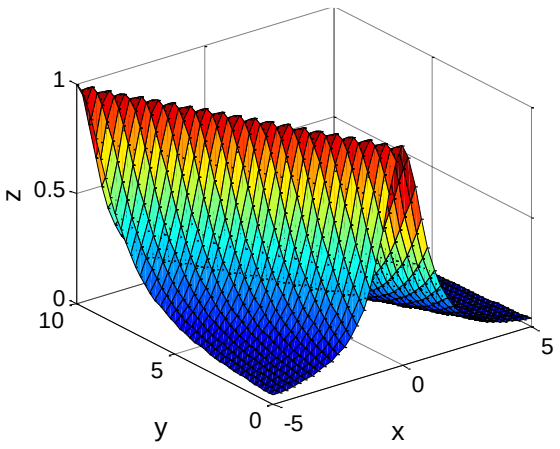


Fig. 15 Demonstration of surface mesh plot i.e., *surf* command

Contour plot

Contour plots provides a means of illustrating regions of constant elevation in a graph. The syntax is

contour(x,y,z)

This function is used the same way as the mesh function; that is, first use the meshgrid function to generate the grid and then generate the function values.

Recall that the surface mesh plot for the expression

$$z = xe^{-\left[\left(x-y^2\right)^2+y^2\right]},$$

for $-2 \le x \le 2$ and $-2 \le y \le 2$

was illustrated in Fig. 14. The corresponding contour plot illustrated in Fig. 16 can be realized using the following script:

```
[x,y] = meshgrid(-2:0.1:2);
z = x.*exp(-((x-y.^2).^2+y.^2));
contour(x,y,z),xlabel('x'),ylabel('y')
```

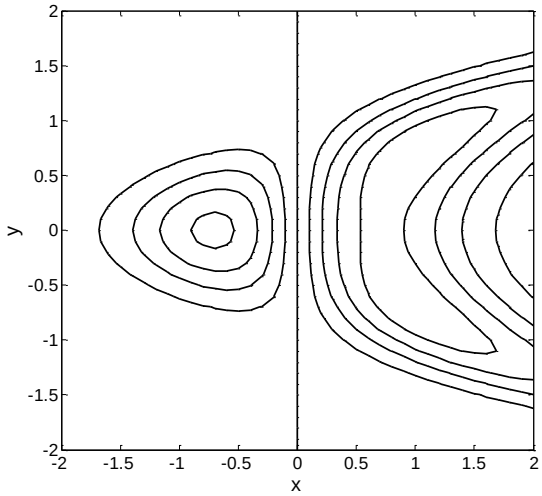


Fig. 16 Demonstration of contour plot command

Table 23 Some other 3D plotting functions

Command	Description
meshc(x,y,z)	Same as mesh but draws a contour plot under the surface.
meshz(x,y,z)	Same as mesh but draws a series of vertical reference lines under the surface.
surfc(x,y,z)	Same as surf but draws a contour plot under the surface.
[x,y] = meshgrid(x)	Same as [x,y] = meshgrid(x,x).

Some other 3D plotting functions are illustrated in Table 23

Polynomials

A polynomial has the form:

$$f(x) = a_1x^n + a_2x^{n-1} + a_3x^{n-2} + \cdots + a_{n-1}x^2 + a_nx + a_{n+1}$$

This can be defined in MATLAB as row vector:

$[a_1, a_2, a_3, \ldots, a_{n-1}, a_n, a_{n+1}]$

For example, the vector [4,-8,7,-5] represents the polynomial $4x^3 - 8x^2 + 7x - 5$.

Polynomial roots

Polynomial roots can be found with the function:

roots(a)

where a is the array containing the polynomial coefficients.

For example, to obtain the roots of the polynomial

$x^3 + 12x^2 + 45x + 50 = 0$,

it can be done as follows:

```
>> y = roots([1,12,45,50])
y =
        -5.0000 + 0.0000i
        -5.0000 - 0.0000i
        -2.0000
```

Polynomial coefficients

To determine the coefficients of the polynomial whose roots are specified by the array r, the syntax is:

poly(r)

The result is a *row* array that contains the polynomial's coefficients. For example, to find the polynomial whose roots are 1 and $3 \pm 5i$, it can be done as follows:

```
>> p = poly([1,3+5i, 3-5i])
p =
     1    -7    40   -34
```

Thus the polynomial is $x^3 - 7x^2 + 40x - 34 = 0$,

Polynomial Addition and Subtraction

To add two polynomials, add the arrays that describe their coefficients.

If the polynomials are of different degrees, add zeros to the coefficient array of the lower-degree polynomial. For example, consider

$f(x) = 9x^3 - 5x^2 + 3x + 7$

whose coefficient array is f = [9,-5,3,7] and

$g(x) = 6x^2 - x + 2$

whose coefficient array is g = [6,-1,2].

The degree of $g(x)$ is 1 less that of $f(x)$. Therefore, to add $f(x)$ and $g(x)$, we append one zero to g(x). That is,

g = [0,6,-1,2]

The two polynomials can then be added as follows:

```
>> f = [9,-5,3,7];
>>g = [0,6,-1,2] ;
>>f+g
ans =
    9    1    2    9
```

Polynomial Multiplication and Division

(i) multiply/divide polynomial by Scalar: Multiplying a polynomial by a scalar is straight forward. For example, the following illustrates how the polynomial $f(x) = 9x^3 - 5x^2 + 3x + 7$ can be multiplied by scalar number 5:

```
>> f = [9,-5,3,7];
>> 5 * f
ans =
    45   -25    15    35
```

Dividing by a polynomial by a scalar is done in a similar way as multiplication except that in this case, the multiplication operator is replaced by a division operator.

(ii) multiply/divide polynomial by another polynomial: To multiply a polynomial by another polynomial, the *conv* function (*conv* stands for "convolve") can be used.

To divide a polynomial by another polynomial, the *deconv* function (*deconv* stands for "deconvolve") can be used.

Consider polynomials

$f(x) = 9x^3 - 5x^2 + 3x + 7$ and

$g(x) = 6x^2 - x + 2$,

their product is

$f(x)g(x) = (9x^3 - 5x^2 + 3x + 7)(6x^2 - x + 2)$

also, their division is

$f(x)/g(x) = \dfrac{9x^3 - 5x^2 + 3x + 7}{6x^2 - x + 2} = 1.5x - 0.5833$ with a remainder of $-0.5833x + 8.1667$.

The above multiplication and division can be executed in MATLAB as follows:

```
>> f = [9,-5,3,7];
>> g = [6,-1,2];
>> product = conv(f,g)
product =
      54   -39    41    29    -1    14
>> [quotient, remainder] = deconv(f,g)
quotient =
      1.5000   -0.5833
remainder =
       0         0   -0.5833    8.1667
```

Polynomial evaluation

The *polyval*(*a*,*x*) function evaluates a polynomial at specified values of its independent variable *x*, which can be a matrix or a vector.

The polynomial's coefficient array is denoted by *a*. The result is the same size as *x*. For example, the following illustrates how to evaluate the polynomial $f(x) = 9x^3 - 5x^2 + 3x + 7$ at the points $x = 0, 2, 4, \ldots, 10$,

```
>> f = polyval([9,-5,3,7],[0:2:10]);
>> evaluations = f '
evaluations =
                7
               65
              515
             1789
             4319
             8537
```

The resulting vector f contains six values that correspond to $f(0)$, $f(2)$, $f(4)$, . . . , $f(10)$.

Nodal Analysis

Nodal analysis provides a general procedure for analysing circuits using node voltages as the circuit variables.

Steps to Determine Node Voltages:
1. Select a node as the reference node. Assign voltages to the remaining nodes. The voltages are referenced with respect to the reference node.
2. Apply KCL to each of the non-reference nodes.
3. Solve the resulting simultaneous equations to obtain the unknown node voltages.

Example 8

Write nodal equations for the circuit shown in Fig. 17(a), and solve for the unknowns in these equations using
(i) Cramer's rule
(ii) MATLAB script

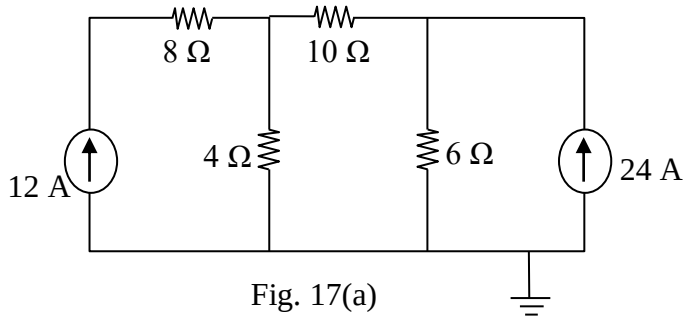


Fig. 17(a)

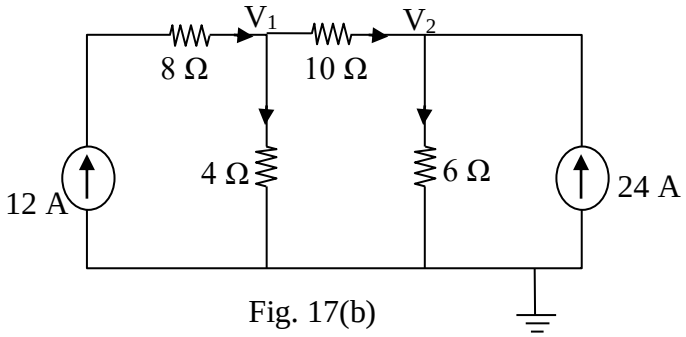


Fig. 17(b)

Solution

(i)

Redraw the circuit as shown in Fig. 17(b) with the arbitrary direction of flow of currents indicated. Apply KCL

KCL @ Node #1:

$$\frac{V_1}{4}+\frac{V_1-V_2}{10}=12$$

$$7V_1-2V_2=240 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

KCL @ Node #2:

$$\frac{V_1-V_2}{10}+24=\frac{V_2}{6}$$

$$3V_1-8V_2=720 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

The unknown variables V_1 and V_2 can be determined using either
(a) Elimination method; or
(b) Cramer's rule

To use Cramer's rule, Eqs. (1) and (2) have to be written in matrix form as shown in Eq. (3):

$$\underbrace{\begin{bmatrix} 7 & -2 \\ 3 & -8 \end{bmatrix}}_{G}\underbrace{\begin{bmatrix} V_1 \\ V_2 \end{bmatrix}}_{V} = \underbrace{\begin{bmatrix} 240 \\ 720 \end{bmatrix}}_{I} \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The determinant of the matrix is

$$\Delta = \begin{vmatrix} 7 & -2 \\ 3 & -8 \end{vmatrix} = -56 - (-6) = -50$$

We now obtain V_1 and V_2 as

$$V_1 = \frac{\Delta_1}{\Delta} = \frac{\begin{vmatrix} 240 & -2 \\ 720 & -8 \end{vmatrix}}{-50} = \frac{-1920 - (-1440)}{-50} = \frac{-480}{-50} = 9.6$$

$$V_2 = \frac{\Delta_2}{\Delta} = \frac{\begin{vmatrix} 7 & 240 \\ 3 & 720 \end{vmatrix}}{-50} = \frac{5040 - (720)}{-50} = \frac{4320}{-50} = -86.4$$

Hence $V_1 = 9.6$ V and $V_2 = -86.4$

(ii)
MATLAB can be used to solve the matrix. Equation (3) can be written as

GV = I → **V = G^{-1} I**

where **G** is the 2 by 2 matrix, **I** is the column vector, and **V** is a column vector comprised of V_1 and V_2 that we want to determine.
MATLAB can be used to determine **V** as follows:

```
>> G=[7,-2;3,-8];
>> I=[240;720];
>> V = inv(G)*I
V =
          9.6000
         -86.4000
```

Example 9

Write nodal equations for the circuit shown in Fig. 18(a), and solve for the unknowns in these equations using
(i) Cramer's rule
(ii) MATLAB script

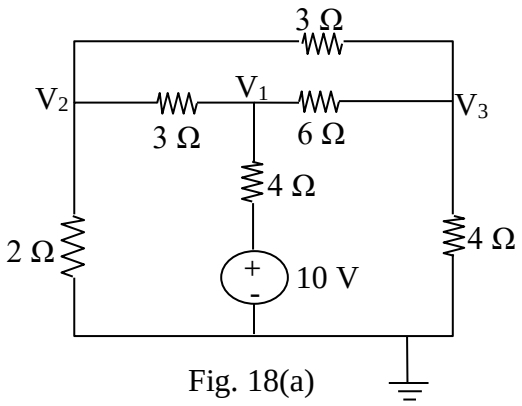


Fig. 18(a)

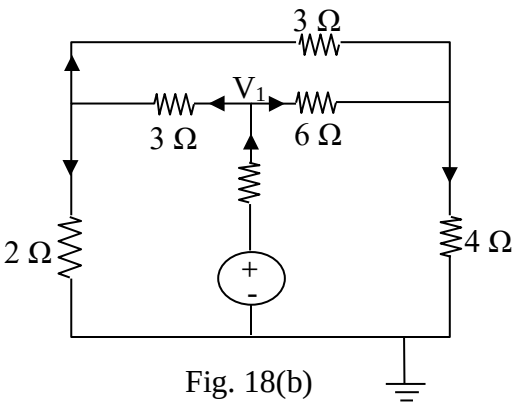


Fig. 18(b)

Solution

(i) Redraw the circuit as shown in Fig. 18(b) with the arbitrary direction of flow of currents indicated. Apply KCL

KCL @ Node #1:

$$\frac{V_1 - V_2}{3} + \frac{V_1 - V_3}{6} = \frac{10 - V_1}{4}$$

$$9V_1 - 4V_2 - 2V_3 = 30 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

KCL @ Node #2:

$$\frac{V_2 - V_3}{3} + \frac{V_2}{2} = \frac{V_1 - V_2}{3}$$

$$-2V_1 - 7V_2 - 2V_3 = 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

KCL @ Node #3:

$$\frac{V_1 - V_3}{6} + \frac{V_2 - V_3}{3} = \frac{V_3}{4}$$

$$2V_1 + 4V_2 - 9V_3 = 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

To use Cramer's rule, Eqs. (1), (2), and (3) have to be written in matrix form as shown in Eq. (4):

$$\underbrace{\begin{bmatrix} 9 & -4 & -2 \\ -2 & -7 & -2 \\ 2 & 4 & -9 \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}}_{V} = \underbrace{\begin{bmatrix} 30 \\ 0 \\ 0 \end{bmatrix}}_{I} \ldots\ldots\ldots\ldots\ldots\ldots(4)$$

The determinant of the matrix is

$$\Delta = \begin{vmatrix} 9 & -4 & -2 \\ -2 & -7 & -2 \\ 2 & 4 & -9 \end{vmatrix} = \begin{array}{ccc} 9 & -4 & -2 \\ -2 & -7 & -2 \\ 2 & 4 & -9 \\ 9 & -4 & -2 \\ -2 & -7 & -2 \end{array} \quad \begin{array}{c} + \\ + \\ + \\ - \\ - \\ - \end{array}$$

$$= 567 + 16 + 16 - 28 + 72 + 72 = 715$$

$$\Delta_1 = \begin{vmatrix} 30 & -4 & -2 \\ 0 & -7 & -2 \\ 0 & 4 & -9 \end{vmatrix} = \begin{array}{ccc} 30 & -4 & -2 \\ 0 & -7 & -2 \\ 0 & 4 & -9 \\ 30 & -4 & -2 \\ 0 & -7 & -2 \end{array} \quad \begin{array}{c} + \\ + \\ + \\ - \\ - \\ - \end{array}$$

$$= 1890 - 0 - 0 - 0 + 240 - 0 = 2130$$

$$\Delta_2 = \begin{vmatrix} 9 & 30 & -2 \\ -2 & 0 & -2 \\ 2 & 0 & -9 \end{vmatrix} = \begin{array}{ccc} 9 & 30 & -2 \\ -2 & 0 & -2 \\ 2 & 0 & -9 \\ 9 & 30 & -2 \\ -2 & 0 & -2 \end{array} \quad \begin{array}{c} + \\ + \\ + \\ - \\ - \\ - \end{array}$$

$$\Delta_3 = \begin{vmatrix} 9 & -4 & 30 \\ -2 & -7 & 0 \\ 2 & 4 & 0 \end{vmatrix} = \begin{array}{ccc} 9 & -4 & 30 \\ -2 & -7 & 0 \\ 2 & 4 & 0 \\ 9 & -4 & 30 \\ -2 & -7 & 0 \end{array} \quad \begin{array}{c} + \\ + \\ + \\ - \\ - \\ - \end{array}$$

$= 0 + 0 - 120 - 0 - 0 - 540 = -660$ $= 0 - 240 - 0 + 420 - 0 - 0 = 180$

$$V_1 = \frac{\Delta_1}{\Delta} = \frac{2130}{715} = 2.98 \qquad V_2 = \frac{\Delta_2}{\Delta} = \frac{-660}{715} = -0.92 \qquad V_3 = \frac{\Delta_3}{\Delta} = \frac{180}{715} = 0.25$$

(ii)
MATLAB can be used to solve the matrix. Equation (4) can be written as

$$\mathbf{GV = I} \quad \rightarrow \quad \mathbf{V = G^{-1} I} \quad \textbf{or} \quad V=G\backslash I$$

where **G** is the 3 by 3 matrix, **I** is the column vector, and **V** is a column vector comprised of V_1, V_2 and V_3 that we want to determine.
MATLAB can be used to determine **V** as follows:

```
>> G=[9,-4, -2;-2,-7,-2;2,4,-9];
>> I=[30  0   0]';
>> V=G\I
    V =
          2.9790
         -0.9231
          0.2517
```

or

```
>> G=[9,-4, -2;-2,-7,-2;2,4,-9];
>> I=[30; 0; 0];
>> V=inv(G)*I
    V =
          2.9790
         -0.9231
          0.2517
```

Mesh analysis
Mesh analysis provides another general procedure for analyzing circuits, using mesh currents as the circuit variables.

Steps to Determine Mesh Currents:
1. Assign mesh currents to the *n* meshes.
2. Apply KVL to each of the *n* meshes.
3. Solve the resulting *n* simultaneous equations to get the mesh currents.

Example 10

Write mesh equations for the circuit shown in Fig. 19(a), and solve for the unknowns in these equations using
(i) Cramer's rule
(ii) MATLAB script

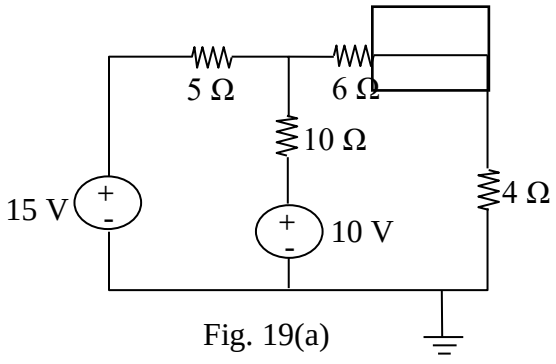


Fig. 19(a)

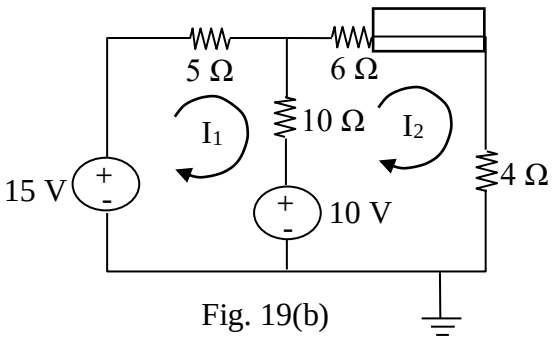


Fig. 19(b)

Solution

(i) Redraw the circuit as shown in Fig. 19(b) with the arbitrary direction of flow of currents indicated and apply KVL around each mesh:

KVL @ Mesh #1:

The mesh current is written as

$-15 + 5I_1 + 10(I_1 - I_2) + 10 = 0$

$3I_1 - 2I_2 = 1$……(1)

KVL @ Mesh #2:

The mesh current is written as

$-10 + 10(I_2 - I_1) + 6I_2 + 4I_2 = 0$

$-I_1 + 2I_2 = 1$……(2)

To use Cramer's rule, Eqs. (1) and (2) have to be written in matrix form as shown in Eq. (3):

$$\underbrace{\begin{bmatrix} 3 & -2 \\ -1 & 2 \end{bmatrix}}_{R}\underbrace{\begin{bmatrix} I_1 \\ I_2 \end{bmatrix}}_{I} = \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{V} \quad \text{……(3)}$$

The determinant of the matrix is

$$\Delta = \begin{vmatrix} 3 & -2 \\ -1 & 2 \end{vmatrix} = 6 - (-2) = 4$$

Then, obtain I_1 and I_2 as

$$I_1 = \frac{\Delta_1}{\Delta} = \frac{\begin{vmatrix} 1 & -2 \\ 1 & 2 \end{vmatrix}}{4} = \frac{2-(-2)}{4} = \frac{4}{4} = 1 \qquad I_2 = \frac{\Delta_2}{\Delta} = \frac{\begin{vmatrix} 3 & 1 \\ -1 & 1 \end{vmatrix}}{4} = \frac{3-(-1)}{4} = \frac{4}{4} = 1$$

Hence $I_1 = 1$ A and $I_2 = 1$ A

(ii)

MATLAB can be used to solve the matrix. Equation (3) can be written as

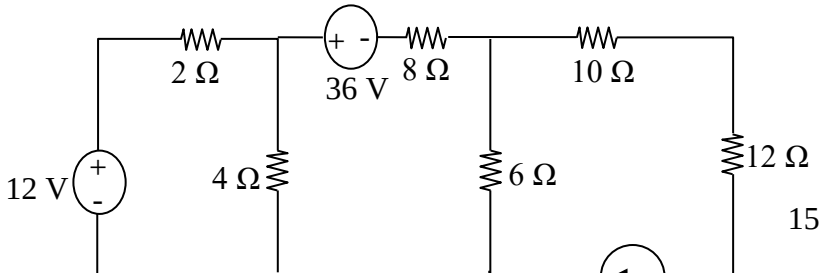
RI = V → **I = R^{-1} V**

where **R** is the 2 by 2 matrix, **V** is the column vector, and **I** is a column vector comprised of I_1 and J_2 that we want to determine.

MATLAB can be used to determine **I** as follows:

```
>> R=[3,-2;-1,2];
>> V=[1,1]';
>> I=inv(R)*V
   I =
         1.0000
         1.0000
```

Example 11



Write mesh equations for the circuit shown in Fig. 20(a), and solve for the unknowns in these equations using
(i) Cramer's rule
(ii) MATLAB script

Solution

(i) Redraw the circuit as shown in Fig. 20(b) with the arbitrary direction of flow of currents indicated and apply KVL around each mesh:

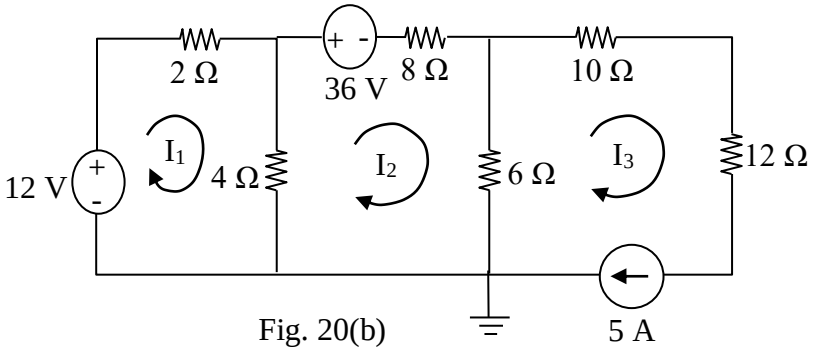


Fig. 20(b)

KVL @ Mesh #1:
The mesh current is written as

$$-12 + 2I_1 + 4(I_1 - I_2) = 0$$
$$3I_1 - 2I_2 = 6 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

KVL @ Mesh #2:
The mesh current is written as

$$36 + 8I_2 + 6(I_2 - I_3) + 4(I_2 - I_1) = 0$$
$$-2I_1 + 9I_2 - 3I_3 = -18 \ldots\ldots\ldots\ldots(2)$$

KVL @ Mesh #3:
The mesh current is written as

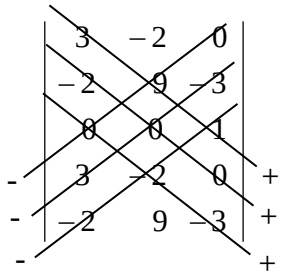
$$I_3 = 5 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

To use Cramer's rule, Eqs. (1), (2), and (3) have to be written in matrix form as shown in Eq. (3):

$$\begin{bmatrix} 3 & -2 & 0 \\ -2 & 9 & -3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -18 \\ 5 \end{bmatrix} \ldots\ldots\ldots(4)$$

The determinant of the matrix is

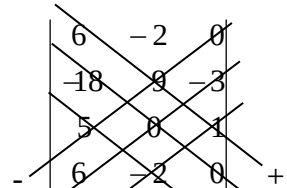
$$\Delta = \begin{vmatrix} 3 & -2 & 0 \\ -2 & 9 & -3 \\ 0 & 0 & 1 \end{vmatrix} =$$



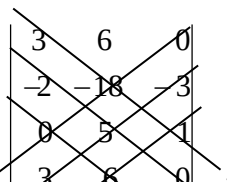
$$= 27 + 0 + 0 - 0 - 0 - 4 = 23$$

Then, obtain I_1, I_2, and I_3 as

$$\Delta_1 = \begin{vmatrix} 6 & -2 & 0 \\ -18 & 9 & -3 \\ 5 & 0 & 1 \end{vmatrix} =$$

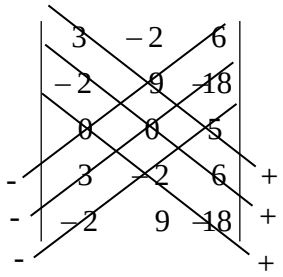


$$\Delta_2 = \begin{vmatrix} 3 & 6 & 0 \\ -2 & -18 & -3 \\ 0 & 5 & 1 \end{vmatrix} =$$



= 54 + 0 + 30 – 0 – 0 – 36 = 48 = -54 + 0 + 0 – 0 – (-45) – (-12) = 3

$$\Delta_3 = \begin{vmatrix} 3 & -2 & 6 \\ -2 & 9 & -18 \\ 0 & 0 & 5 \end{vmatrix} =$$



= 135 + 0 + 0 - 0 – 0 – 20 = 115

Hence,

$$I_1 = \frac{\Delta_1}{\Delta} = \frac{48}{23} = 2.08\,\text{A} \qquad I_2 = \frac{\Delta_2}{\Delta} = \frac{3}{23} = 0.13\,\text{A} \qquad I_3 = \frac{\Delta_3}{\Delta} = \frac{115}{23} = 5.00\,\text{A}$$

(ii)
MATLAB can be used to solve the matrix. Equation (4) can be written as

$$\mathbf{RI = V} \quad \rightarrow \quad \mathbf{I = R^{-1} V}$$

where **R** is the 3 by 3 matrix, **V** is the column vector, and **I** is a column vector comprised of I_1, I_2, and J_3 that we want to determine.
MATLAB can be used to determine **I** as follows:

```
>> R = [3,-2,0;-2,9,-3;0,0,1];
>>V = [6;-18;5];
>>I = inv(R)*V
   I =
        2.0870
        0.1304
        5.0000
```

Sinusoidal Steady State Analysis

Nodal analysis and mesh analysis can be applied in the analysis of ac circuits.

Steps to Analyze AC Circuits:
1. Transform the circuit to the phasor or frequency domain.
2. Solve the problem using circuit techniques (e.g., nodal analysis, mesh analysis, superposition, etc.)

Step 1 is not necessary if the problem is specified in the frequency domain. In step 2, the analysis is performed in the same manner as dc circuit analysis except that complex numbers are involved.

Nodal analysis (ac)

Since KCL is valid for phasors as its usage will be demonstrated in Example 12

Example 12

Write nodal equations for the circuit shown in Fig. 21(a), and solve for the unknowns in these equations using
(i) Cramer's rule
(ii) MATLAB codes

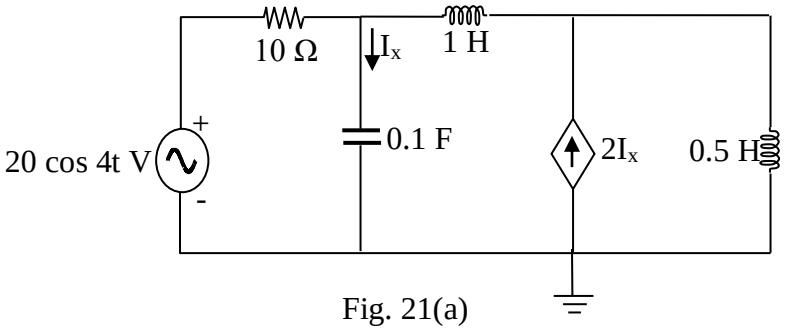


Fig. 21(a)

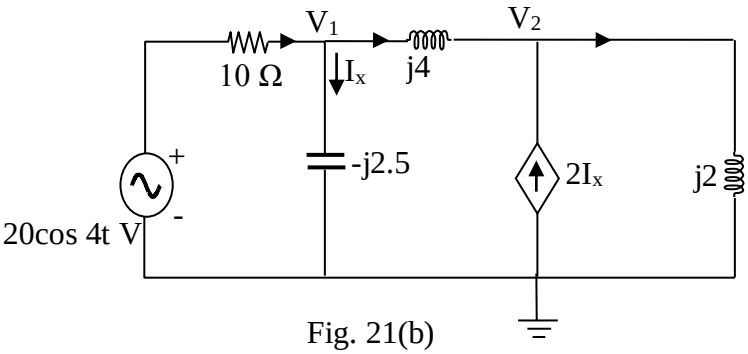


Fig. 21(b)

Solution

We first convert the circuit to the frequency domain:

$$20\cos 4t \;\Rightarrow\; 20\angle 0^{\circ}, \quad \omega = 4\,\text{rad/s}$$
$$1\text{H} \;\Rightarrow\; j\omega L = j4$$
$$0.5\text{H} \;\Rightarrow\; j\omega L = j2$$
$$0.1\text{F} \;\Rightarrow\; \frac{1}{j\omega C} = -j2.5$$

Thus, the frequency domain equivalent circuit is as shown in Fig. 21(b).

KCL @ Node #1:

$$\frac{V_1 - V_2}{j4} + \frac{V_1 - 0}{-j2.5} = \frac{20\angle 0^{\circ} - V_1}{10}$$
$$-j(2.5V_1 - 2.5V_2) + -(-j4V_1) = 20(\cos 0^{\circ} + j\sin 0^{\circ}) - V_1$$
$$(1 + j1.5)V_1 + j2.5V_2 = 20 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

KCL @ Node #2:

$$\frac{V_1 - V_2}{j4} + 2I_x = \frac{V_2 - 0}{j2}$$

But $I_x = \dfrac{V_1 - 0}{-j2.5}$

Substitute for I_x

$$\frac{V_1 - V_2}{j4} + 2\left(\frac{V_1 - 0}{-j2.5}\right) = \frac{V_2 - 0}{j2}$$
$$-j(2.5V_1 - 2.5V_2) + j8V_1 = -j5V_2$$
$$j5.5V_1 + j7.5V_2 = 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

To use Cramer's rule, Eqns. (1) and (2) can be written in matrix form as shown in Eq. (3):

$$\underbrace{\begin{bmatrix}(1+\text{j}1.5) & \text{j}2.5\\ \text{j}5.5 & \text{j}7.5\end{bmatrix}}_{\text{G}}\underbrace{\begin{bmatrix}\text{V}_1\\ \text{V}_2\end{bmatrix}}_{\text{V}}=\underbrace{\begin{bmatrix}20\\ 0\end{bmatrix}}_{\text{I}}\quad\ldots\ldots\ldots\ldots(3)$$

The determinants are obtained as:

$$\Delta=\begin{vmatrix}(1+\text{j}1.5) & \text{j}2.5\\ \text{j}5.5 & \text{j}7.5\end{vmatrix}=j7.5-11.25+13.75=2.5+j7.5$$

$$\Delta_1=\begin{vmatrix}20 & \text{j}2.5\\ 0 & \text{j}7.5\end{vmatrix}=j150-0=j150 \qquad \Delta_2=\begin{vmatrix}(1+\text{j}1.5) & 20\\ \text{j}5.5 & 0\end{vmatrix}=0-j110=-j110$$

Then, obtain V_1 and V_2 as

$$\text{V}_1=\frac{\Delta_1}{\Delta}=\frac{\text{j}150}{2.5+\text{j}7.5}=\frac{150\angle 90^{\text{o}}}{7.91\angle 71.57^{\text{o}}}=18.96\angle 18.43^{\text{o}}\ \text{V} \quad \text{or} \quad 17.98 + \text{j}5.99\ \text{V}$$

$$\text{V}_2=\frac{\Delta_2}{\Delta}=\frac{-\text{j}110}{2.5+\text{j}7.5}=\frac{110\angle 270^{\text{o}}}{7.91\angle 71.57^{\text{o}}}=13.91\angle 198.43^{\text{o}}\ \text{V} \quad \text{or} \quad -13.2 - \text{j}4.4\ \text{V}$$

(ii)
MATLAB can be used to solve the matrix. Equation (3) can be written as

GV = I → **V = G^{-1} I**

where **G** is the 2 by 2 matrix, **I** is the column vector, and **V** is a column vector comprised of V_1 and V_2 that we want to determine.
MATLAB can be used to determine **V** as follows:

```
>> G=[1+1.5*j  2.5*j; 5.5*j 7.5*j];
>> I = [20;0];
>> V = inv(G)*I
   V =
         18.0000 + 6.0000i
        -13.2000 - 4.4000i
```

Example 13

(i) Write nodal equations for the circuit shown in Fig. 22(a)
(ii) Write a MATLAB script for determining the amount of current flowing through the 100 Ω resistor

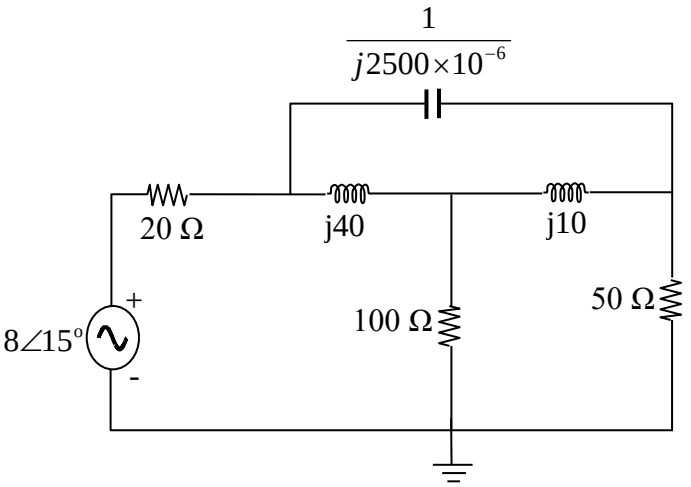


Fig. 22(a)

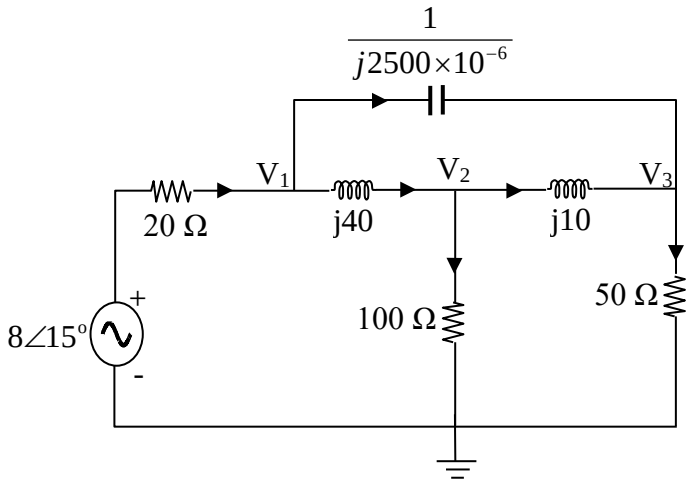


Fig. 22(b)

Solution

(i)

KCL @ Node #1:

$$\frac{V_1 - V_2}{j40} + \frac{V_1 - V_3}{1/j2500\times10^{-6}} = \frac{8\angle 15^o - V_1}{20}$$

$$-j0.025(V_1 - V_2) + j0.0025(V_1 - V_3) = 0.4e^{j15} - 0.05V_1$$

$$(0.05 - j0.0225)V_1 + j0.025V_2 - j0.0025V_3 = 0.4e^{j15} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

KCL @ Node #2:

$$\frac{V_2 - V_3}{j80} + \frac{V_2}{100} = \frac{V_1 - V_2}{j40}$$

$$-j0.0125(V_2 - V_3) + j0.01V_2 = -j0.025(V_1 - V_2)$$

$$j0.025V_1 + (0.01 - j0.0375)V_2 + j0.0125V_3 = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

KCL @ Node #3:

$$\frac{V_2 - V_3}{j80} + \frac{V_1 - V_3}{1/j2500\times10^{-6}} = \frac{V_3}{50}$$

$$j0.0125(V_2 - V_3) + j0.0025(V_1 - V_3) = 0.02V_3$$

$$j0.0025V_1 + j0.0125V_2 - (0.02 + j0.015) = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

(ii)

Eqns. (1), (2) and (3) can be written in matrix form as shown in Eq. (4):

$$\underbrace{\begin{bmatrix} (0.05 - j0.0225) & j0.025 & -j0.0025 \\ j0.025 & +(0.01 - j0.0375) & +j0.0125 \\ j0.0025 & +j0.0125 & -(0.02 + j0.015) \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}}_{V} = \underbrace{\begin{bmatrix} 0.4e^{j15} \\ 0 \\ 0 \end{bmatrix}}_{I} \quad \ldots\ldots\ldots(4)$$

The MATLAB script is as follows:

```
%This code computes the amount of current flowing through the 100 Ohms resistor
G=[0.05-.0225*j,0.025*j,-0.0025*j; 0.025*j, 0.01-0.0375*j, 0.0125*j; 0.0025*j, 0.0125*j, -(0.02+0.015*j)];
I=[0.4*exp(pi*15/180*j);0;0];
V = inv(G)*I;
I2 = V(2)/100;      % note that V(2) is the second row in the voltage column vector
fprintf('The amount current flowing through the 100 ohms resistor = %5.2f  Amps ', I2)
```

The result obtained after running the script is as follows:

The amount current flowing through the 100 ohms resistor = 0.06 Amps

Mesh analysis (ac)

Since KVL is valid for phasors as its usage will be demonstrated in Example 14

Example 14

For the circuit shown in Fig. 23(a), find the current $i(t)$ and the voltage $V_c(t)$

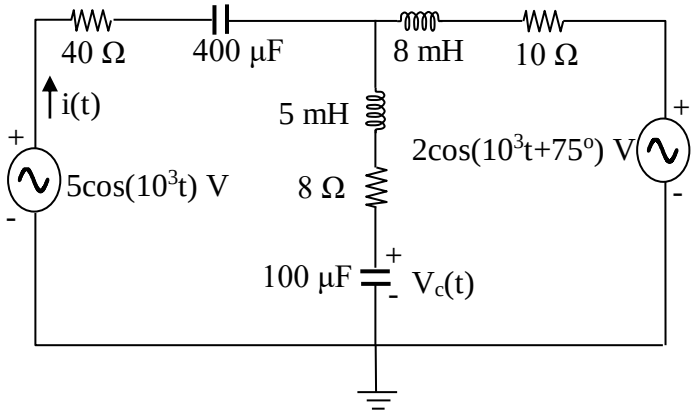


Fig. 23(a)

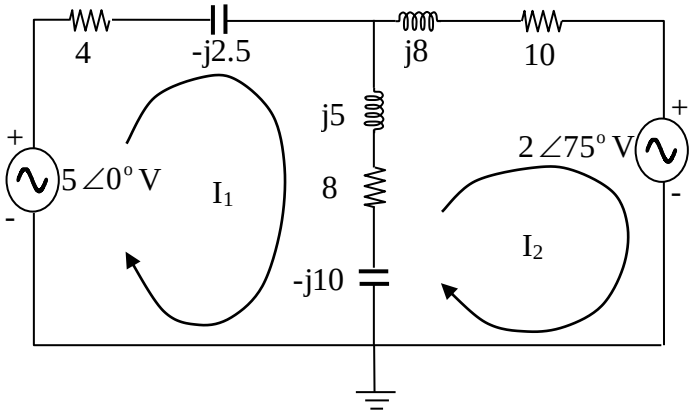


Fig. 23(b)

Solution

We first convert the circuit to the frequency domain:

$$\begin{aligned}
5\cos 1000t &\Rightarrow 5\angle 0^o, \qquad \omega = 1000\,\text{rad/s} \\
2\cos(1000t + 75^o) &\Rightarrow 2\angle 75^o \\
8\,\text{mH} &\Rightarrow j\omega L = j8 \\
5\,\text{mH} &\Rightarrow j\omega L = j5 \\
400\,\mu\text{F} &\Rightarrow \frac{1}{j\omega C} = -j2.5 \\
100\,\mu\text{F} &\Rightarrow \frac{1}{j\omega C} = -j10
\end{aligned}$$

Thus, the frequency domain equivalent circuit is as shown in Fig. 23(b).

KVL @ Mesh #1:

The mesh current is written as

$-5\angle 0^o + (4-j2.5)I_1 + (6+j5-j10)(I_1 - I_2) = 0$

$-5\angle 0^o + 4I_1 - j2.5I_1 + 6I_1 - 6I_2 + j5I_1 - j5I_2 - j10I_1 + j10I_2 = 0$

$(10 - j7.5)I_1 + (-6+j5)I_2 = 5\angle 0^o$ ……(1)

KVL @ Mesh #2:

The mesh current is written as

$(6+j5-j10)(I_2- I_1) + (10+j8)I_2 = -2\angle 75^o$

$6I_2 - 6I_1 + j5I_2 - j5I_1 - j10I_2 + j10I_1 + 10I_2 + j8I_2 == -2\angle 75^o$

$(-6+j5)I_1 + (16+j3)I_2 = -2\angle 75^o$ ……(2)

Eqns. (1) and (2) can be written in matrix form as shown in Eqn. (3):

$$\underbrace{\begin{bmatrix} (10-j7.5) & (-6+j5) \\ (-6+j5) & (16+j3) \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} I_1 \\ I_2 \end{bmatrix}}_{I} = \underbrace{\begin{bmatrix} 5\angle 0^o \\ -2\angle 75^o \end{bmatrix}}_{V} \quad \text{……(3)}$$

The matrix in Eqn. (3) has the form:

[G][I] = [V].

Where the current vector [I] can be determined using the MATLAB command

I = G\V

and the voltage V_C can be obtained as:

$V_c = (-j10)(I_1 - I_2)$

A MATLAB program for determining I_1 and V_c is as follows:

```
% This programs calculates the phasor current I1 and
% phasor voltage Vc.
% G is impedance matrix
% V is voltage vector
% I is current vector
G = [10-7.5*j   -6+5*j;  -6+5*j   16+3*j];
V = [5; -2*exp(j*pi*75/180)]    % voltage vector in column form
I = G\V; % solve for loop currents
I1 = I(1);
I2 = I(2);
Vc = -10*j*(I1 - I2);
I1_abs = abs(I(1));
I1_ang = angle(I(1))*180/pi;
Vc_abs = abs(Vc);
Vc_ang = angle(Vc)*180/pi;
%results are printed by the following lines of code
fprintf('phasor current I1, magnitude: %f \n phasor current I1, angle in degree: %f \n', I1_abs, I1_ang)
fprintf('phasor voltage Vc, magnitude: %f \n phasor voltage Vc, angle in degree: %f \n',Vc_abs,Vc_ang)
```

Running the program gives the following result

```
V =
   5.0000
  -0.5176 - 1.9319i

 phasor current I1, magnitude: 0.387710
 phasor current I1, angle in degree: 15.019255
 phasor voltage Vc, magnitude: 4.218263
 phasor voltage Vc, angle in degree: -40.861691
```

The required current i(t) is

i(t) = I1_abs cos(10^3t+ I1_ang) Amps

i.e., i(t) = 0.388cos(10^3 t + 15.03º) Amps

and the required voltage is $V_c(t)$

$V_c(t)$ = Vc_abs cos(10^3 t - Vc_ang) volts

i.e., $V_c(t)$ = 4.22 cos(10^3 t – 40.86º) volts

Solving differential equations in MATLAB

When KCL and/or KVL are applied to RC or RL circuits, first order differential equations are produced. When the same laws are applied to RLC circuits, second order differential equations are produced. These equations are more difficult to solve than the algebraic equations encountered so far in this course. Four ways through which these equations can be solved using MATLAB are

(i) Numerical Solution using ode solver e.g., *ode23, ode45, ode15s* etc
(ii) Numerical solution using Simulink
(iii) Symbolic solution using *dsolve* command
(iv) Symbolic solution using Laplace transforms methods

The *ode23* function will be considered in this course.

MATLAB Ordinary differential Equation (ODE) solver

When used to solve the equation, the basic syntax for *ode23* command is

[*t, y*] = *ode23*('ydot', tspan, y0)

where ydot is the function file whose inputs must be *t* and *y*, and whose output must be a column vector representing *dy*/*dt*, that is, *f* (*t, y*).

The number of rows in this column vector must be equal to the order of the equation.

The vector tspan contains the starting and ending values of the independent variable *t*, for example tspan = [t0, tfinal], where t0 and tfinal are the desired starting and ending values of *t*.

The parameter y0 is the initial value y(0).

The function file must have its first two input arguments as *t* and *y* in that order. Array operations must not be used in the function file because the ODE solvers call the file with scalar values for the arguments.

Transient Analysis

Four steps involved in the analysis

(i) Solve for initial conditions
(ii) Flip switch and redraw the circuit (if necessary)
(iii) Apply KVL to networks containing inductors or KCL to networks containing capacitors
(iv) Put resulting equations in the standard form and solve

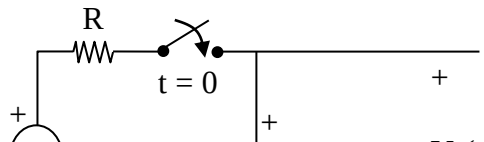
In what follows, the first order and second order circuits will be examined.

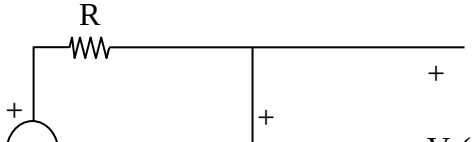
(a) First order circuits

Example 15

Consider the circuit illustrated in Fig. 24(a). V_s = 10 V, R = 10 kΩ, C = 10 μF. Assuming the voltage across the capacitor before the switching action is 0 V (i.e., $V_o(t) = V_o(0^-) = 0$), write a MATLAB program for

(i) computing the output voltage $V_o(t)$ between the interval 0 and 0.4 s
(ii) plotting the output against time between the interval 0 and 0.4 s





Solution
The resulting circuit after the switch has been closed is illustrated in Fig. 24(b)

KCL @ Node V_o:

$$C\frac{dV_o(t)}{dt}+\frac{V_o(t)-V_s}{R}=0$$

Thus

$$\frac{dV_o(t)}{dt}=\frac{V_s}{RC}-\frac{V_o(t)}{RC}=100-10V_o t$$

```
% Transient analysis of RC circuit using ode
% numerical solution
t0 = 0; % start time
tf = 0.4; % end time
xo = 0; % initial condition, i.e., initial output voltage
[t, vo] = ode23('diff1',t0,tf,xo);
% plot the solution
plot(t,vo,'b')
title('State Variable Approach')
xlabel('Time, s'),ylabel('Capacitor Voltage, V'),grid

%The following function is stored in a separate m.file
function dy = diff1(t,y)
dy = 100 - 10*y;
end
```

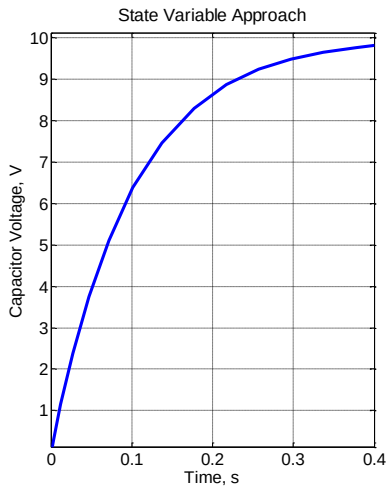


Fig. 24(c)

The result is illustrated in Fig. 24(c)

(b) <u>Second order circuits</u>

<u>Example 16</u>

Consider the circuit illustrated in Fig. 25(a). V_s = 100 V, R = 200 Ω, C = 5 μF, L = 0.5 H. Assuming the voltage across the capacitor before the switching action is 0 V, write a MATLAB program that plots the voltages across each of the 3 circuit elements (i.e., resistor R, inductor L and capacitor C) against time in the interval 0 and 0.1 seconds after the switching action. Also plot the inductor current against time within the afore-mentioned time interval.

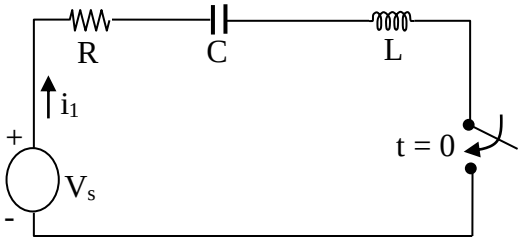


Fig. 25(a)

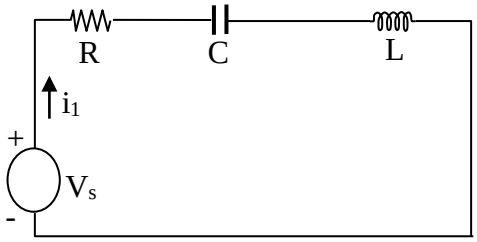


Fig. 25(b)

Solution
The resulting circuit after the switch has been closed is illustrated in Fig. 25(b)

Applying KVL on the resulting circuit:

$$L\frac{di_1}{dt} + i_1 R + \frac{1}{C}\int i_1 dt = V_s$$

$$L\frac{d^2 i_1}{dt} + R\frac{di_1}{dt} + \frac{1}{C} i_1 = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

The MATLAB ode solver can only handle problems of the first order form; therefore, Eqn. (1) needs to be modified to appear in the first order form using substitution.

$$i_2 = \frac{di_1}{dt} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

Substitute $i_2 = {}^{di_1}\!/_{dt}$ in Eqn. (1):

$$L\frac{di_2}{dt} + Ri_2 + \frac{1}{C} i_1 = 0 \quad \ldots\ldots\ldots\ldots\ldots..\ldots\ldots\ldots(3)$$

Make ${}^{di_2}\!/_{dt}$ the subject of Eqn. (3)

$$\frac{di_2}{dt} = -\frac{R}{L} i_2 - \frac{1}{LC} i_1 \ldots\ldots\ldots\ldots\ldots\ldots..\ldots\ldots\ldots(4)$$

Hence the resulting two differential equations are Eqns. (2) and (4) i.e.,

$$i_2 = \frac{di_1}{dt}$$

and

$$\frac{di_2}{dt} = -\frac{R}{L} i_2 - \frac{1}{LC} i_1$$

Some useful expressions:

Inductor voltage:

$$V_L = L\frac{di_1}{dt} = Li_2$$

The resistor voltage:

$$V_r = i_1 R$$

The capacitor voltage (applying KVL):

$$V_c = V_s - V_r - V_l$$

The initial conditions for the solution of the differential equation must be determined.
Physical initial conditions for the time $t = 0$ are known:

$i_1(0^-) = i_1(0^+) = i_1(0) = 0$ (current doesn't flow through a circuit)
$V_c\ (0^-) = V_c\ (0^+) = V_c\ (0) = 0$ (the capacitor isn't charged)

Applying KVL:

$$V_l = V_s - V_r - V_c$$

The following formula applies:

$$V_L = L\frac{di_1}{dt}$$

We substitute V_l into the Kirchhoff's equation:

$$V_L = L\frac{di_1}{dt} = V_s - V_r - V_c \quad \Rightarrow \frac{di_1}{dt} = \frac{V_s}{L} - \frac{V_r}{L} - \frac{V_c}{L}$$

For the time $t = 0^+$

$$\frac{di_1}{dt} = \frac{V_s - V_c(0)}{L} - \frac{V_r}{L}$$

Now we substitute $V_r = iR$ into the equation above

$$\frac{di_1}{dt} = \frac{V_s - V_c(0)}{L} - \frac{Ri_1(0)}{L}$$

After substituting of initial conditions:

$$\frac{di_1}{dt} = \frac{100-0}{0.5} - \frac{200\times 0}{0.5} = \frac{100}{0.5} = 200$$

Hence, the initial conditions for the given system of two the differential equation are therefore as follows:

$$i_1(0) = 0$$

$$\frac{di_1}{dt}(0) = i_2(0) = 200$$

This problem will be solved in MATLAB as follows:

```
function dcurrent = RLC_diff_eq(t,current)
R=200; % R=200 Ohm
L=0.5; % L=0.5 H
C=5e-6; % C=5 microF
dcurrent=[current(2);(-1/C*current(1)-R*current(2))/L];

%function RLC_solution

R=200; % R=200 Ohm
L=0.5; % L=0.5 H
C=5e-6; % C=5 microF
Vs=100; % Vs=100V

[t, current]=ode23('RLC_diff_eq',[0,0.1],[0,(200)]);
subplot(2,2,1) % dividing graph into four parts where the 1st is active
plot(t,current(:,1)) % 2-D line plot
xlabel('time (s)') % x-axis label
ylabel(' Inductor current (A)') % y-axis label
legend('Inductor current (A)') % legend

% the calculations
% the inductor voltage
Vl = L*current(:,2);
```

```
% the resistor voltage
Vr = R*current(:,1);

% the capacitor voltage
Vc = Vs-Vr-Vl;

% the graphs:
subplot(2,2,2) % dividing graph into four parts where the 2nd is active
plot(t, Vc,'r') % 2-D line plot, r - color of line
xlabel('time (s)') % x-axis label
ylabel('Capacitor  Voltage (V)') % y-axis label, character _ creates subscript
legend('Capacitor Voltage (V)') % legend, character _ creates subscript

subplot(2,2,3) % dividing graph into four parts where the 3rd is active
plot(t,Vr,'b') % 2-D line plot, g - color of line
xlabel('time (s)') % x-axis label
ylabel('Resistor Voltage (V)') % y-axis label, character _ creates subscript
legend('Resistor Voltage (V)') % legend, character _ creates subscript

subplot(2,2,4) % dividing graph into four parts where the 4th is active
plot(t,Vl,'b') % 2-D line plot
xlabel(' time (s)') % x-axis label
ylabel('Inductor Voltage (V)') % y-axis label
legend('Inductor Voltage (V)') % legend
```

The result obtained after executing the MATLAB program is illustrated in Fig. 25(c)

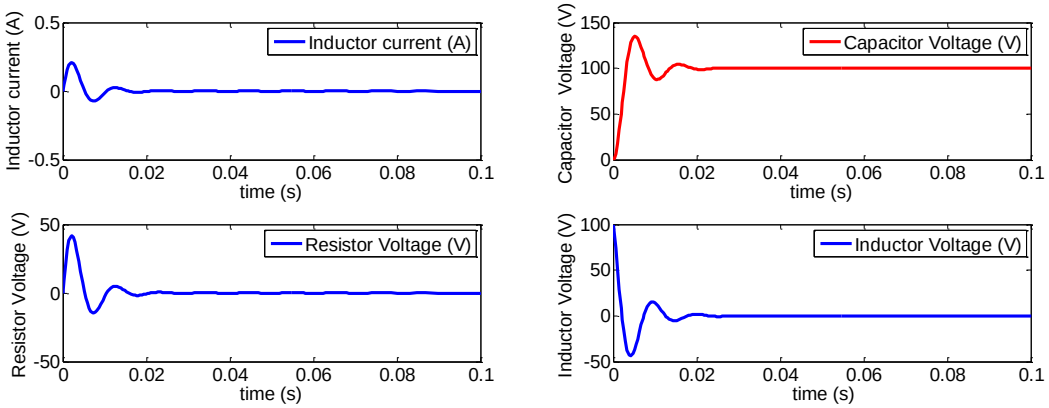


Fig. 25(c)

Network Transfer Characteristics

The transfer function (also called the *network function*) is a useful analytical tool for finding the frequency response of a circuit. The transfer function **H(ω)** of a circuit is the frequency-dependent ratio of a phasor output Y(ω) (an element voltage or current) to a phasor input **X(ω)** (source voltage or current).

Consider the block diagram representation of a system shown in Fig. 26; the transfer function of the system can be expressed mathematically as

$$\mathbf{H}(\omega) = \frac{\mathbf{Y}(\omega)}{\mathbf{X}(\omega)}$$

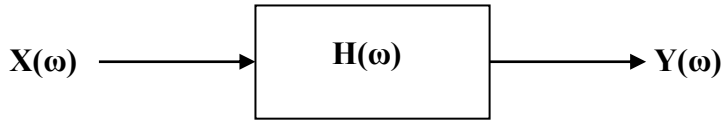


Fig 26 Transfer function

Since the input and output can be either voltage or current at any place in the circuit, there are four possible transfer functions:

$$H(\omega) = \text{Voltage gain} = \frac{V_o(\omega)}{V_i(\omega)}$$

$$H(\omega) = \text{Current gain} = \frac{I_o(\omega)}{I_i(\omega)}$$

$$H(\omega) = \text{Transfer impedance} = \frac{V_o(\omega)}{I_i(\omega)}$$

$$H(\omega) = \text{Transfer admittance} = \frac{I_o(\omega)}{V_i(\omega)}$$

where subscripts *i* and *o* denote input and output values.

The transfer function **H(ω)** can be expressed in terms of its numerator polynomial **N(ω)** and denominator polynomial **D(ω)**

$$\mathbf{H}(\omega) = \frac{\mathbf{N}(\omega)}{\mathbf{D}(\omega)}$$

The roots of **N(ω) = 0** are called the *zeros* of **H(ω)** and are usually represented as $j\omega = z_1, z_2, \ldots$
Similarly, the roots of **D(ω) = 0** of are the *poles* of **H(ω)** and are represented as $j\omega = p_1, p_2, \ldots$

To avoid complex algebra, it is expedient to replace jω temporarily with *s* when working with **H(ω)** and replace *s* with jω at the end.

Frequency response

The frequency response of a circuit is the variation in its behaviour with change in signal frequency.

Being a complex quantity, **H(ω)** has magnitude H(ω) and phase ϕ that is

$$\mathbf{H}(\omega) = H(\omega)\angle\phi \qquad \text{where} \quad H(\omega) = |\mathbf{H}(\omega)| \quad \text{and} \quad \phi = \arg\{\mathbf{H}(\omega)\}$$

The frequency response of a circuit is obtained by plotting the magnitude and phase of the transfer function as the frequency varies.

Steps for obtaining the frequency response:

(i) obtain the frequency-domain equivalent of the circuit by replacing resistors, inductors, and capacitors with their impedances *R*, jωL and $1/j\omega C$

(ii) use any circuit technique to obtain the desired transfer function

(iii) plot the magnitude and phase of the transfer function against frequency

Using MATLAB for Determining Network Transfer Characteristics

The MATLAB function '***roots'*** when used on the numerator and denominator polynomials of the transfer function determines the zeros and poles of the transfer function, respectively.

If the poles and zeros are known, the coefficients of the polynomial of the transfer function can be determined by using the MATLAB function **poly**

The MATLAB function **polyval** is used for polynomial evaluation

Example 17

For the *RLC* circuit in Fig. 27(a), determine the transfer function $H(\omega)=\frac{I_o(\omega)}{I_i(\omega)}$ and write a MATLAB program

(i) computes the poles and zeros of H(ω).
(ii) plot a graph of magnitude of H(ω) against radian frequency using linear scale for y-axis and logarithm scale for the x-axis
(iii) plot a graph of phase of H(ω) against radian frequency using linear scale for y-axis and logarithm scale for the x-axis

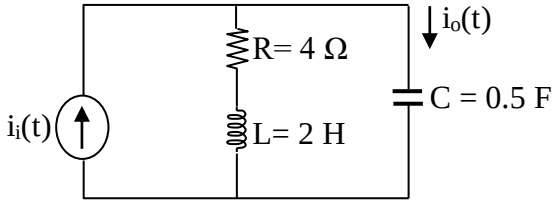


Fig. 27(a)

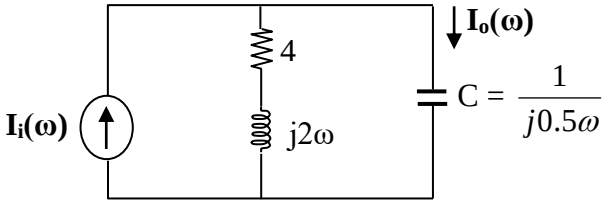


Fig. 27(b)

Solution

By current division.

$$I_o(\omega)=\frac{4+j2\omega}{4+j2\omega+\frac{1}{j0.5\omega}}\times I_i(\omega)$$

The frequency domain representation of the circuit is illustrated in Fig. 27(b)

$$\frac{I_o(\omega)}{I_i(\omega)}=\frac{j0.5\omega(4+j2\omega)}{1+j2\omega+(j\omega)^2}=\frac{2s+s^2}{1+2s+s^2}=\frac{s^2+2s}{s^2+2s+1}$$ where s = jω

The following MATLAB program computes the poles and zeros

```
% Program for poles and zeros
num = [1 2 0];
den = [1 2 1];
disp('the zeros are')
z = roots(num)
disp('the poles are');
p = roots(den);
w = 0:0.01:100;
s = j.*w;
numerator = (s).^2+(2.*s);
```

Alternatively, this will also work

```
numerator = polyval(num,s);
denominator = polyval(den,s);
```

```
denominator = (s).^2+(2.*s)+1;
Hs = numerator./denominator;
Hs_abs = abs(Hs);
subplot(2,1,1)
semilogx (w,Hs_abs)
Hs_ang = angle(Hs)/pi *180;
xlabel('Radian Frequency w rad/s −  log scale'); ylabel(' |G(w)| ')
grid
subplot(2,1,2)
semilogx(w,Hs_ang)
xlabel('Radian Frequency w rad/s −  log scale'); ylabel(' phase (deg) ')
grid
```

The plotted graphs are illustrated in Fig. 27(c) while the computed result obtained on executing the program is as follows:

the zeros are

z =

0

-2

the poles are

p =

-1

-1

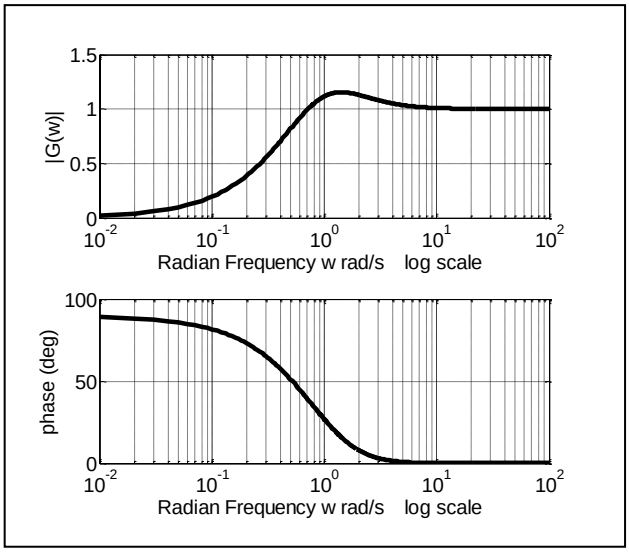


Fig. 27(c)

Example 18

For the circuit shown in Fig. 28(a),

(a) Find the circuit transfer function $H(s)=\dfrac{V_o(s)}{V_i(s)}$

(b) Write a MATLAB program that determines
(i) the poles and zeros of H(s), and
(ii) $V_o(t)$, if $V_s(t) = 10e^{-3t}\cos(2t + 40^o)$

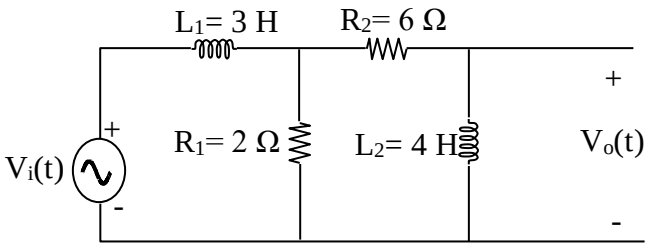


Fig. 28(a)

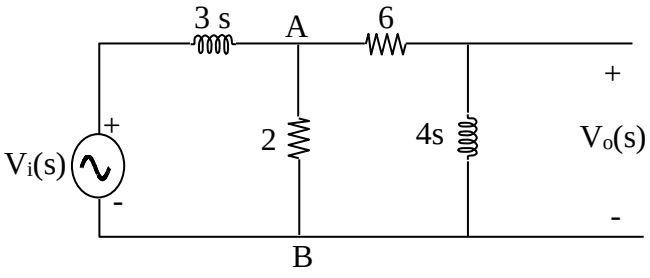


Fig. 28(b)

By voltage divider, the voltage across terminals A and B is

$$V_{AB}=\frac{(6+4s)\|2}{3s+((6+4s)\|2)}\times V_i(s)$$

Therefore,

$$V_o(s)=\frac{4s}{(6+4s)}\times V_{AB}=\frac{4s}{(6+4s)}\times\frac{(6+4s)\|2}{3s+((6+4s)\|2)}\times V_i(s)$$

$$\frac{V_o(s)}{V_i(s)} = \frac{4s}{(6+4s)} \times \frac{(6+4s)\|2}{3s+((6+4s)\|2)}$$

But $\quad (6+4s)\|2 = \frac{2\times(6+4s)}{2+(6+4s)} = \frac{6+4s}{4+2s}$

$$\frac{V_o(s)}{V_i(s)} = \frac{4s}{(6+4s)} \times \frac{(6+4s)\|2}{3s+((6+4s)\|2)} = \frac{4s}{(6+4s)} \times \frac{\left(\frac{6+4s}{4+2s}\right)}{3s+\left(\frac{6+4s}{4+2s}\right)}$$

$$= \frac{24s+16s^2}{24s^3+100s^2+120s+36} = \frac{6s+4s^2}{6s^3+25s^2+30s+9}$$

Given the input voltage

$V_s(t) = 10e^{-3t}\cos(2t + 40^o)$ i.e., $V_m e^{-\alpha t}\cos(\omega t + \phi)$

$\Rightarrow s=-3+j2$

$$\therefore V_o(s) = (10\angle 40^o)H(s)\big|_{s=-3+j2}$$

The following MATLAB program is used to compute the poles, zeros and $V_o(t)$

```
% Program for poles and zeros
num = [4 6 0];
den = [6 25 30 9];
disp('the zeros are')
z = roots(num)
disp('the poles are')
p = roots(den)
% program to evaluate transfer function and
% find the output voltage
s1 = -3+2*j;
n1 = polyval(num,s1);
d1 = polyval(den,s1);
vo = 10.0*exp(j*pi*(40/180))*n1/d1;
vo_abs = abs(vo);
vo_ang = angle(vo)*180/pi;
% print magnitude and phase of output voltage
fprintf('phasor voltage vo, magnitude: %f \n phasor voltage vo, angle in degrees: %f ', vo_abs, vo_ang)
```

The result obtained on executing the program is as follows:

```
the zeros are
z =
     0
-1.5000

the poles are
p =
```

-2.2153
-1.5000
-0.4514

phasor voltage vo, magnitude: 3.453492
phasor voltage vo, angle in degrees: -66.990823

Tutorial Questions

Question 1

Consider the matrix

$$A = \begin{pmatrix} 2 & 10 & 50 \\ -3 & 13 & 7 \\ -6 & 0 & 8 \\ 1 & 4 & 10 \end{pmatrix}$$

Using relational operators, write a MATLAB script that sets all the terms of this matrix which are greater than 2 and less than 15 equal to 0.

Solution

(Approach 1)

```
A =[2,10,50;-3,13,7;-6,0,8;1,4,10];
B = 2*ones(4,3);
C = 15*ones(4,3);
% Make the comparisons
D = B < A; % 1, if the term in A > 2
E = C > A; % 1, if the term in A < 15
% Positions of the numbers which lie between 2 and 15
F = D & E; %The matrix F contains a 1 if the terms of A lie within the given limits; otherwise a 0.
result = F.*A
```

(Approach 2)

```
A =[2,10,50;-3,13,7;-6,0,8;1,4,10];
    for i = 1:4
        for j = 1:3
            if (A(i,j) > 2 && A(i,j) < 15)
                A(i,j) = 0;
            end
        end
    end
disp(A)
```

Question 2

Write a MATLAB script for computing the values of the signal (function)

$s(t) = \sin(2\pi 5t)\cos(2\pi 3t) + e^{-0.1t}$

for a time vector between 0 and 10 with a step size of 0.1.

Solution

```
% Define time vector
t = 0:0.1:10;
% Define signal value
s = sin(2*pi*5*t).*cos(2*pi*3*t) + exp(-0.1*t)
```

```
plot(t,s)
```

Question 3

A 3-bit analog to digital converter, with an analog input x and digital output y, is represented by the equation:

y = 0	x< -2.5
= 1	-2.5 ≤ x < -1.5
= 2	-1.5 ≤ x < -0.5
= 3	-0.5 ≤ x < 0.5
= 4	0.5 ≤ x < 1.5
= 5	1.5 ≤ x < 2.5
= 6	2.5 ≤ x < 3.5
= 7	x≥ 3.5

Based on the MATLAB SWITH…CASE construct, write a function that converts analog signal x to digital signal y.

Solution

The required program is as follows and it is written and saved with the file name "analog_to_digital.m":

```
%{
    analog_to_digital is a function program for obtaining  the digital value given an input analog
    signal
    Note that:
            Y_dig is the digital number (in integer form)
            X_analog is the analog input (in decimal form)
  %}

function   Y_dig = analog_to_digital (X_analog)
      switch true
         case  X_analog < -3.2
                  Y_dig = 0;
         case   X_analog >= -2.5 && X_analog < -1.5
                  Y_dig = 1;
         case X_analog >= -1.5 && X_analog < -0.5
                  Y_dig = 2;
         case  X_analog >= -0.5 && X_analog < 0.5
                  Y_dig = 3;
         case X_analog >= 0.5 && X_analog < 1.5
                  Y_dig = 4;
         case X_analog >= 1.5 && X_analog < 2.5
                  Y_dig = 5;
         case X_analog >= 2.5 && X_analog < 3.5
                  Y_dig = 6;
         otherwise
                  Y_dig = 7;
      end
    Y_dig
end
```

Question 4

Write a MATLAB script that plots the exponential functions $e^{-t/2}$ and $e^{-2t/5}$ against variable *t* (assuming *t* spans the interval [0, 2])
(a) in a single overlay plot,
(b) beside each other in different plots, and
(c) one above the other in different plots.

Solution

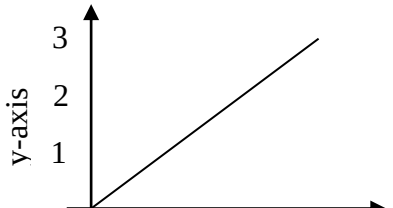
```
t = 0:0.01:2;
f1 = exp(-t/2);
f2 = exp(-2*t/5);

figure
%Question (a): A single overlay plot,
    plot(t,f1,'b',t,f2,'r');

figure
%Question (b): beside each other in different plots
    subplot(1,2,1); % Plot the first function
    plot(t,f1,'b');
    subplot(1,2,2); % Plot the second function next to it
    plot(t,f2,'r');

figure
%Question (c): one above the other in different plots.
    subplot(2,1,1); % Plot the first function
    plot(t,f1,'b');
    subplot(2,1,2); % Plot the second function below the first
    plot(t,f2,'r');
```

Question 5
Sketch the result(s) obtained if the MATLAB script in Fig. Q5 is executed

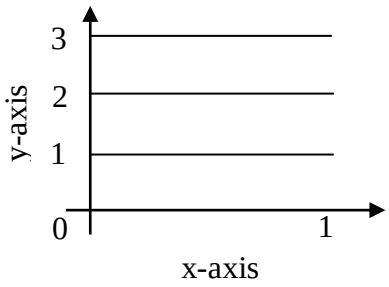
```
A = [0, 1, 2, 3; 0, 1, 2, 3];
B = [0,1,2,3];
C = [0,1];
subplot(2,1,1)
plot(B,A)
xlabel('x-axis')
ylabel('y-axis')
subplot(2,1,2)
plot(C,A)
xlabel('x-axis')
ylabel('y-axis')
```

Fig. Q5

Solution



The length of vector B is equal to the number of columns in matrix A, therefore, the rows in A are plotted against vector B



The length of vector C is equal to the number of rows in matrix A; therefore, the columns in matrix A are plotted against vector C

Question 6

Consider the MATLAB command in Fig. Q6 which contains the academic record of a student.

```
>> Student = {'Ade', '13/30GC1000'; {'ELE447'; 'ELE562'}, [25,45; 22,50]};
```

Fig. Q6

If the name of the student is "ADE", his Matric No is 13/30GC1000, and he registered for the courses "ELE447" and "ELE562", identify the cell where each data is entered and write a command for

(a) Changing his name to "Ade Omofatajeun"

(b) Displaying the courses he registered for

(c) Add 20 to the last number shown in the command of Fig. Q6 (i.e., the number 50)

Solution

(a) Student's name is in cell(1,1) . The following command can be used to change the student's name:

```
Student{1,1} = 'Ade Omofatajeun '
```

(b) The courses registered for are in cell(2,1). The following commands can be used to display the courses:

```
Student{2,1}{1}, Student{2,1}{2}
```

(c) The last number is in cell(2,2). The following command can be used to add 20 to it:

```
Student{2,2}([4]) = Student{2,2}([4]) + 20
```

Question 7

Write a MATLAB script for computing the base 2 and base 10 logarithms of the vector

```
b = (1024  1000   100    2   1)
```

Solution

```
% Define the vector b
b = [1024  1000  100  2  1];
log10ofb = log10(b)   % Calculate the base 10 logarithm
log2ofb = log2(b)     % Calculate the base 2 logarithm
```

Question 8

Write a MATLAB script for plotting the quadratic expression x^2+7x-3 from x equals -3 to +3 in steps of 0.2.

Solution

```
x = -3:0.2:3;
y = x.^2+7.*x-3;
grid on
plot(x,y)
```

Question 9

Write a MATLAB script for computing the sum of the following geometric progression:

$$\sum_{n=1}^{10} 2^n$$

Solution

```
total = 0
for n = 1:10
        total = total + 2 .^ n;
end
```

Question 10

Consider the inverting amplifier circuit shown in Fig. Q10. If the relationship between the output voltage V_o and the input voltage V_g is

$$V_o = \left(-\frac{R_2}{R_1}\right) V_g \quad \text{where} \quad V_g = \frac{10}{\pi^2}\left(\cos 3t + \frac{1}{9}\cos 9t + \frac{1}{25}\cos 15t\right),$$

Write a MATLAB script that

(i) computes the values of voltage V_o, assuming time *t* ranges from 0 to 3 in steps of 0.05.

(ii) plots the values of output voltage V_o computed in part (i) and input voltage V_i against time *t*.

(**Hint**: Assume that V_o is clipped at ±2, which is a constraint imposed by the power supply rails.)

Solution

```
%Question (i)
 t = 0:0.05:3
 r1 = 2e3;
 r2 = 10e3;
y1 = 10/(pi^2);
y2 = cos(3*pi.*t);
y3 = 1/9*cos(9*pi.*t);
y4 = 1/25*cos(15*pi.*t);
vg = y1.*(y2+y3+y4);
 vo = (-r2/r1).*vg;

 for i = 1:length(vo)
   if vo(i)>2
         vo(i)=2;
   elseif vo(i)<-2
         vo(i)=-2;
   end
 end
 [t ; vo]'
```

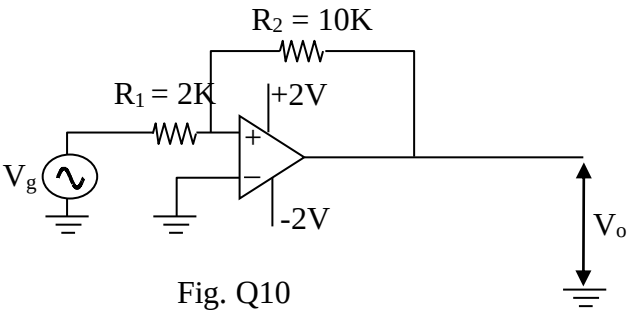
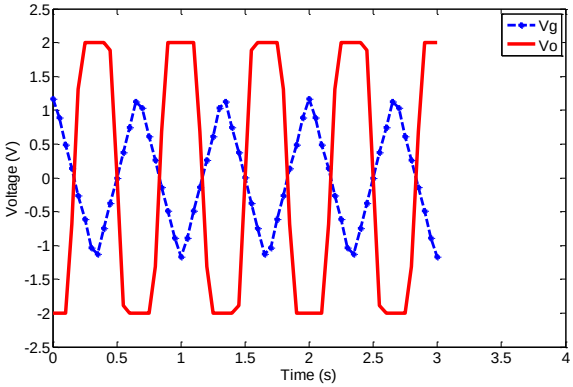


Fig. Q10



```
%Question (ii)
%this script plots vo and vi against t assuming
% that vo has already been computed in part (i)

 plot(t,vg,'b*-- ',t,vo,'r ')
 axis([0 4 -2.5 2.5])
 legend('Vg','Vo')
 xlabel('Time (s)')
 ylabel('Voltage (V)')
```

Question 11

Write a MATLAB script that rounds off the values of the vector

$s(t) = 20 \sin(2\pi 5.3t)$

once up and once towards zero. Assume time vector ranges between 0 and 10 with a step size of 0.1.

Solution

```
% Define the time vector
t = 0:0.1:10;
% Define signal values
s = 20*sin(2*pi*5.3.*t);
% Round s toward infinity (up) using the function ceil
s2infinity = ceil(s)
% Round s toward 0 with the function fix
s2zero = fix(s)
```

Question 12

Determine the result of the following MATLAB command

```
char(2*34-3)
```

Solution

2*34-3 = 65

Referring to Table 11 which contains the ASCII characters and their decimal values, it can be observed that decimal 65 equals ASCII character "A"

Therefore,

char(2*34-3) = char(65) = A

Question 13

Refer to Fig. Q13 and provide the MATLAB command for executing the following:

```
A =[2,10,50;-3,13,7;-6,0,8;1,4,10];
B = [1,2,3;4,5,6;7,8,9,10,11,12];
C = [1,3,5;5,3,1;7,5,3];
```

Fig. Q13

(i) Replace the third row in matrix B with the row vector [12 14 15]
(ii) Raise matrix C to the power of 4
(iii) Raise each element in matrix C to the power of 4
(iv) Add each element in matrix A to a corresponding element in Matrix B
(v) Sum the elements in each column of the matrix **A** and return a row vector containing the sums.
(vi) Interchange the first and last columns of matrix B
(vii) Interchange the second and third rows of matrix A

Solution

(i) B(3,:) = [12 14 15]
(ii) C ^ 4
(iii) C **.**^ 4
(iv) A **.**+ B
(v) sum(A)
(vi) B(:,[3 2 1])
(vii) A([1 3 2 4]; :)

Question 14

For the electric circuit shown in Fig. Q14, compute the impedance Z_{ab} and write a MATLAB script to
(i) Plot the real part of the impedance Z_{ab} versus frequency ω.
(ii) Plot the imaginary part of the impedance Z_{ab} versus frequency ω.
(iii) Plot the impedance Z_{ab} versus frequency ω in polar coordinates.

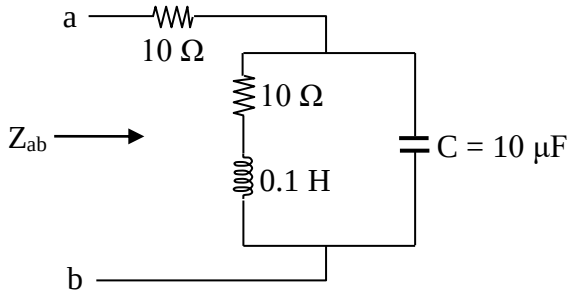


Fig. Q14

Solution

$$Z_{ab} = 10 + \frac{(10 + j0.1\omega)\left(\frac{1}{j10\times10^{-6}\omega}\right)}{10 + j0.1\omega + \frac{1}{j10\times10^{-6}\omega}}$$

$$= 10 + \frac{(10 + j0.1\omega)\left(-j10^{5}\frac{1}{\omega}\right)}{10 + j0.1\omega - j10^{5}\frac{1}{\omega}}$$

$$= 10 + \frac{10^{4} - \frac{j10^{6}}{\omega}}{10 + j\left(0.1\omega - \frac{10^{5}}{\omega}\right)}$$

The required MATLAB program is as follows:

```
w = 0:2000; % Define interval with one radian interval
z = (10+(10 .^ 4 -j .* 10 .^ 6 ./ (w)) ./ (10 + j .* (0.1 .* w -10.^5./ (w))));

%Question (i)
figure
real_part = real(z);
plot(w,real_part)
 grid
xlabel('radian frequency w');
ylabel('Real part of Z');


%Question (ii)
figure
```

```
imag_part = imag(z);
plot(w,imag_part);
grid;
xlabel('radian frequency w');
ylabel('Imaginary part of Z');


%Question (iii)
figure
mag = abs(z); % Computes |Z|
theta = angle(z); % Computes the phase angle of impedance Z
polar(theta,mag); % Polar plot
grid;
ylabel('Polar Plot of Z');
```

Question 15

Refer to the RLC circuit in Fig. Q15 and

(i) determine the transfer function $H(\omega)=\frac{V_o(\omega)}{V_i(\omega)}$

(ii) write a MATLAB script to compute the poles and zeros of the transfer function H(ω).

(iii) write a MATLAB script to plot the frequency response of the circuit.

(4 marks)

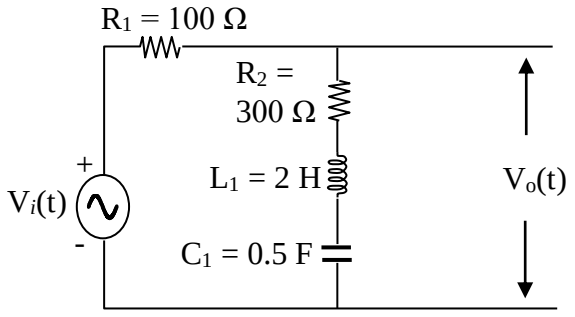


Fig. Q15

Question (i)

Apply voltage divider rule to obtain $V_o(s)$:

$$V_O(s)=\frac{\left(300+2s+\frac{1}{0.5s}\right)}{\left(300+2s+\frac{1}{0.5s}\right)+100}\times V_i(s)$$

Multiply numerator and denominator by 0.5s and rearrange terms:

$$\frac{V_O(s)}{V_i(s)}=\frac{0.5s(300+2s)+1}{0.5s(400+2s)+1}=\frac{s^2+150s+1}{s^2+200s+1}$$

Hence, the transfer function is

$$H(s)=\frac{s^2+150s+1}{s^2+200s+1},$$

Question (ii)

-The zeros are computed by finding the roots of the numerator of the transfer function
The numerator = $s^2 + 150s + 1$
This polynomial is written as row vector [1 150 1] in MATLAB parlance
-The poles are computed by finding the roots of the denominator of the transfer function
The numerator = $s^2 + 200s + 1$
This polynomial is written as row vector [1 200 1] in MATLAB parlance

The following program computes the poles and zeroes of the transfer function

```
% Program for poles and zeros
num = [1 150 1];    % numerator polynomial
den = [1 200 1];    % denominator polynomial

disp('the zeros are')
z = roots(num)       % the zeros of the transfer function
disp('the poles are');
p = roots(den)       % the poles of the transfer function
```

%Question (iii)

```
% Frequency response
num = [1 150 1];    % numerator polynomial
den = [1 200 1];    % denominator polynomial
w = 0:1000;
s = j.*w;

%evaluate the transfer function at s = jw
numerator = polyval(num,s);      %Evaluate transfer function numerator at s = jω
denominator = polyval(den,s);    %Evaluate transfer function denominator at s = jω

Hs = numerator./denominator;
Hs_abs = abs(Hs);         % Compute the magnitude of the transfer function
subplot(1,2,1)
semilogx (w,Hs_abs)
xlabel('Radian Frequency w rad/s − log scale'); ylabel(' |G(w)| ')
grid

Hs_ang = angle(Hs)/pi *180;    % Compute the phase of the transfer function (in degrees)
subplot(1,2,2)
semilogx(w,Hs_ang)
xlabel('Radian Frequency w rad/s − log scale'); ylabel(' phase (deg) ')
grid
```

Alternatively:
numerator = (s).^2+(150.*s) +1 ;
denominator = (s).^2+(200.*s) +1;