
**Faculty of Engineering**
**Department of Electrical & Computer Engineering (ECE)**

# Control Systems (ECE 331)

## Experiment No: 01

"Analysis and Simulation of Control Systems using MATLAB"

**Submitted By:**

_____

University ID No: (  )

Date of Submission:

Signature of Faculty: Marks:
Name of Faculty: Ankit Patel

# Control Systems ECE 331

Experiment 01

Analysis and Simulation of Control Systems Using MATLAB

## Contents

# 1 Introduction

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s.He designed it to give his students access to LINPACK and EISPACK without them having to learn Fortran. It soon spread to other universities and found a strong audience within the applied mathematics community. Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and founded MathWorks in 1984 to continue its development.

MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra, numerical analysis, and is popular amongst scientists involved in image processing.

■ **What is MATLAB??**

MATLAB (MATrix LABoratory) is a high performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

The main functions of MATLAB includes:

- Math and Computation
- Algorithm development
- Modeling, Simulation and Prototyping
- Data Analysis, Exploration and Visualization
- Scientific and Engineering Graphics
- Application development, including GUI building

The MATLAB consists of five main parts:

1. The MATLAB Language: This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.It allows both ”programming in the small” to rapidly create quick and dirty throw-away programs, and ”programming in the large” to create complete large and complex application programs.
2. The MATLAB working environment: This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing

and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

3. Handle Graphics: It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.
4. The MATLAB mathematical function library: This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen-values, Bessel functions, and fast Fourier transforms.
5. The MATLAB Application Program Interface (API): This is a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

⊡ **Starting of MATLAB :**

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut icon on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:

- The Command Window
- The Command History
- The Workspace
- The Current Directory
- The Help Browser
- The Start Button

When MATLAB is started for the first time, the screen looks like the one that shown in the Figure 1. This illustration also shows the default configuration of the MATLAB desktop.

# 2 Polynomials, Vectors & Matrix Operations

## □ Polynomials:

Since polynomials occur frequently in mathematics and engineering, MATLAB has a collection of very useful functions for working with them.

**`Example:1`**

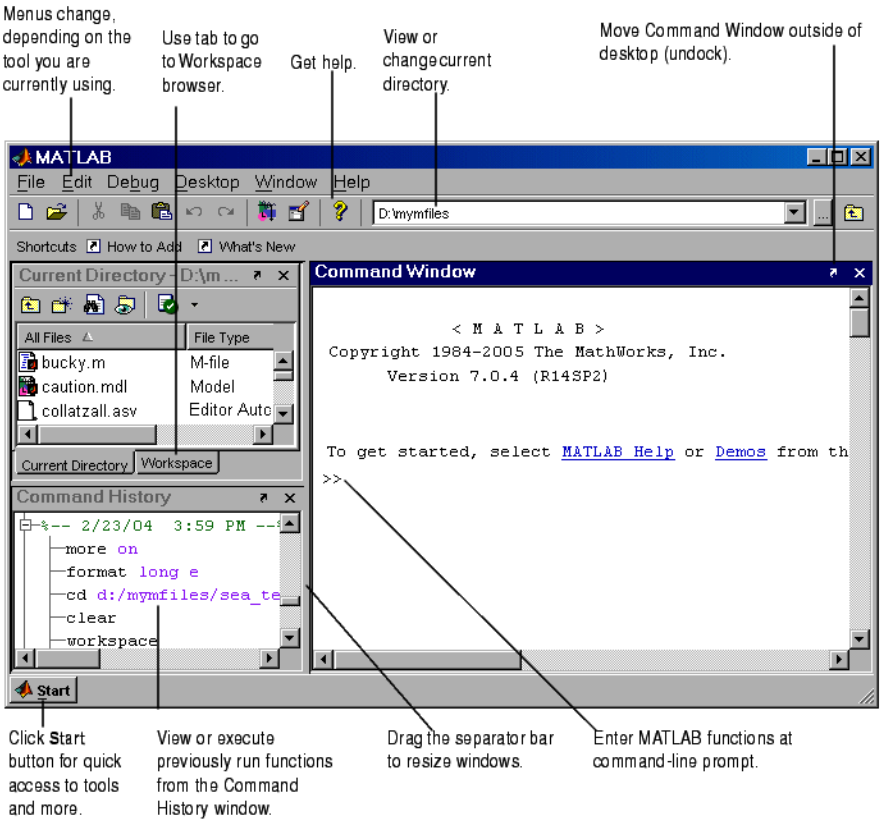


Figure 1: The graphical interface to the MATLAB workspace

$a(x) = x^2 + 2x + 1$ and $b(x) = x^2 + 4x + 2$

Find Convolution or Multiplication of two functions $a(x)$ *and* $b(x)$.

**Answer:**

```
>> a=[1 2 1]

a =

     1     2     1

>> b=[1 4 2]

b =

     1     4     2

>> y=conv(a,b)

y =
```

```
     1     6    11     8     2
```

It means that the final answer is: $y = x^4 + 6x^3 + 11x^2 + 8x + 2$

**How to find roots of polynomials?**

**Example:2**

$f(x) = 2x^4 + 5x^3 + 10x^2 + 7x + 6$

**Answer:**

```
>> a=[2 5 10 7 6]

a =

     2     5    10     7     6

>> r=roots(a)

r =

  -1.0000 + 1.4142i
  -1.0000 - 1.4142i
  -0.2500 + 0.9682i
  -0.2500 - 0.9682i
```

## □ Vectors:

A vector is a one-dimensional array of numbers. MATLAB allows creating two types of vectors:

1. Row Vectors
2. Column Vectors

**Row Vectors:**

Row vectors are created by enclosing the set of elements in square brackets, using space or comma to delimit the elements.

```
if,

a=[1 2 3 4 5]

MATLAB will show as

>> a=[1 2 3 4 5]
```

```
a =

     1     2     3     4     5
```

**Column Vectors:**

Column vectors are created by enclosing the set of elements in square brackets, using semicolon to delimit the elements.

if,

```
a=[1; 2; 3; 4; 5]
```

MATLAB will show as

```
>> a=[1; 2; 3; 4; 5]

a =

     1
     2
     3
     4
     5
```

**Addition & Substraction Vectors:**

**Example:**

if A=[1 3 5 7]

and

```
B=[2 4 6 8]
```

Then A+B will execute as:

```
>> A=[1 3 5 7]

A =

     1     3     5     7

>> B=[2 4 6 8]

B =

     2     4     6     8
```

```
>> X=A+B

X =

     3     7    11    15
```

And A-B will be

```
>> Y=A-B

Y =

    -1    -1    -1    -1
```

The combine result of addition and substraction will display as

```
>> disp(X);
     3     7    11    15

>> disp(Y);
    -1    -1    -1    -1
```

**Scalar Multiplication of Vectors:**

When you multiply a vector by a number, this is called the scalar multiplication. Scalar multiplication produces a new vector of same type with each element of the original vector multiplied by the number.

**Example:**

```
>> v=[20 10 30 55]

v =

    20    10    30    55

>> m=v*11

m =

   220   110   330   605
```

**Transpose of Vectors:**

The transpose operation changes a column vector into a row vector and vice versa. The transpose operation is represented by a single quote (').

**Example:**

```
>> q=[1 3 6 9]

q =

     1     3     6     9

>> tr=q'

tr =

     1
     3
     6
     9
```

**Magnitude of Vectors:**

**Example:**

```
>> v = [1: 2: 20];
sv = v.* v; %the vector with elements % as square of v's elements
dp = sum(sv); % sum of squares -- the dot product
mag = sqrt(dp); % magnitude
>> disp('Magnitude:'); disp(mag);


Magnitude:
   36.4692
```

**Dot Product of Vectors:**

**Example:**

```
>> v1 = [2 3 4];
v2 = [1 2 3];
dp = dot(v1, v2);
disp('Dot Product:'); disp(dp);


Dot Product:
    20
```

**Plotting of Vectors:**

**Example 1:**

```
>> x = [1 2 3 4 5 6];
y = [3 -1 2 4 5 1];
plot(x,y)
```

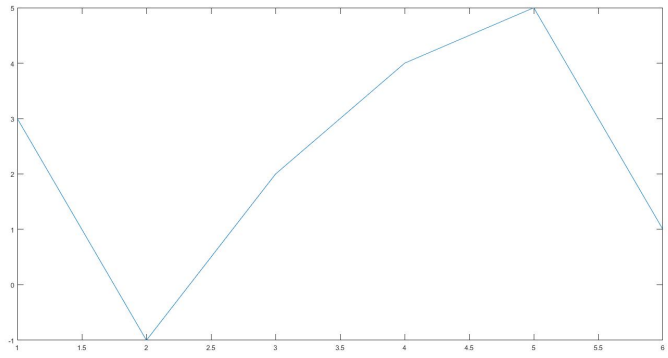

Figure 2: Vector Plotting

**Example 2:**

Write the following in script file, known as .m file in MATLAB

```
x = 0:pi/100:2*pi;
 y = sin(x);
 plot(x,y)
 xlabel('x = 0:2\pi')
 ylabel('Sine of x')
 title('Plot of the Sine function')
```

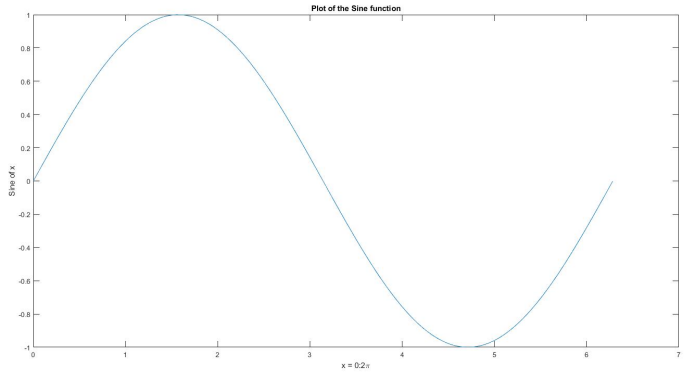


Figure 3: Sine Function Plot

**Example 3:**

How to plot multiple functions in MATLAB

```
Write the following in script file, known as .m file in MATLAB

x = 0:pi/100:2*pi;
y1 = 2*cos(x);
y2 = cos(x);
y3 = 0.5*cos(x);
plot(x,y1,'--',x,y2,'-',x,y3,':')
xlabel('0 \leq x \leq 2\pi')
ylabel('Cosine functions')
legend('2*cos(x)','cos(x)','0.5*cos(x)')
title('Typical example of multiple plots')
axis([0 2*pi -3 3])
```

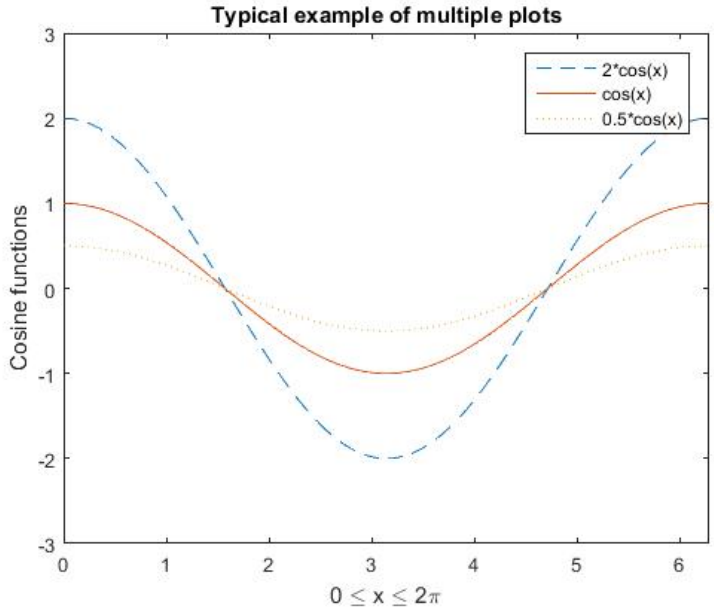


Figure 4: Typical example of multiple plots

## □ Matrix:

We use matrices in mathematics and engineering because often we need to deal with several variables at onceeg the coordinates of a point in the plane are written (x, y) or in space as (x, y, z) and these are often written as column matrices in the form:

$$\begin{pmatrix} x \\ y \end{pmatrix} \text{ or } \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

It turns out that many operations that are needed to be performed on coordinates of points are linear operations and so can be organized in terms of

rectangular arrays of numbers, matrices. Then we find that matrices themselves can under certain conditions be added, subtracted and multiplied so that there arises a whole new set of algebraic rules for their manipulation.

A matrix is an array of numbers. To type a matrix into MATLAB you must:

- begin with a square bracket, [
- separate elements in a row with spaces or commas (, )
- use a semicolon (; ) to separate rows
- end the matrix with another square bracket, ].

Here is a typical example. To enter a matrix A, such as,

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

type,

```
>> A = [1 2 3; 4 5 6; 7 8 9]

MATLAB will display matrix as below
A =

     1     2     3
     4     5     6
     7     8     9
```

* **Matrix Operations:**
  - Addition and Substraction of Matrices
  - Division of Matrices
  - Scalar Operation of Matrices
  - Transpose of Matrix
  - Matrix Multiplication
  - Determinant of Matrix
  - Inverse of a Matrix

⊙ Addition and Substraction of Matrices

Create a script file with the following code:

```
a = [ 1 2 3 ; 4 5 6; 7 8 9];
b = [ 7 5 6 ; 2 0 8; 5 7 1];
c = a + b
d = a - b

The MATLAB shows as the following
```

```
c =

     8     7     9
     6     5    14
    12    15    10


d =

    -6    -3    -3
     2     5    -2
     2     1     8
```

⊙ Division of Matrices

Create a script file with the following code:

```
a = [ 1 2 3 ; 4 5 6; 7 8 9];
b = [ 7 5 6 ; 2 0 8; 5 7 1];
c = a / b
d = a \ b
```

```
c =

   -0.5254    0.6864    0.6610
   -0.4237    0.9407    1.0169
   -0.3220    1.1949    1.3729

d =

   -0.3603   -0.5404    0.4053
    0.7206    1.0809   -0.8106
   -0.3603   -0.5404    0.4053
```

⊙ Scalar operations of Matrices
When you add, subtract, multiply or divide a matrix by a number, this is called the scalar operation. Scalar operations produce a new matrix with same number of rows and columns with each element of the original matrix added to, subtracted from, multiplied by or divided by the number.

```
a = [ 10 12 23 ; 14 8 6; 27 8 9];
b = 2;
c = a + b
d = a - b
e = a * b
f = a / b
```

```
c =

    12    14    25
    16    10     8
    29    10    11


d =

     8    10    21
    12     6     4
    25     6     7


e =

    20    24    46
    28    16    12
    54    16    18


f =

    5.0000    6.0000   11.5000
    7.0000    4.0000    3.0000
   13.5000    4.0000    4.5000
```

⊙ Transpose of Matrix

The transpose operation switches the rows and columns in a matrix. It is represented by a single quote(′).

```
a = [ 10 12 23 ; 14 8 6; 27 8 9]
b = a'

a =

    10    12    23
    14     8     6
    27     8     9


b =

    10    14    27
    12     8     8
    23     6     9
```

⊙ Matrix Multiplication

Consider two matrices A and B. If A is an *mxn* matrix and B is an n x p matrix, they could be multiplied together to produce an *mxn* matrix C. Matrix multiplication is possible only if the number of columns n in A is equal to the number of rows n in B.

```
a = [ 1 2 3; 2 3 4; 1 2 5]
b = [ 2 1 3 ; 5 0 -2; 2 3 -1]
prod = a * b

a =

     1     2     3
     2     3     4
     1     2     5


b =

     2     1     3
     5     0    -2
     2     3    -1


prod =

    18    10    -4
    27    14    -4
    22    16    -6
```

⊙ Determinant of Matrix

Determinant of a matrix is calculated using the det function of MATLAB. Determinant of a matrix A is given by det(A).

```
a = [ 1 2 3; 2 3 4; 1 2 5]
det(a)

a =

     1     2     3
     2     3     4
     1     2     5


ans =

    -2
```

⊙ Inverse of Matrix

The inverse of a matrix A is denoted by $A^{-1}$ such that the following relationship holds:

$A \cdot A^{-1} = A^{-1} \cdot A = 1$

The inverse of a matrix does not always exist. If the determinant of the matrix is zero, then the inverse does not exist and the matrix is singular. Inverse of a matrix in MATLAB is calculated using the inv function. Inverse of a matrix A is given by inv(A).

```
a = [ 1 2 3; 2 3 4; 1 2 5]
inv(a)

a =

     1     2     3
     2     3     4
     1     2     5


ans =

   -3.5000    2.0000    0.5000
    3.0000   -1.0000   -1.0000
   -0.5000         0    0.5000
```

# 3 Transfer Functions

To find or plot the poles and zeros of any transfer function [TF], the PZMAP command can be used as shown. Suppose the TF is given by

$$G(s) = \frac{s+1}{s^2+s+4}$$

The MATLAB program of above transfer function as under:

```
num=[ 1 1];den=[1 1 4];
[p,z]=pzmap(num,den)

p =

  -0.5000 + 1.9365i
  -0.5000 - 1.9365i


z =
```

-1

Where p = Poles and z = Zeros of the systems.

If we want to go for pole-zero map, then type in script file or command window

```
>> pzmap([ 1 1],[1 1 4])
```

and we get

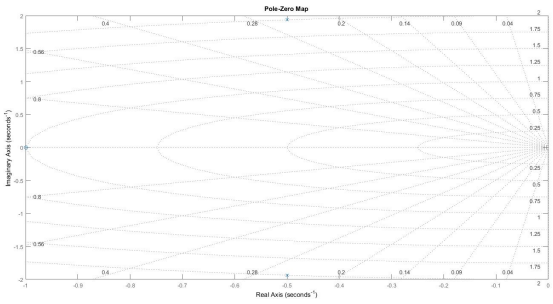


Figure 5: Location of Poles and Zeros for given Transfer Function

If we want to draw a step response of the given transfer function, write down the following command in .m file of MATLAB,

```
num=[ 1 1];den=[1 1 4];
>> t=[0:0.01:5];
step(num,den)
```

It will show the step response as:

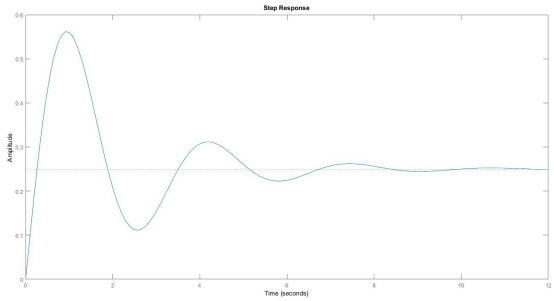


Figure 6: Step Response for given Transfer Function