PREMIER UNIVERSITY

Department of Computer Science & Engineering

Course Code	:	**EEE-202**
Course Title	:	**Signals & Systems Laboratory**
Report No	:	01
Name of Report	:	Familiarization With MATLAB and Functions in MATLAB
Date of Performance	:	18/09/2023
Date of Submission	:	09/12/2023
Course Instructor	:	Mohammed Saifuddin Munna

Submitted By :

Name	**: Mohammad Hafizur Rahman Sakib**
ID	**: 0222210005101118**
Section	**: C**
Semester	**: 4th Semester**
Session	**: Fall 2023**

REMARKS

PREMIER UNIVERSITY

Department of Computer Science & Engineering

Course Code	: **EEE-202**
Course Title	: **Signals & Systems Laboratory**
Report No	: 02
Name of Report	: Matrix Operationas In MATLAB
Date of Performance	: 18/09/2023
Date of Submission	: 09/12/2023
Course Instructor	: Mohammed Saifuddin Munna

Submitted By :

Name	: **Mohammad Hafizur Rahman Sakib**
ID	: **0222210005101118**
Section	: **C**
Semester	: **4th Semester**
Session	: **Fall 2023**

REMARKS

PREMIER UNIVERSITY

Department of Computer Science & Engineering

Course Code	: **EEE-202**
Course Title	: **Signals & Systems Laboratory**
Report No	: 03
Name of Report	: Plotting Various Signals On MATLAB
Date of Performance	: 25/09/2023
Date of Submission	: 09/12/2023
Course Instructor	: Mohammed Saifuddin Munna

Submitted By :

Name	: **Mohammad Hafizur Rahman Sakib**
ID	: **0222210005101118**
Section	: **C**
Semester	: **4th Semester**
Session	: **Fall 2023**

REMARKS

PREMIER UNIVERSITY

Department of Computer Science & Engineering

Course Code	:	**EEE-202**
Course Title	:	**Signals & Systems Laboratory**
Report No	:	04
Name of Report	:	Operation of Signals and Transformation of Independent Variables
Date of Performance	:	02/12/2023
Date of Submission	:	09/12/2023
Course Instructor	:	Mohammed Saifuddin Munna

Submitted By :

Name	:	**Mohammad Hafizur Rahman Sakib**
ID	:	**0222210005101118**
Section	:	**C**
Semester	:	**4th Semester**
Session	:	**Fall 2023**

REMARKS

PREMIER UNIVERSITY

Department of Computer Science & Engineering

Course Code	:	**EEE-202**
Course Title	:	**Signals & Systems Laboratory**
Report No	:	05
Name of Report	:	Observation of convolution in MATLAB
Date of Performance	:	09/12/2023
Date of Submission	:	23/12/2023
Course Instructor	:	Mohammed Saifuddin Munna

Submitted By :

Name	**: Mohammad Hafizur Rahman Sakib**
ID	**: 0222210005101118**
Section	**: C**
Semester	**: 4th Semester**
Session	**: Fall 2023**

REMARKS

Experiment No: 01

Name of the experiment: Familiarization with MATLAB and Functions in MATLAB

Objective: To familiarize the students with MATLAB

Software Requirement: MATLAB 2014

Theory:

Introduction to MATLAB

MATLAB is a high-level technical computing language equipped with a user-friendly interface. Its name stems from the words ***matrix*** and ***Laboratory*** as it is based on the use of matrices. MATLAB is an extremely powerful tool useful for scientists and engineers from various disciplines. For example, MATLAB can be used in a wide range of applications, such as telecommunications, signal and image processing, control, mathematics, financial modelling, bioengineering, aeronautics, and many more.

M-Files

In order to write many commands that are executed all together, the program must be written in a text editor. In this editor, one can type all the needed commands to form a program, save the program, and execute it any time he or she wants. The text files are called *M-files* due to their suffice * .m.

There are two categories of M-files: the Scripts and the Functions.

Scripts

Scripts are the M-files with MATLAB commands. Their name must have a .m suffix. Scripts are suitable for solving the problems that require many commands. The advantage of the scripts is that they are implemented very easily.

Functions

Function are also M-files, That is, are files with extension .m and must be saved in the current Directory of MATLAB. The difference between functions and scripts is that a function accepts one or more input arguments and returns one or more output arguments. To declare that an M-file is a function the first line of the m file must contain the syntax definition. More specifically, the first line of the M-file must be of the form function[y1,y2,y3,…yn]=name{x1,x2,x3… xm}. The variable y1,y2,…yn are the outputs of the function while x1,x2,…xm are the input arguments. In case there is only one output, the square brackets are not necessary. The "name" specifies the name of the function. In order to execute a function, first the M-File is saved in Current Directory.

Useful Commands

Here we will learn and practice useful (when working with vectors and matrices) commands. As already discussed, the command sumreturns the sum of the elements of a vector. The command cumsumreturns a vector whose elements are the cumulative sum of the previous elements, the command prodis the product of the vector elements, while the command diffreturns a vector in which each element is given by its subtraction with the previous element. The command max and min return the largest and smallest elements of the vector, respectively, as well as their index. The command sortsorts the vector elements in ascending (by default) or descending order. The command mean computes the mean value, while the command medianreturns the median value. All these commands are suitable also for matrices by slightly changing their syntax.

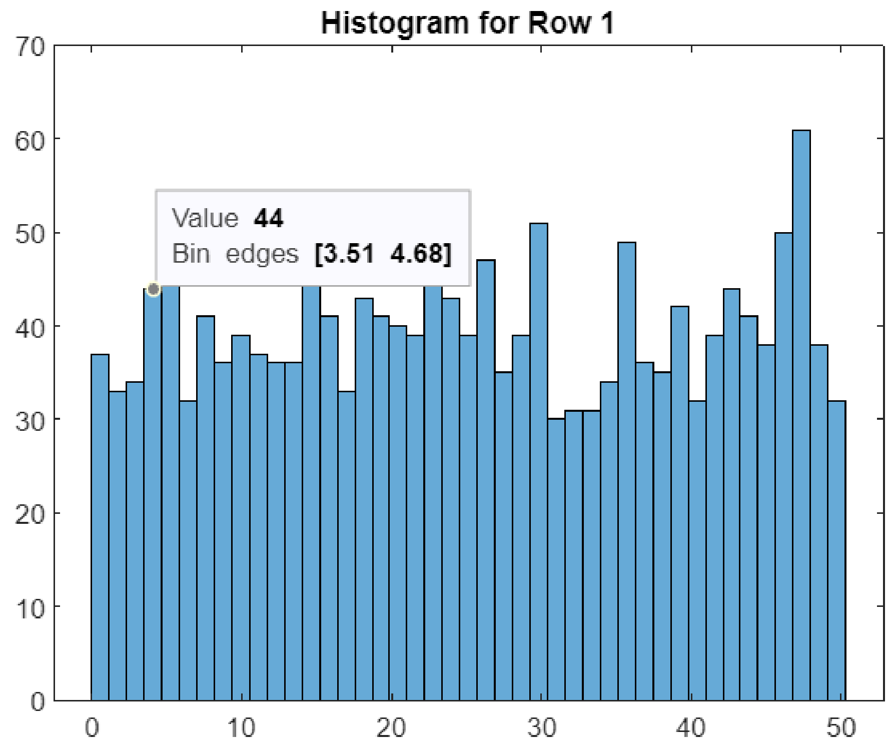
Lab Tasks:

1. Write a script file and execute.

Editor:

```
columns = 1700;
rows = 1; % Set to 1 for one row
bins = round(columns / 40);
rng(now);
data = 50 * rand(rows, columns);

% Create histogram for the single row
histogram(data, bins);
title('Histogram for Row 1');
```

Figure:



2. Write a function file and execute

Editor:

```
function result = myfunction(x, y)
    result = x^2 + y^2;
end
```

Command Window:

```
>> x=2; y=3;

>> a=myfunction(x,y)
>> a = 13;
```

Discussion:

MATLAB is a programming language renowned for its matrix-centric approach to data representation. In this paradigm, information is organized into matrices—structured arrays with rows and columns. MATLAB is equipped with an extensive library of pre-built functions that facilitate an extensive array of mathematical tasks. These tasks range from elementary operations like extracting square roots to more complex endeavors such as solving systems of equations or generating graphical plots.

```
x =

     8

    17

    30


v = [2 0 -1];

y = v*B


y =

    12    -7    10
```

Identity Matrix

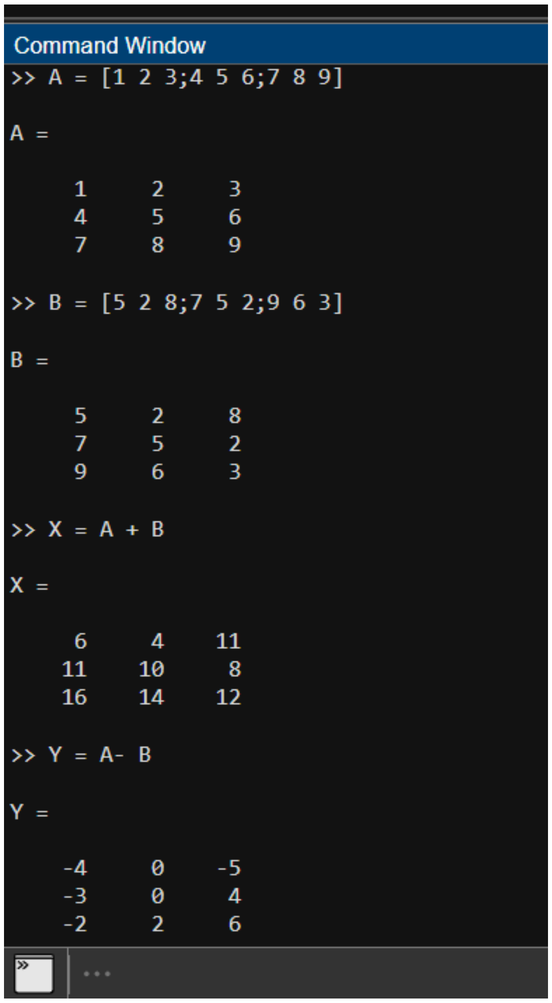
Generally accepted mathematical notation uses the capital letter I to denote identity matrices, matrices of various sizes with ones on the main diagonal and zeros elsewhere. These matrices have the property that AI = A and IA = A whenever the dimensions are compatible. The original version of MATLAB could not use I for this purpose because it did not distinguish between uppercase and lowercase letters and i already served as a subscript and as the complex unit. So an English language pun was introduced. The function

```
eye(m,n)
```

returns an m-by-n rectangular identity matrix and eye(n) returns an n-by-n square identity matrix.

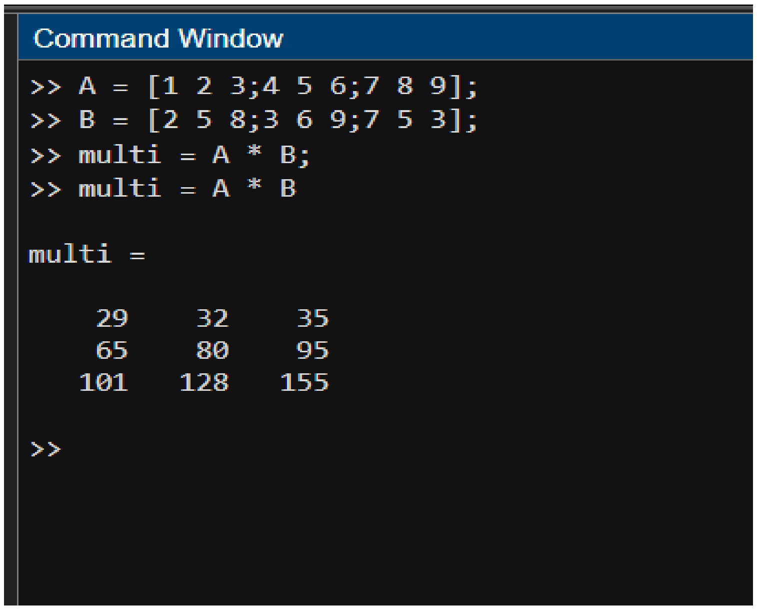
Lab Tasks:

1. Execute addition and subtraction of matrics



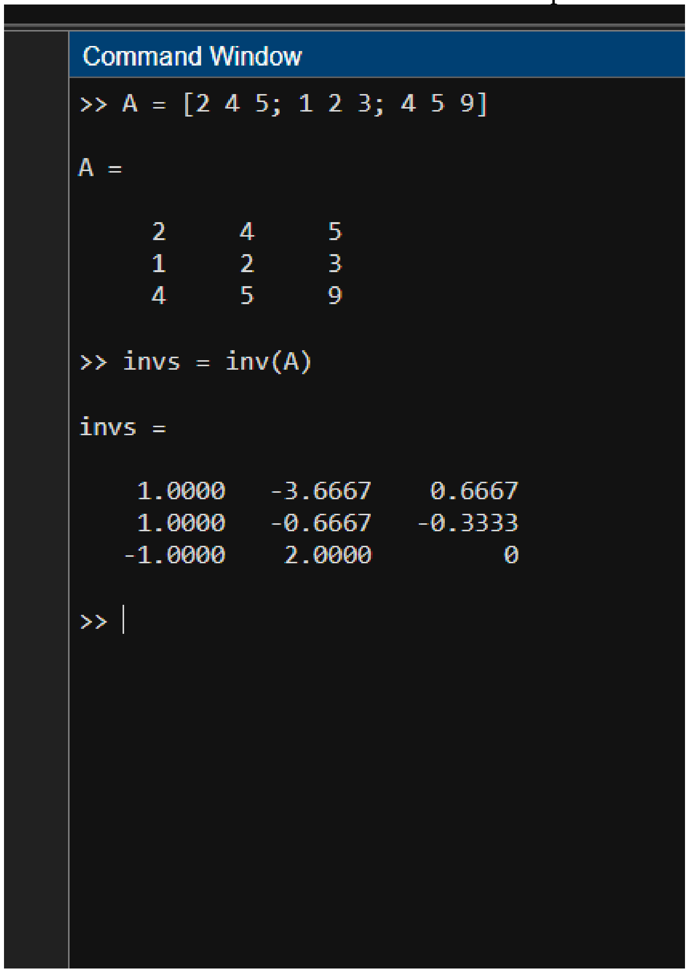
```
>> A = [1 2 3;4 5 6;7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> B = [5 2 8;7 5 2;9 6 3]

B =

     5     2     8
     7     5     2
     9     6     3

>> X = A + B

X =

     6     4    11
    11    10     8
    16    14    12

>> Y = A- B

Y =

    -4     0    -5
    -3     0     4
    -2     2     6
```

2. Execute multiplication of matrics



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

multi =

    29    32    35
    65    80    95
   101   128   155

>>
```

3. Find out INVERSE MATRIX of Your Input A Matrix.



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

A =

     2     4     5
     1     2     3
     4     5     9

>> invs = inv(A)

invs =

    1.0000   -3.6667    0.6667
    1.0000   -0.6667   -0.3333
   -1.0000    2.0000         0

>> 
```

Discussion:

In this lab, we focused on matrix implementation and key operations—addition, subtraction, multiplication, and inversion. The hands-on exercises, executed error-free, demonstrated proficiency in utilizing MATLAB for efficient matrix manipulations. This foundational understanding lays the groundwork for applying matrices in various mathematical and engineering applications.

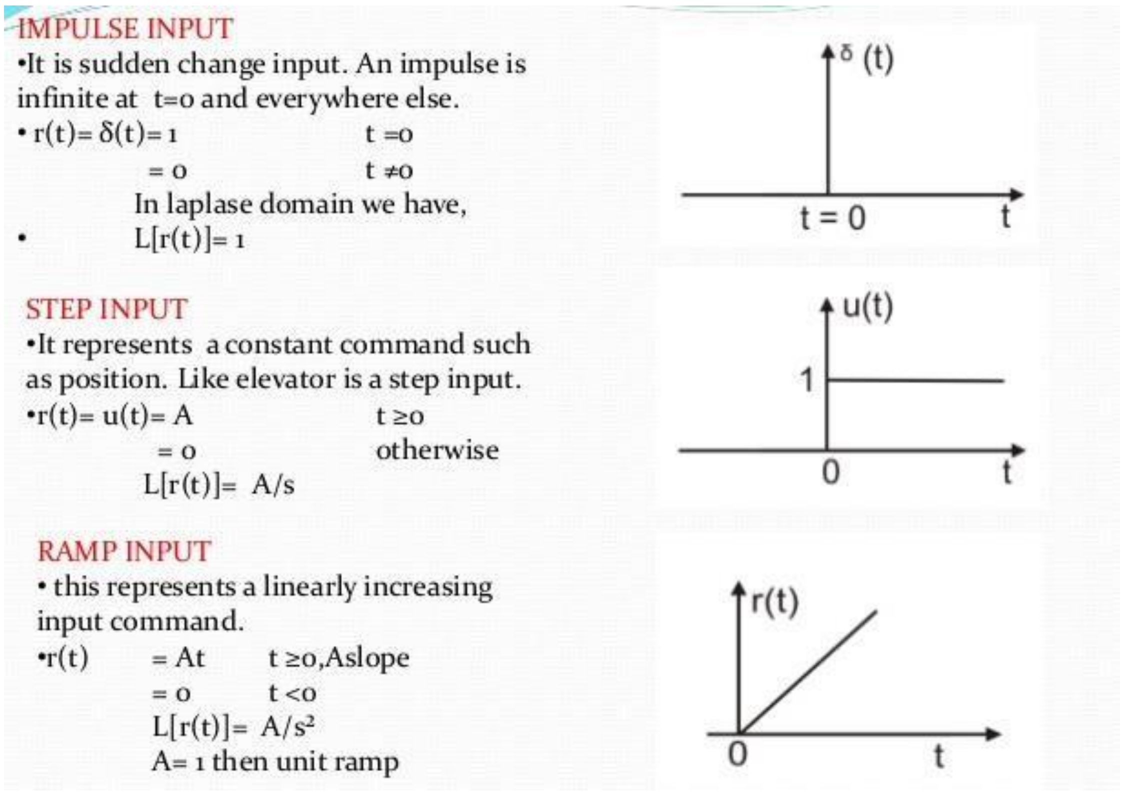
Experiment No: 03

Experiment Name: Plotting Various signals on Matlab

Objective: To write programs that can plot several signals.

Software Requirement: Matlab

Theory: Any signal can be plotted on matlab. Matlab can plot continuous time and discrete time signal. In this experiment some basic signals will analyzed.



IMPULSE INPUT

- It is sudden change input. An impulse is infinite at t=o and everywhere else.
- r(t)= δ(t)= 1   t =o
      = o   t ≠o
  In laplace domain we have,
- L[r(t)]= 1

STEP INPUT

- It represents a constant command such as position. Like elevator is a step input.
- r(t)= u(t)= A   t ≥o
      = o   otherwise
  L[r(t)]= A/s

RAMP INPUT

- this represents a linearly increasing input command.
- r(t)   = At   t ≥o,Aslope
      = o   t <o
  L[r(t)]= A/s²
  A= 1 then unit ramp

Fig 1: Unit step, impulse and ramp signal

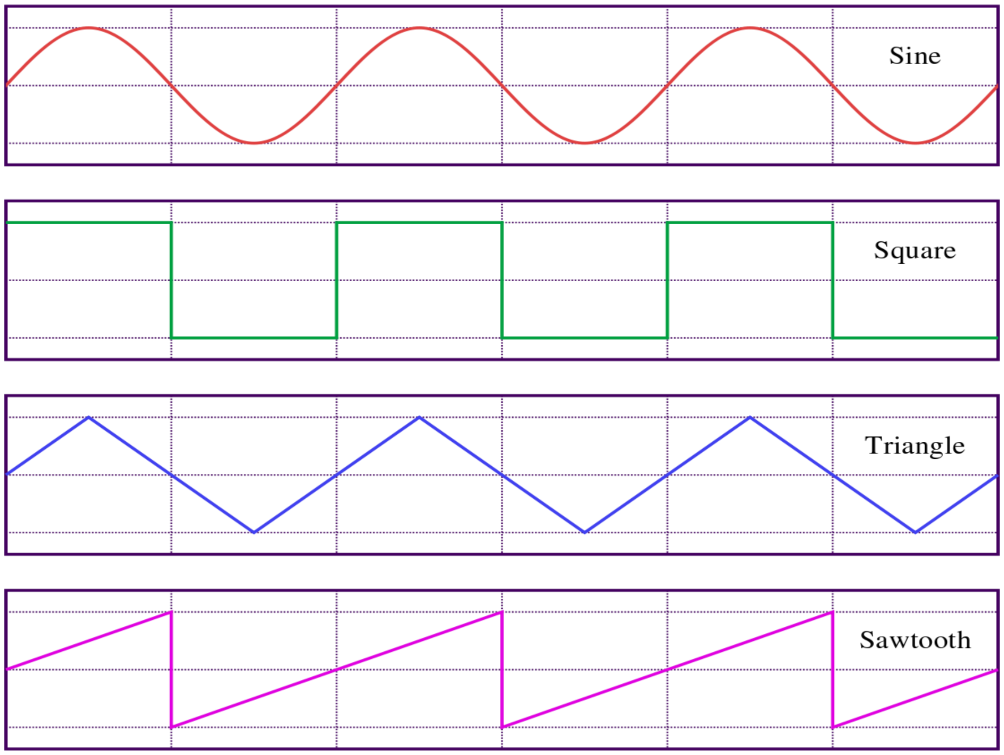


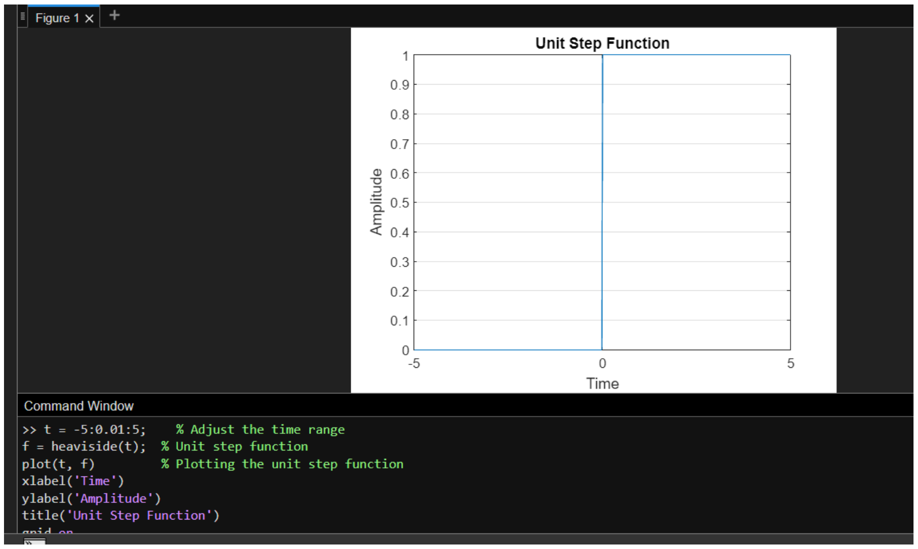
Fig: Some periodic waveform

Matlab Program & Diagrams:

For Unit Step Function:

```
>> t=-10:0.01:10;        % step is small enough to represent continuous-time signal

>> f=heaviside(t);       % the unit step function.

>> plot(t,f)             % plotting unit step function
```

Diagram:

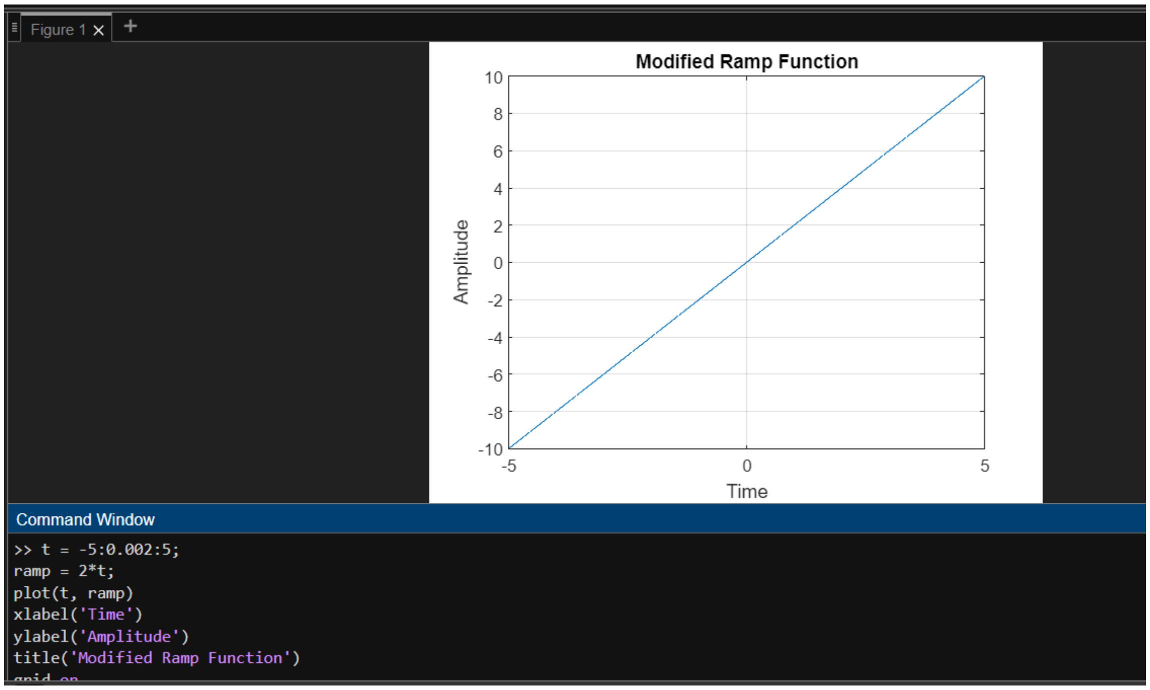


```
>> t = -5:0.01:5;    % Adjust the time range
f = heaviside(t);  % Unit step function
plot(t, f)         % Plotting the unit step function
xlabel('Time')
ylabel('Amplitude')
title('Unit Step Function')
grid on
```

For Ramp Signal:

```
>> t = -10:0.001:10;

>> ramp = t;

>> plot(t,ramp)
```

Diagram:



```
>> t = -5:0.002:5;
ramp = 2*t;
plot(t, ramp)
xlabel('Time')
ylabel('Amplitude')
title('Modified Ramp Function')
grid on
```

For Sine Wave:

% plotting sine wave

t = 0:0.00001:1;

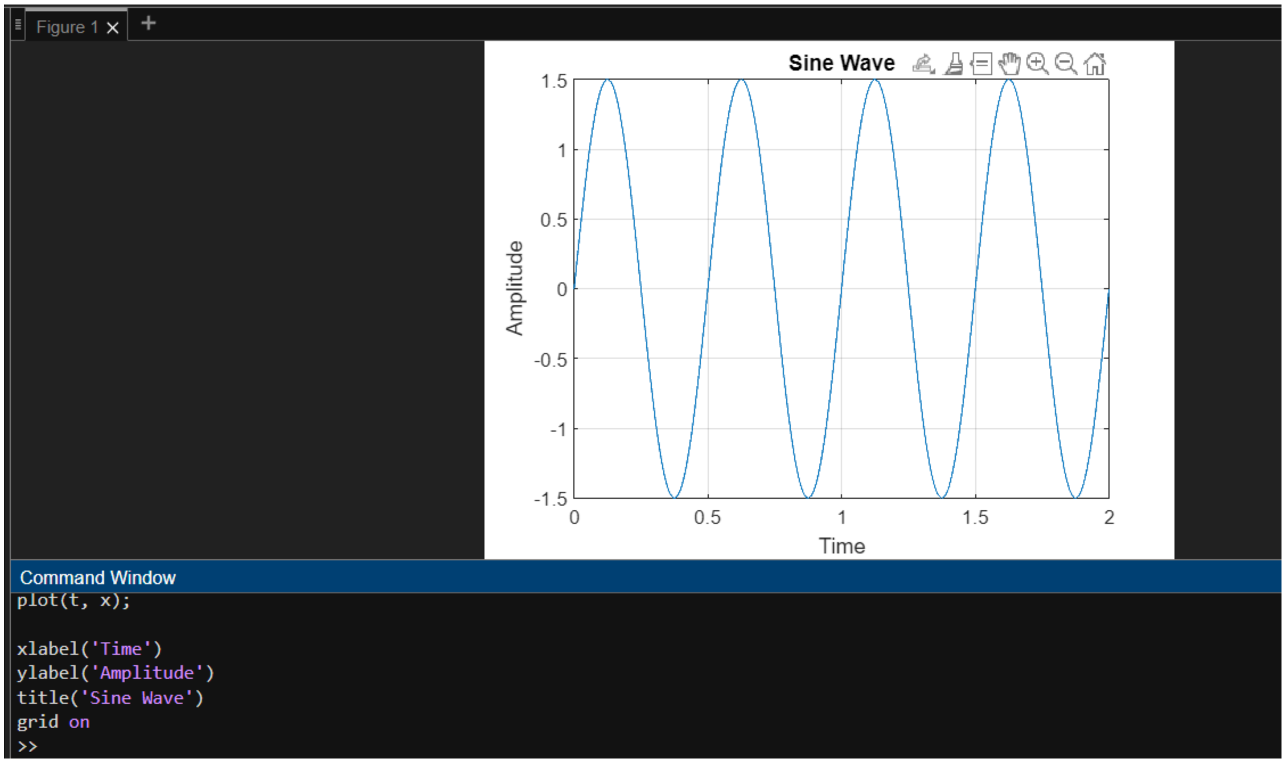
f = 1;

w = 2*pi*f;

x = 2*sin(w*t);

plot(t,x);

Diagram:



```
plot(t, x);

xlabel('Time')
ylabel('Amplitude')
title('Sine Wave')
grid on
>>
```

For square wave:

t = 0:0.001:20;

A = 3;   % amplitude

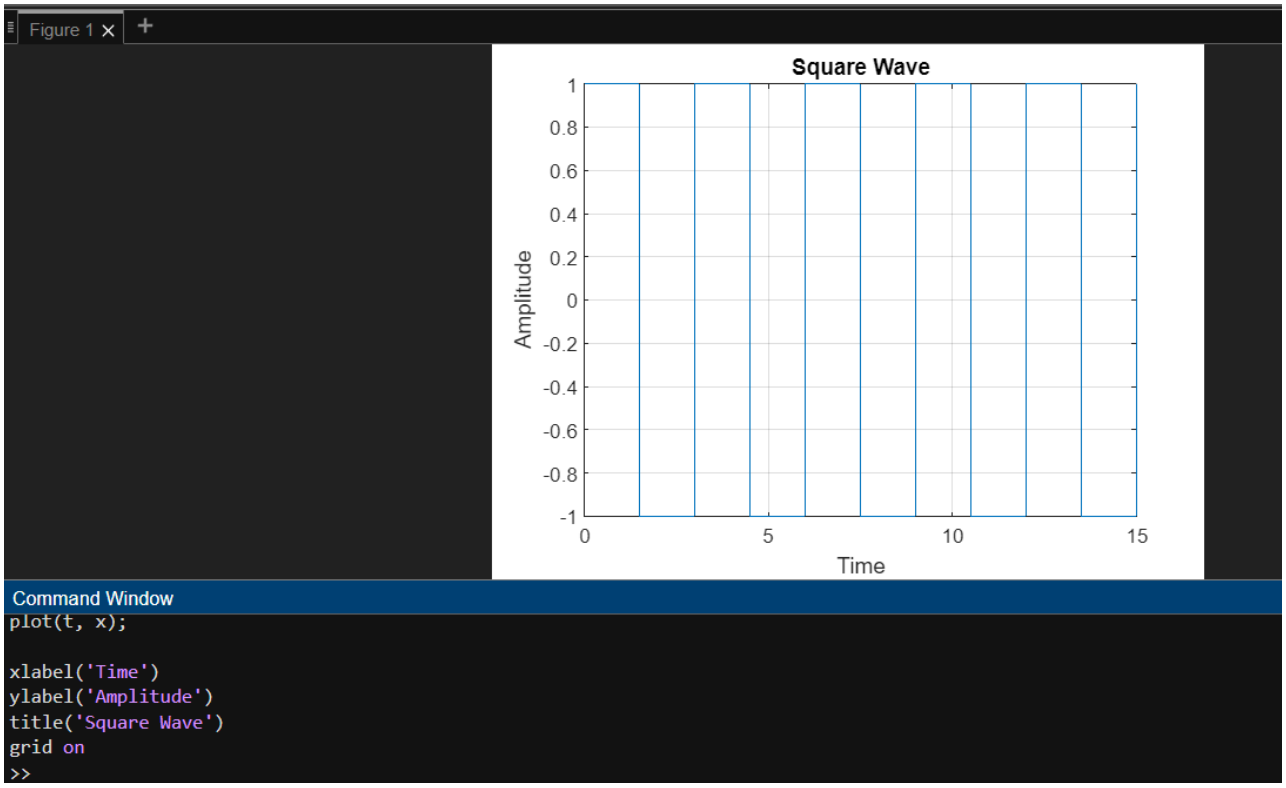
T = 2;   % period

w =(2*3.14)/T; % angular frequency

x = square(w*t);

axis([0 10 -2 2]); % changing the axis

Diagram:



```
plot(t, x);

xlabel('Time')
ylabel('Amplitude')
title('Square Wave')
grid on
>>
```

For sawtooth wave:

t = 0:0.001:20; t = 0:0.001:20;

A = 3;   % amplitude

T = 2;   % period

w =(2*3.14)/T; % angular frequency

x = square(w*t);

A = 3; % amplitude

T = 2; % period

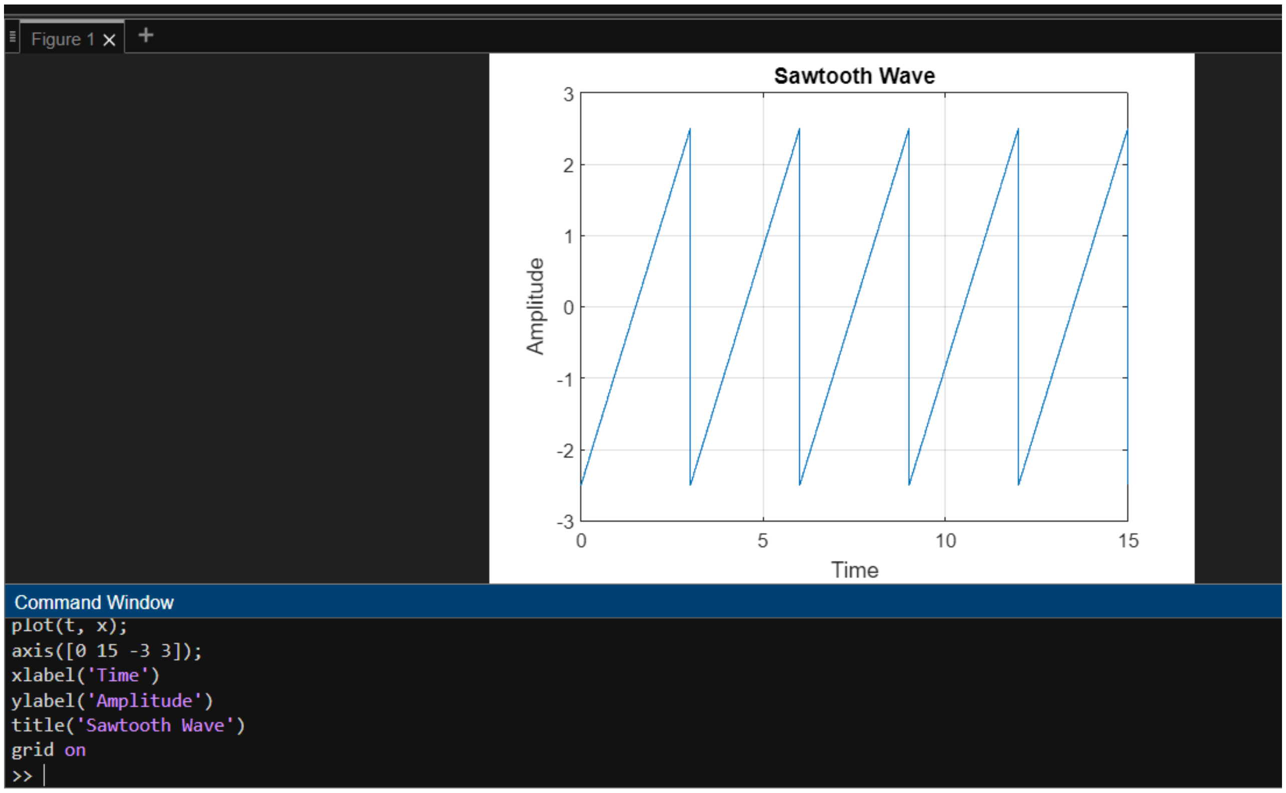
w =(2*3.14)/T; % angular frequency

x =A*sawtooth(w*t);

plot(t,x);

axis([0 10 -4 4]); % changing the axis

Diagram :



```
plot(t, x);
axis([0 15 -3 3]);
xlabel('Time')
ylabel('Amplitude')
title('Sawtooth Wave')
grid on
>>
```

Discussion :

In this lab, I learned to draw fundamental signal functions using MATLAB. The covered functions included the unit step function, ramp function, sine wave, square wave, and sawtooth wave. By manipulating parameters like amplitude, period, and time range, I gained practical insights into signal representation. This hands-on experience with MATLAB enhances my understanding of signal processing concepts and equips me with valuable skills applicable in engineering and scientific contexts

Experiment No: 04

Experiment Name: Operation of signals and Transformation of Independent variable.

Objective: To perform various operation like adding dc level, multiplication, subtraction on dependent variable and shifting, scaling, reversing on the independent variable.

Software Requirement : MATLAB

Theory: Signals have two parameter :1) Amplitude 2) Time. The amplitude of the signals can be modified by addition, multiplication,   division, squaring etc. Suppose y(t) is signal where time is the independent variable. Let us perform some basic operation on y(t).

Suppose, y(t) = 4t

y1(t) = 4t+ 3(Addition)

y2(t) = 4t-2 (Substraction)

y3(t) = 4t*4 (multiplication)

In the above equation, only dependent variable is modified. We can also modify the independent variable.

Again , y(t) = 2sin(wt)

y1(t) = 2sin(wt+2); (shifting of a signal to left)

y2(t) = 2sin(wt-2); time reversal of a signal)

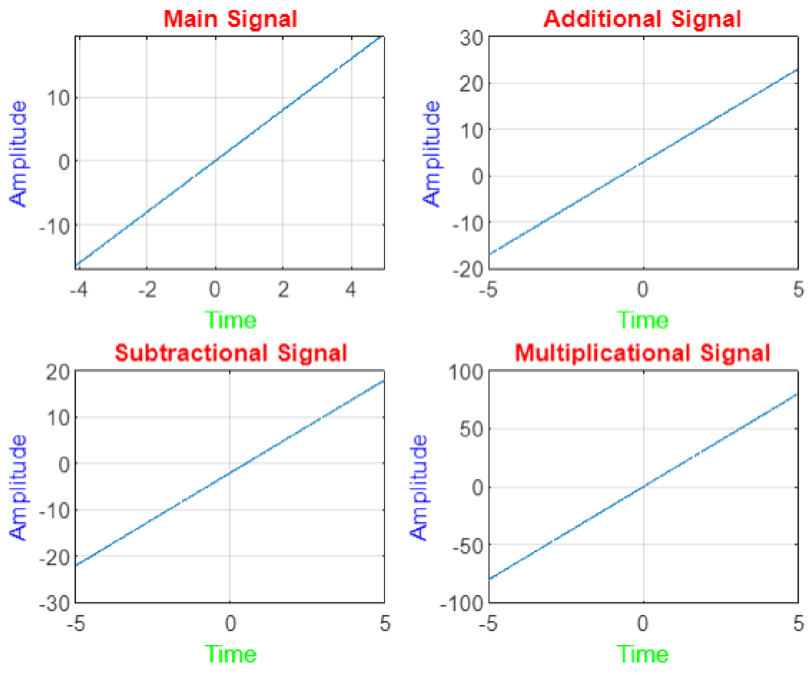
y3(t) = 2sin(wt*2); (scaling of a signal) (

Matlab Program:

- t = -5 : 0.001:5;
- y1 = 4*t;
- y3 = 4*t - 2;
- y4 = 4*t * 4;
- subplot(221);
- title('Main Signal','Color','r');
- xlabel('Time','Color','g');
- ylabel('Amplitude','Color','b');
- grid on;
- plot(t,y1);

- y2 = 4*t + 3;
- subplot(222);
- title('Additional Signal','Color','r')
- xlabel('Time','Color','g');
- ylabel('Amplitude','Color','b');
- grid on;
- plot(t,y2);
- y3 = 4*t -2;
- subplot(223);
- title('Subtractional Signal','Color','r')
- xlabel('Time','Color','g');
- ylabel('Amplitude','Color','b');
- grid on;
- plot(t,y3);
- y4 = 4*t *4;
- subplot(224);
- title('Multiplication Signal','Color','r')
- xlabel('Time','Color','g');
- ylabel('Amplitude','Color','b');
- grid on;
- plot(t,y4);

Diagram :



Operation on dependent variable:

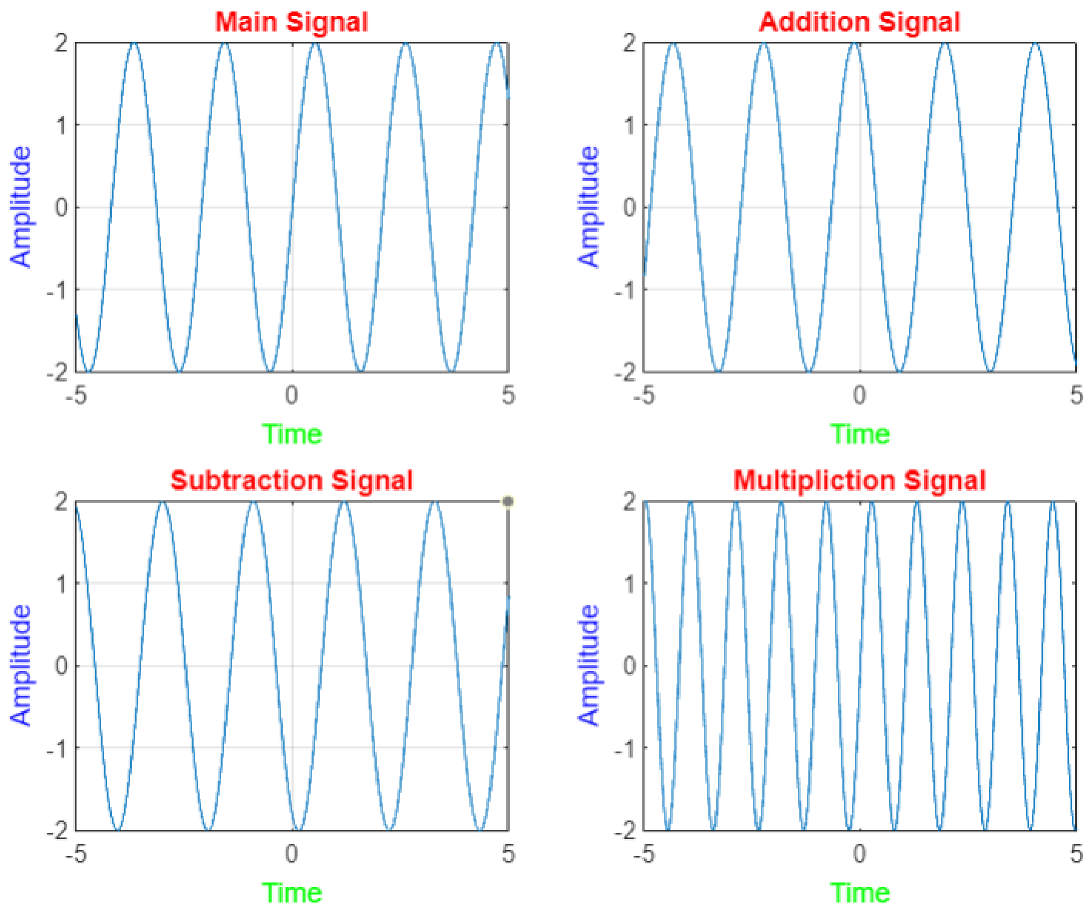
- t = -5 : 0.001:5;
- y1 = 2*sin(w*t);
- subplot(221);
- title('Main Signal','Color','r');
- xlabel('Time','Color','g');
- ylabel('Amplitude','Color','b');
- grid on;
- plot(t,y1);

- y2 = 2*sin(w*t + 2);
- subplot(222);
- title('Addition Signal','Color','r');
- xlabel('Time','Color','g');
- ylabel('Amplitude','Color','b');
- grid on;
- plot(t,y2);

- y3 = 2*sin(w*t - 2);
- subplot(223);
- title('Subtraction Signal','Color','r');
- xlabel('Time','Color','g');
- ylabel('Amplitude','Color','b');
- grid on;
- plot(t,y3);

- y4 = 2*sin(w*t * 2);
- subplot(224);
- title(Multiplication Signal','Color','r');
- xlabel('Time','Color','g');
- ylabel('Amplitude','Color','b');
- grid on;
- plot(t,y4);

Diagram:



Discussion :

In our MATLAB experiment, we found that changing signals had predictable effects on their features, like amplitude and frequency. Basic operations aligned with expectations. Time and frequency transformations, using MATLAB, showed intuitive impacts on signal behavior, emphasizing the practical value of MATLAB for engineers.

<u>**Experiment No:05**</u>

<u>Experiment Name</u>: Observation of convolution in MATLAB

<u>Objective</u>: To observe the convolution of two signals

<u>Software Requirement</u>: MATLAB

<u>Theory</u>: The convolution of two signals is a fundamental operation in signal processing. Mainly, because the output of any linear time-invariant (LTI) system is given by the convolution of its impulse response with the input signal. It means the output of any system due to any input signal can be found if the impulse response is known. The formula of convolution by definition if f1(t) and f2(t):

$$
\begin{aligned}
f_1(t) * f_2(t) &= \int_0^t f_1(\tau) f_2(t-\tau)\,d\tau \\
&= \int_0^t f_2(\tau) f_1(t-\tau)\,d\tau \qquad \dots (1)
\end{aligned}
$$

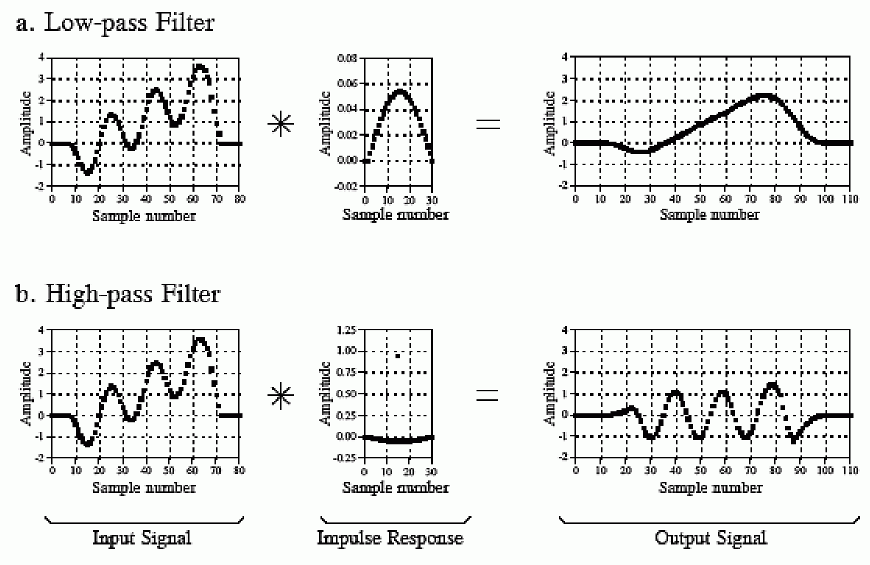


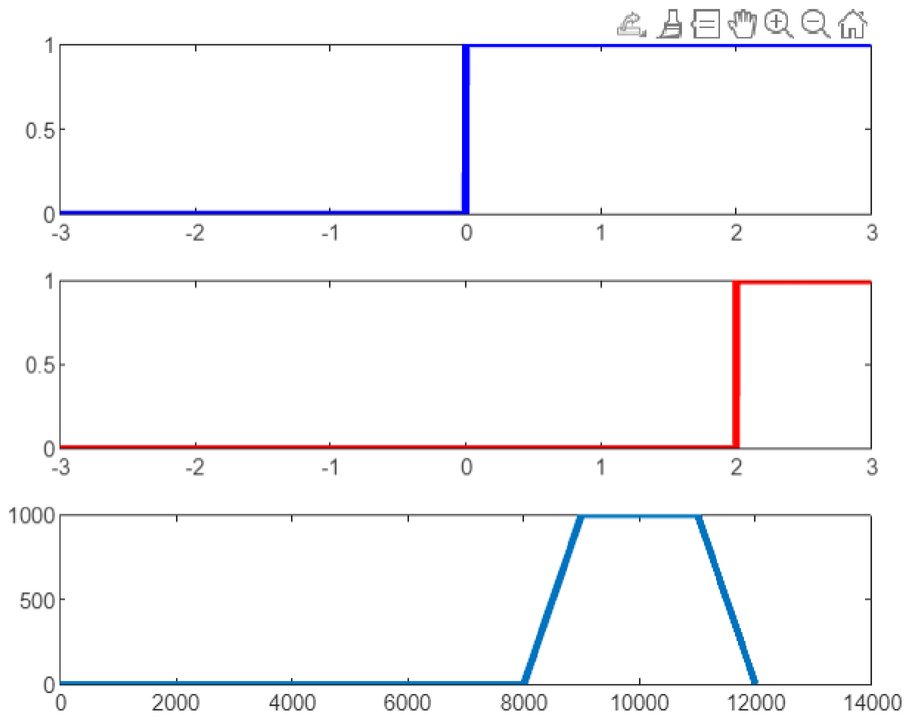
FIGURE 6-3
Examples of low-pass and high-pass filtering using convolution. In this example, the input signal is a few cycles of a sine wave plus a slowly rising ramp. These two components are separated by using properly selected impulse responses.

Figure: Graphical view of convolution.

Matlab Program:

```
t = -5:0.001:5;     % for continuous time signal

impulse_resp = heaviside(t);

subplot(3,1,1);

plot(t, impulse_resp, 'b-', 'Linewidth', 3);

input = heaviside(t-1);

subplot(3,1,2);

plot(t, input, 'r-', 'Linewidth', 3);

z = conv(impulse_resp, input);

subplot(3,1,3);

plot(z, 'Linewidth', 3);
```

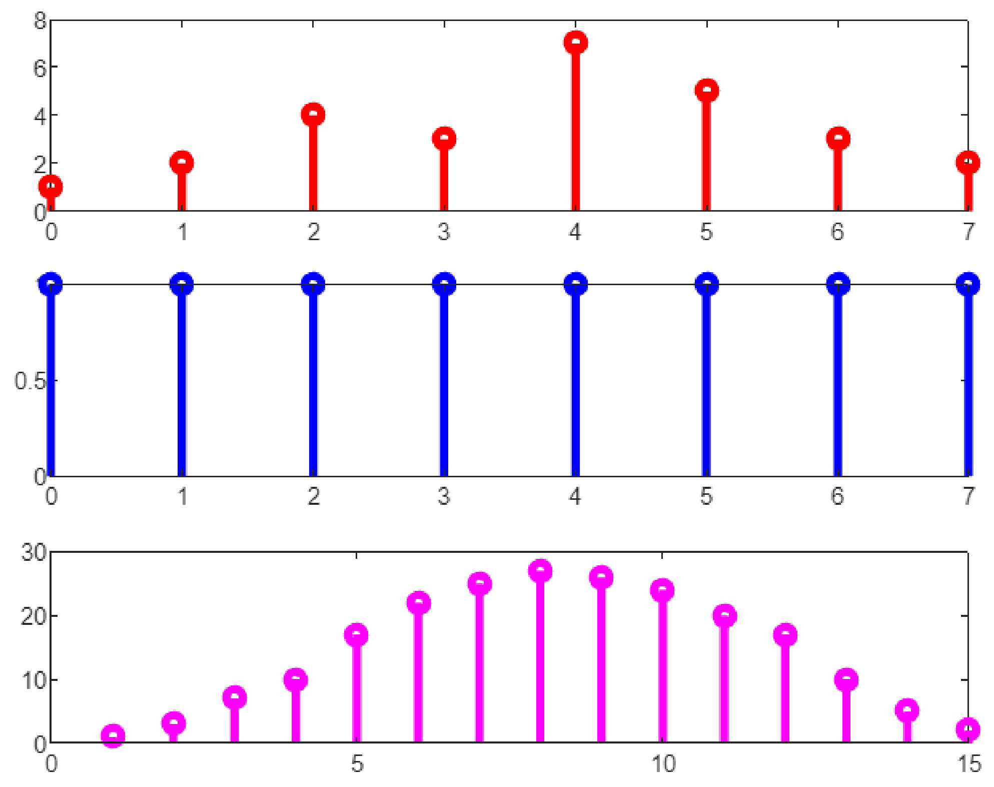
Diagram:



Matlab Program:

```
n = 0:8;
xn = [1 3 4 8 7 4 3 2];
subplot(3,1,1);
stem(n,xn,'r-','Linewidth',4);
hn = [ 1 1 1 1 1 1 1 1 ];
subplot(3,1,3);
stem(n,hn,'b-', 'Linewidth',4);
yn = conv(xn,hn);
subplot(3.5,1,3.5);
stem(yn,'m-','Linewidth',3);
```

Diagram:



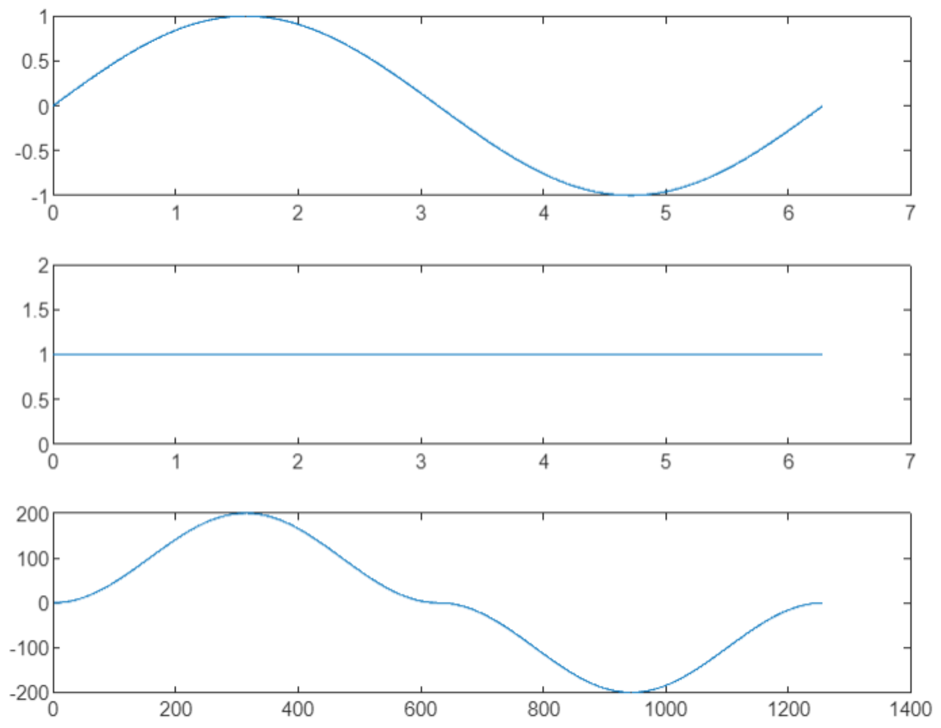
Home work:

convolution of two sign signal.

Matlab Program:

```
t = 0:0.01:4*pi;

x = sin(t);

input = heaviside(t + 2);

subplot(3,1,1);

plot(t, x);

subplot(3,1,2);

plot(t, input);

y = conv(x, input);

subplot(3,1,3);

plot(y);
```

Diagram :



Discussion:

In MATLAB, the observation of convolution involves applying the convolution operation to two signals or functions. The conv function is commonly used for this purpose, and it computes the convolution of two sequences efficiently. It is crucial to note that convolution in MATLAB is implemented using different methods, such as the direct method or the FFT-based method, which can impact computational efficiency. Observing the convolution process helps understand how signals interact and how convolution can be applied in various signal processing applications. MATLAB provides a versatile platform for visualizing and analyzing convolution, making it an essential tool in signal processing and communication system design.