***1.*** ***Elementary matrix/array operations:***

1. **a=[12345]** 🡪 creates an array (vector) ‘a’ with only one element 12345.
2. **a=[10 20 30 40 50]** 🡪 creates an array (vector) ‘a’ with 5 elements 10,20,30,40,50.

**b=[2 2 2 2 2]** 🡪 creates an array (vector) ‘b’ with 5 elements 2,2,2,2,2.

1. **a(2) 🡪** displays the second element 20 in the array;

**a(1:3)** 🡪 displays the elements from index 1-3 i.e 10,20,30

1. **c = a .\* b** 🡪 multiplies array (vector) ‘a’ with ‘b’ and stores result in the array ‘c’

**c = a + b** 🡪 adds array (vector) ‘a’ with ‘b’ and stores result in the array ‘c’

1. **M1=[1 2 3; 4 5 6; 7 8 9]** 🡪 creates a 3x3 integer matrix ‘M1’

**M2=[10 20 30; 40 50 60; 70 80 90]** 🡪 creates a 3x3 integer matrix ‘M2’

**M3=[1.1 2 3.3; 4 5.5 6; 7 8.8 9.9]** 🡪 creates a 3x3 real matrix ‘M3’

**M4=[5+3j 2+4j 7+8j; 9+2j 1+4j -2-5j; 11+2j -4+6j 8+2j]** 🡪 creates a 3x3 complex matrix ‘M4’

1. **M1(2 , :)** 🡪 displays the 2nd row of M1

**M1(: , 3)** 🡪 displays the 3rd column of M1

**M1(1,3)🡪** displays the element of 1st row and 3rd column i.e ‘3’

**Y=M1(3, [2 3])**🡪 extracts the 3rd row,2nd column element & 3rd row, 3rd column element

**M1(1:2,2:3) 🡪** extracts the sub-matrix

**M1 (M1>8)** 🡪 displays all the elements greater than 8

1. **M1\_trans = M1'** 🡪 computes transpose of matrix ‘M1’ and stores result in the matrix ‘M1\_trans’

**M1\_det = det(M1)** 🡪 computes determinant of matrix ‘M1’ and stores the result in variable ‘M1\_det’

**M1\_inv = inv(M1)** 🡪 computes inverse of matrix ‘M1’ and stores result in the matrix ‘M1\_inv’

**MUL = M1\*M2** 🡪multiplies matrix ‘M1’ with ‘M2’ and stores result in the matrix ‘MUL’

1. **Concatenating two matrices:**

**M3 = [M1 M2] 🡪** Concatenates Matrix ‘M1’ and ‘M2’ Horizontally

**M3 = [M1 ;M2] 🡪** Concatenates Matrix ‘M1’ and ‘M2’ Vertically

1. **Adding a row to matrix:**

**M1(4,: ) = [9 9 9] 🡪** Adds the 4th row to matrix ‘M1’ with 3 element 9,9,9

1. **Deleting a row from matrix:**

**M1(4,: ) = [ ] 🡪** Deletes the 4th row from matrix ‘M1’

1. **Deleting a column from matrix:**

**M1(: ,3 ) = [ ] 🡪** Deletes the 3rd column from matrix ‘M1’

***2. Matrix Generation Functions:***

1. **zeros(3,3) 🡪** 3x3 matrix of zeroes
2. **ones(2,2)** 🡪 2x2 matrix of ones
3. **eye(3) 🡪**  identity matrix of size 3
4. **rand(5) 🡪** 5x5 matrix of random numbers
5. **V=1:2:10🡪** creates a vector V with elements in the range 1 to 10 with the increment 2
6. **V = linspace(**[**x1,x2**](file:///C:\Program%20Files\MATLAB\R2015a\help\matlab\ref\linspace.html?searchHighlight=linspace#inputarg_x1x2)**,y) 🡪** will generate a vector V in the range x1 to x2 with ‘y’ elements in it

***3. Few useful commands:***

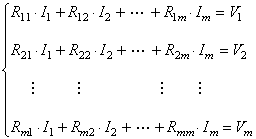
1. **who** 🡪 this command lists variables in the current workspace
2. **whos** 🡪 this command lists variables and their sizes in the current workspace
3. **help** 🡪 this command provides the description about any command
4. **lookfor** 🡪 this command becomes useful when one is not sure of the MATLAB command name
5. **clc 🡪** this command clears the command window
6. **save FILENAME** 🡪 this command saves all the variables of the current workspace in the file ‘FILENAME.mat’
7. **load FILENAME** 🡪 this command loads all the variables of the file ’FILENAME’ in to the current workspace
8. **clear** 🡪 this command clears all the variables from current workspace
9. **pwd** 🡪 this command prints the present working directory
10. **disp(‘STRING’)** 🡪 this function displays the STRING in the command window

***4. Find the loop currents of the circuit given:***

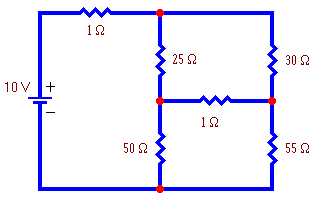
 The steps in the loop current method are:

* Count the [number of loop currents required](http://mathonweb.com/help/backgd4b.htm). Call this number *m*.
* [Choose *m* independent loop currents](http://mathonweb.com/help/backgd4c.htm), call them *I*1, *I*2, . . . , *Im* and draw them on the circuit diagram.
* Write down [Kirchhoff's Voltage Law](http://mathonweb.com/help/backgd2.htm#Kirchoff's Voltage Law) for each loop. The result, after simplification, is a system of *n* linear equations in the *n* unknown loop currents in this form:

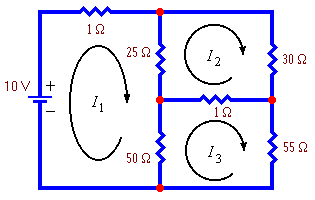
[**Kirchhoff's Voltage Law**](http://mathonweb.com/help/backgd2.htm#Kirchoff's Voltage Law)

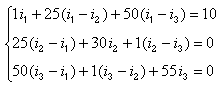
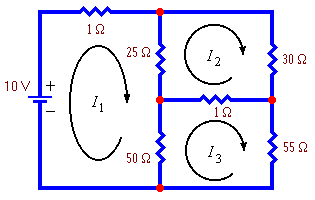


**Example:** Find the loop current flowing in each branch of this circuit.

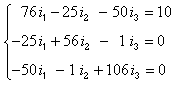


**Solution:1** 🡪Using Command Window

* The number of loop currents required is 3.  
  
* We will choose the loop currents I1, I2, I3 as shown in the figure below.
* Write down Kirchhoff’s Voltage Law for each loop. The result is the following system of equations:



Collecting terms this becomes:



**Solving the equations using matrix:**

**76 -25 -50 i1 10**

**Z = -25 56 -1 i2 V = 0**

**-50 -1 106 i3 0**

Therefore, **i = inv(Z) \* v**

**Solution:2 🡪Creating Script Files**

1. Editing a file*:***Home Menu New****Script**, invokes MATLAB Editor/Debugger.
2. Save file with extension ***.m***
3. Edit the following commands in the Editor and save as **ex\_1b**.

**clear; clc;**

**% Solution of Network equations**

**Z = [170 -100 -30; -100 160 -30; -30 -30 70];**

**v = [0; 0; 10];**

**disp('The mesh currents are : ')**

**i = inv(Z)\*v**

A line beginning with % sign is a comment line.

1. Typing **ex\_1b** at the command prompt will run the script file, and all the 7 commands will be executed sequentially. The script file can also be executed from the **Run** button the MATLAB Editor.

**Solution:3** 🡪 **Interactive Data Inputs**

1. Edit the following commands in the Editor and save as **ex\_1c**.

**clear; clc;**

**% Solution of Network equations**

**Z = input('Enter Z : ');**

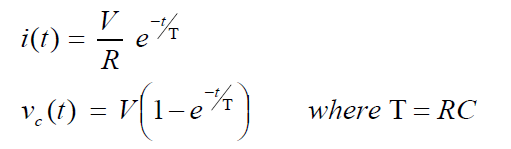
**v = input('Enter v : ');**

**i = Z\v;**

**disp('The results are : ')**

**fprintf('i1 = %f A, i2 = %f A, i3 = %f A \n', i(1),i(2),i(3))**

***5. Find the transient response of the circuit given:***

1. Edit the following commands in the Editor and save as **ex\_1d**.

**clear; clc;**

**disp(' RC transient analysis')**

**v = input(' Enter source voltage : ');**

**r = input(' Enter value of resistance :');**

**c = input(' Enter value of capacitance: ');**

**T = r\*c;**

**fprintf('\n The results are : \n\n')**

**disp('t (sec) i (A) v\_c (V)')**

**for n = 1:2:11**

**t(n) = n; % Time interval in Seconds**

**i(n) = (v/r)\* exp(-t(n)/T);**

**v\_c(n) = v\* (1 - exp(-t(n)/T));**

**fprintf('%6.4f\t%6.4f\t%6.4f\n', t(n),i(n),v\_c(n))**

**end;**

***6. Function File:***

* A *function file* is also an m-file, just like a script file, except it has a **function definition line** at the top that defines the input and output explicitly.

***Function[output\_list] = function\_name (input\_list)***

* Save the file as ***fname.m***
* The filename will become the name of the new command for MATLAB.

***Ex:- Function File to Display Name and Age***

1. Edit the following in the Editor and save as **info.m**

**function [name, age] = info( n, a)**

**name = n;**

**age = a;**

**end**

1. Executing the function *info* in the *Command Window* as follows:

**[name,age] = info('John', 32)**

***7. Write a function factorial to compute factorial of N***

1. Edit the following in the Editor and save as **fact.m**

**function [f]= fact(n)**

**Fct=1;**

**for i=1:n**

**Fct=Fct \* i;**

**end**

**display ('Factorial of N is');**

**display(Fct);**

**f=Fct**

**end**

1. Executing the function *fact* in the *Command Window* as follows:

**Res = fact(5)**