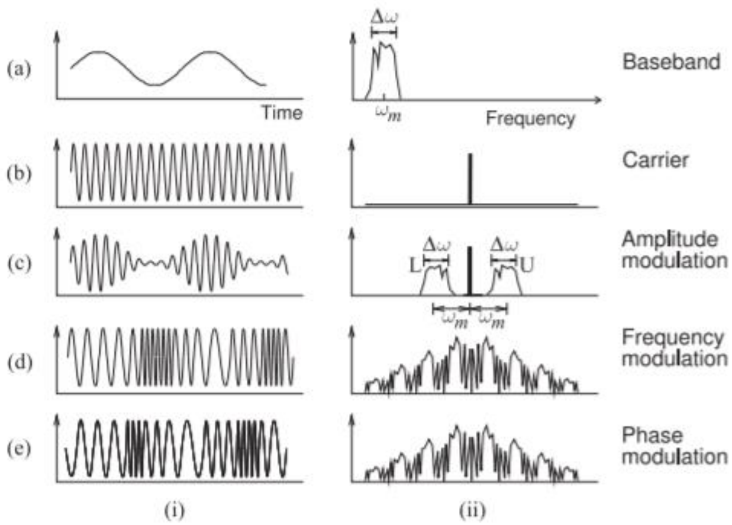
Name	Abdul Mateen, Basit, Arslan
Reg. #	2020-EE-399, 401, 409
Marks	

Lab # 1:Analog Communication Basics

1. Analog Communication:

- ✓ Analog signals are continuous; they vary in **amplitude**, **frequency**, and/or **phase**. Digital signals are discontinuous; they are either high (logic 1) or low (logic 0).
- ✓ Signals are transmitted by conduction through a wire or radiated through space.
- ✓ Message signals (information) have low frequencies from 20 Hz to 20 kHz. Carrier signals have frequencies from 10 kHz to 1000GHz.
- ✓ Modulation is where the message signal changes a characteristic (**amplitude, frequency or phase**) of the carrier signal so that the message can be transmitted at the higher carrier frequency.



- ✓ In the receiver, demodulation is the process of recovering the message signal from the modulated carrier signal.

2. MATLAB:

MATLAB is a powerful tool that is utilized by the engineers and others professionals in development and testing of various projects. It is versatile software, with the help of which you can solve and develop any sort of engineering problem. The name MATLAB stands for **MATRIX LABORAORY**. All the work done in MATLAB is basically in the form of matrices. Scalars are referred as 1-to-1 matrix and vectors are matrices having more than 1 row and column. MATLAB is programmable and have the same logical, relational, conditional and loop structures as in other programming languages, such as C, Java etc. It's very easy to use MATLAB, all we need is to practice it and become a friend of it.

Review of basics of MATLAB:

a) Go to the start button, then programs, MATLAB and then start MATLAB. It is preferred that you have MATLAB2015a. You can then start MATLAB by double clicking on its icon on Desktop, if there is any.

b) The Prompt:

\>>

The operator shows above is the prompt in MATLAB. MATLAB is interactive language like C, Java etc. We can write the commands over here.

Task 1: Help in MATLAB

In order to use the built-in help of the MATLAB we use the **help** keyword. Write it on the prompt and see the output.

```
>> help sin
```

```
>> Task_1_help_sin
 sin    Sine of argument in radians.
    sin(X) is the sine of the elements of X.

    See also asin, sind.

    Reference page for sin
    Other functions named sin
```

Also try

```
>> lookfor sin
BioIndexedFile              - class allows random read access to text files using an index file.
loopswitch                  - Create switch for opening and closing feedback loops.
mbcinline                   - replacement version of inline using anonymous functions
cgslblock                   - Constructor for calibration Generation Simulink block parsing manager
xregaxesinput               - Constructor for the axes input object for a ListCtrl
ExhaustiveSearcher          - Neighbor search object using exhaustive search.
KDTreeSearcher              - Neighbor search object using a kd-tree.
tscollection                - Create a tscollection object using time or time series objects.
detrend                     - Remove a linear trend from a vector, usually for FFT processing.
cell2mat                    - Convert the contents of a cell array into a single matrix.
isfloat                     - True for floating point arrays, both single and double.
isinteger                   - True for arrays of integer data type.
isinterface                 - true for COM Interfaces.
single                      - Convert to single precision.
superiorfloat               - return 'double' or 'single' based on the superior input.
acos                        - Inverse cosine, result in radians.
acosd                       - Inverse cosine, result in degrees.
acosh                       - Inverse hyperbolic cosine.
asin                        - Inverse sine, result in radians.
```

Task 2: Vectors

Vectors are also called arrays in MATLAB. Vectors are declared in the following format.

```
>> X = [1 2 3 4]
```

```
X =

     1     2     3     4
```

```
>> Y = [2 5 8 9]
```

```
Y =

     2     5     8     9
```

Try these two instructions in MATLAB and see the result

>> length (X)

```
>> length (X)

ans =

     4
```

>> size (X)

```
>> size (X)

ans =

     1     4
```

What is the difference between these two?

Ans:

Length(x) is used to find the total number of elements use in vector. And size(x) is used to represent the total size of vector(first value and last value in the vector.

Try these instructions and see the results.

Task#3;

>> X.*Y

```
>> X.*Y

ans =

     2    10    24    36
```

>> X.^Y

```
>> X.^Y

ans =

           1          32        6561      262144
```

>> X+Y

```
>> X+Y

ans =

     3     7    11    13
```

>> X-Y

```
>> X-Y

ans =

    -1    -3    -5    -5
```

>> X./Y

```
>> X./Y

ans =

    0.5000    0.4000    0.3750    0.4444
```

>> X′

```
>> X'

ans =

     1
     2
     3
     4
```

Also try some instructions for this like and notice the outputs in each case.

Task#4;

>> ones (1,4)

```
ans =

     1     1     1     1
```

>> ones (2,4)

```
ans =

     1     1     1     1
     1     1     1     1
```

>> ones (4,1)

```
ans =

     1
     1
     1
     1
```

>> zeros (1,4)

```
ans =

     0     0     0     0
```

>> zeros (2,4)

```
ans =

     0     0     0     0
     0     0     0     0
```

There is an important operator, the colon operator (**:**), it is very important operator and frequently used during these labs. Try this one.

Task#5;

\>> X = [0:0.1:1]

```
X =

  Columns 1 through 10

        0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000

  Column 11

    1.0000
```

Notice the result. And now type this

\>> length (X)

```
>> length (X)

ans =

    11
```

\>> size (X)

```
>> size (X)

ans =

     1    11
```

What did the first and second number represent in the output of last instruction?

Ans:

To get all values from one point to another point, we use colon(:).

Now try this one.

Task#6;

\>> A= [ones(1,3), [2:2:10], zeros(1,3)] **What is the length and size of this?**

\>> Length

```
>> length(A)

ans =

    11
```

\>>Size

```
>> size (A)

ans =

     1    11
```

Task#7;

MATRICES

Try this and see the output.

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

```
A =

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

>> B = [1,2,3;4,5,6;7,8,9]

```
B =

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

Is there any difference between the two?
Ans:

No, there is no difference between the two matrices A and B.

Task#8;

Try to implement 2-to-3 matrix and 3-to-2 matrix.
2-to-3 matrix

```
>> B = [1,2,3;4,5,6]

B =

     1     2     3
     4     5     6
```

3-to-2 matrix

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

A =

     1     2
     4     6
     7     9
```

Task#9;

PLOTTING

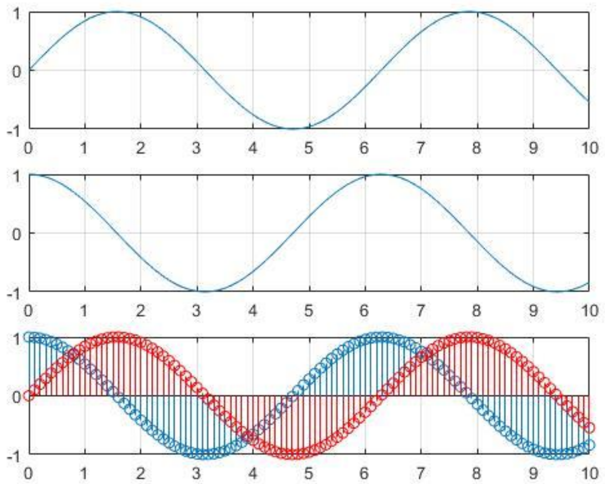
Plotting is very important as we have to deal with various type of waves and we have to view them as well.

Try these and have a look on the results.

Code:

```
x = [0:0.1:10];
y = sin (x);
z = cos (x);
subplot (3,1,1);
plot (x,y);
grid on;
subplot (3,1,2);
plot (x,z);
grid on; hold on;
subplot (3,1,3);
stem (x,z);
grid on;
hold on;
subplot (3,1,3);
stem (x,y,'r');
```

Output:



M-FILES

MATLAB can execute a sequence of statements stored in disk files. Such files are called Mfiles because they must have the file type '**.m**'. Lot of our work will be done with creation of m-files.

There are two types of m-files: Script and function files.

Script Files

We can use script files in order to write long programs such as one on the previous page. A script file may contain any command that can be entered on the prompt. Script files can have any name but they should be saved with ".m" extension. In order to excurse an m-file from the prompt, just type its name on the prompt.

You can make an m-file by typing **edit** on the prompt or by clicking on the file then new and m-file. See an example of m-file. Write it and see the results.

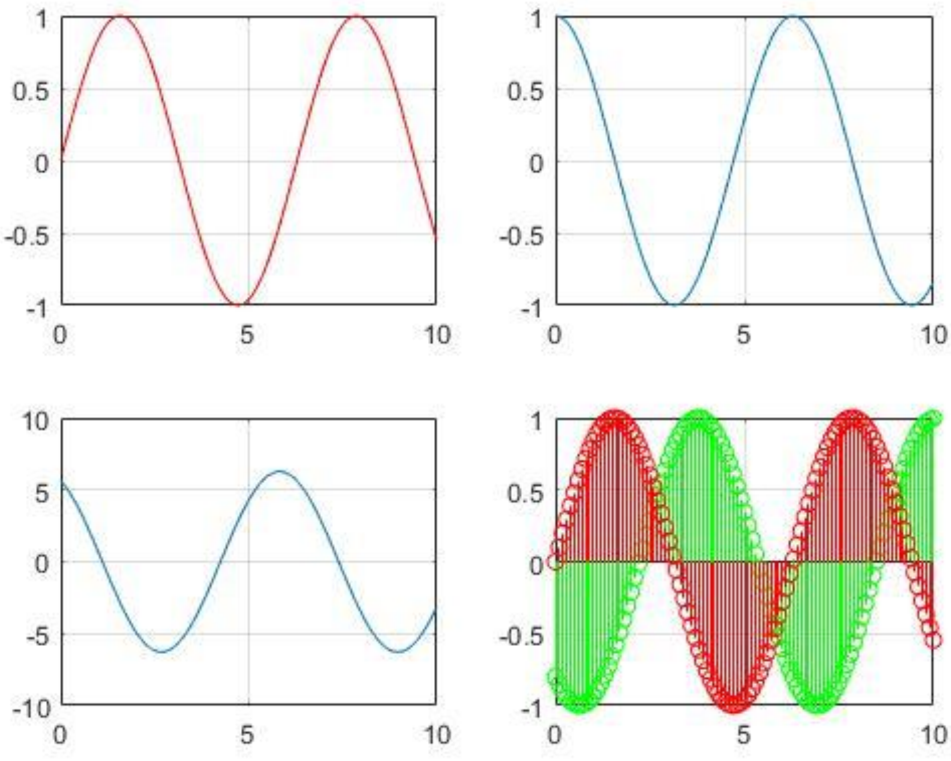
Task#10;

% This is comment
% A comment begins with a percent symbol
% The text written in the comments is ignored by the MATLAB
% comments in your m-files.

Code

```
clear;
clc;
x  = [0:0.1:10];
y = sin (x);
subplot (2,2,1);
plot (x,y,'r');
grid on;
z = cos (x);
subplot (2,2,2);
plot (x,z);
grid on;
w  = 90;
yy = 2*pi*sin (x+w)
subplot (2,2,3);
plot (x,yy);
grid on;
zz = sin (x+2*w); subplot (2,2,4);
stem (x,zz,'g');
hold on;
stem (x,y,'r');
grid on;
```

Graph



Function Files

MATLAB have many built-in functions including trigonometry, logarithm, calculus and hyperbolic functions etc. In addition we can define our own functions and we can use builtin functions in our functions files as well. The function files should be started with the function definition and should be saved with the name of function. The general format of the function file is

Function [output_variables] = function name (input_variables)

See the following example and implement it.

% this is a function file

% this function computes the factorial of a number function [y] = my_func (x)

Task#11;

y = factorial (x)

```
>> x=4

x =

     4

>> y = factorial(x)

y =

    24
```

Generation of Signals

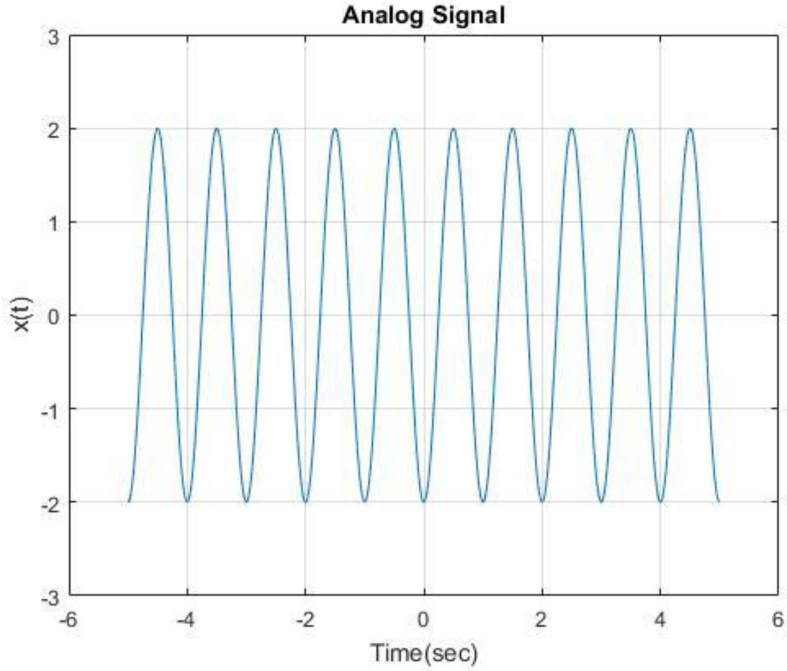
Signals are represented mathematically as a function of one or more independent variables. We will generally refer to the independent variable as time. Therefore, we can say a signal is a function of time. Write these instructions in m-file as execute to see the result.

Task#12;

Sinusoidal Sequence:

```
 % Example 2.1
% Generation of sinusoidal signals
% 2sin( 2??-?/2)
t=[ -5:0.01:5]; x=2*sin((2*pi*t)-(pi/2));
plot(t,x)
grid on;
axis([-6 6 -3 3])
ylabel ('x(t)')
xlabel ('Time(sec)')
title (' Analog Signal ')
```

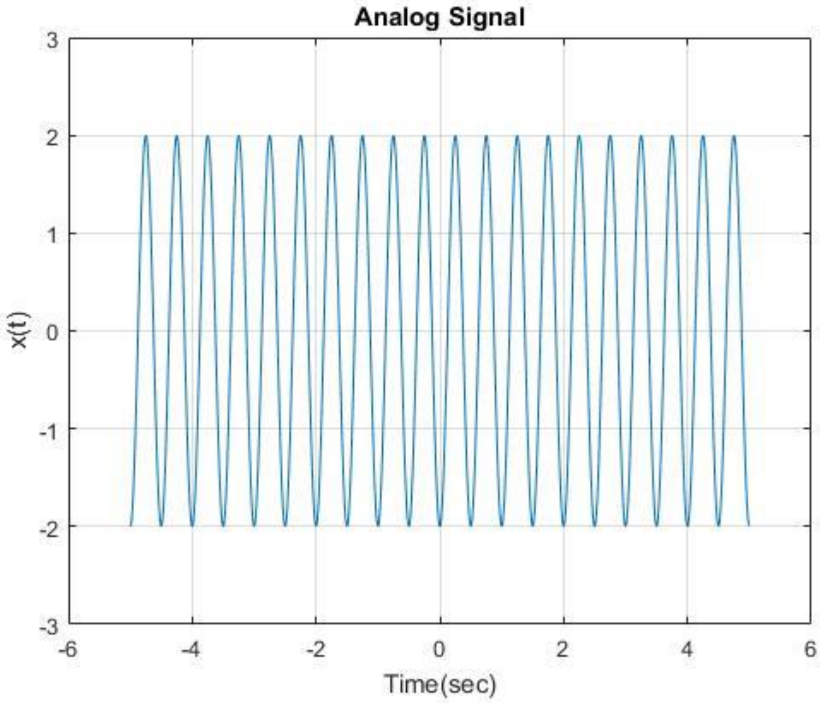
Graph:



See the output, change the phase shift value and observe the differences.

```
 % Example 2.1
% Generation of sinusoidal signals
% 2sin( 2??-?/2)
t=[ -5:0.01:5]; x=2*sin((4*pi*t)-(pi/2));
plot(t,x)
grid on;
axis([-6 6 -3 3])
ylabel ('x(t)')
xlabel ('Time(sec)')
title (' Analog Signal ')
```

Graph



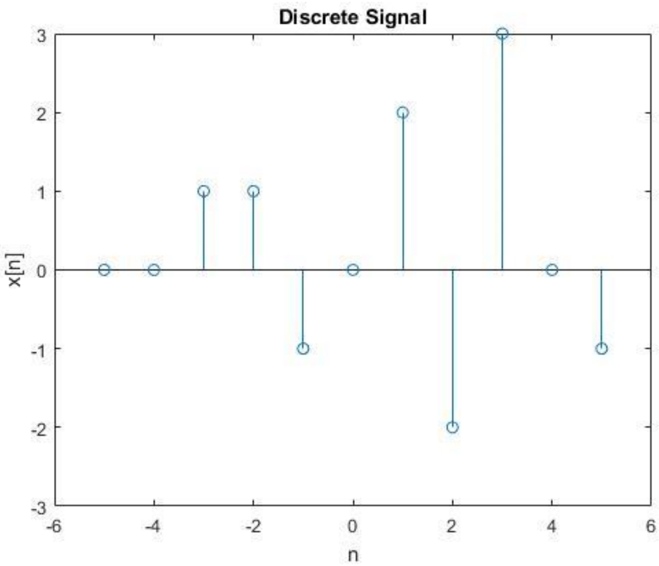
Task#13;

Discrete Time Sequences:

See the example below:

```
% Generation of discrete time signals
n = [ -5:5];
x = [0 0 1 1 -1 0 2 -2 3 0 -1];
stem (n,x);
axis ([-6 6 -3 3]);
xlabel ('n');
ylabel ('x[n]');
title  ('Discrete Signal');
```

Graph:



Task#14;

Unit Step Sequence:

It is defined as

u(n) = 1 n ≥ 0

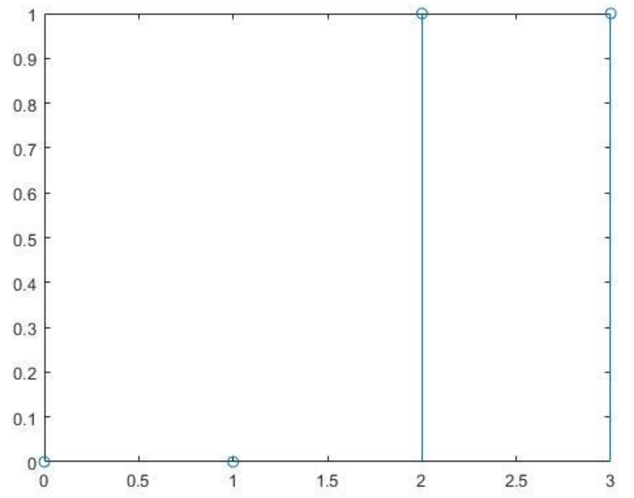
0 n ≤ 0

The MATLAB code for stem sequence function is given below:

Code:

```
function [x,n] = stepseq(n0,n1,n2)
n0 = 2;
n1 = 0;
n2 = 3;
% Generates x(n) = u(n-n0); n1 <= n, n0<=n2
% [x,n] = stepseq(n0,n1,n2)
if ((n0 < n1) || (n0 > n2) || (n1 > n2))
error('arguments must satisfy n1 <= n0 <= n2')
end
n = [n1:n2];
% x = [zeros(1,(n0-n1)),ones(1,(n2-n0+1))];
x = [(n-n0) >= 0];
stem (n,x);
```

Graph:



Random Sequence:

Many practical sequences cannot be described by the mathematical expressions like above, these are called random sequences. In MATLAB two types of random sequences are available. See the code below:

```
>> rand (1,N)
>> randn (1,N)
```

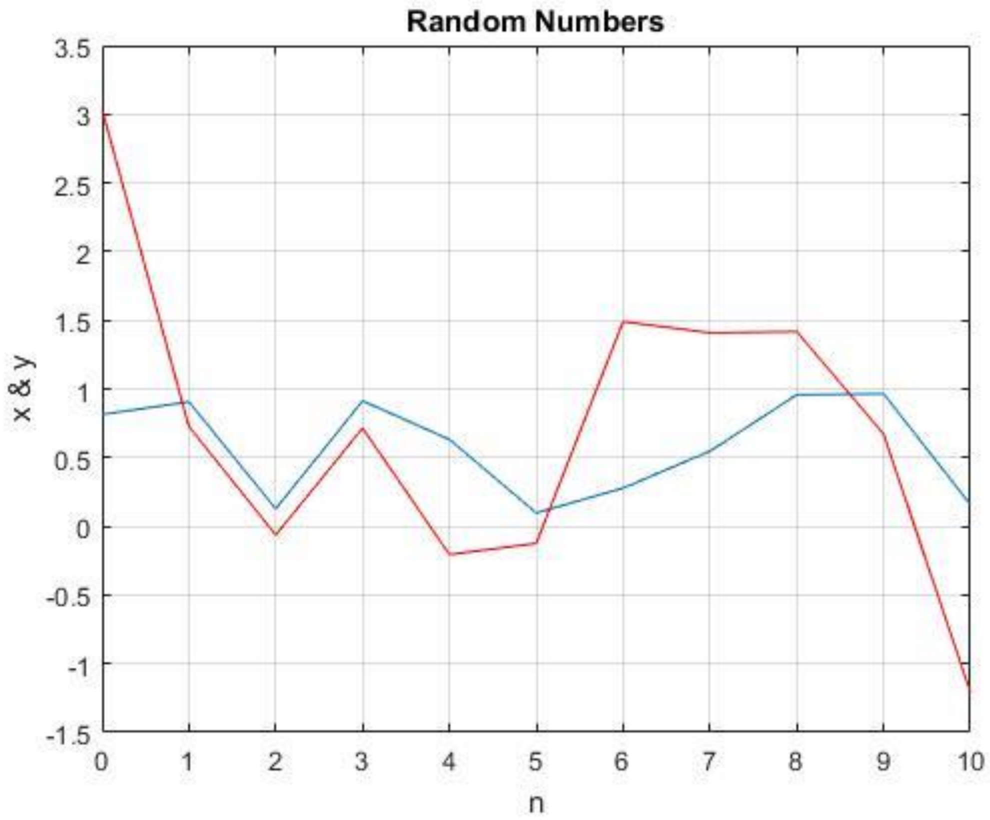
The above instruction generates a length **N** random sequence whose elements are uniformly distributed between [0,1]. And the last instruction, **randn** generates a length **N** Gaussian random sequence with mean 0 and variance 1. Plot these sequences.

Task#15;

Code:

```
%Generation of random sequence
n = [0:10];
x = rand (1, length (n));
y = randn (1, length (n));
plot (n,x) ;
grid on;
hold on;
plot(n,y,'r');
ylabel ('x & y')
xlabel ('n')
title ('Random Numbers')
```

Graph:



Periodic Sequences:

A sequence is periodic if it repeats itself after equal interval of time. The smallest interval is called the fundamental period. Implement code given below and see the periodicity.

Task#16;

Code”

```
% Generation of periodic sequences

n = [0:4];
x = [1 1 2 -1 0];
subplot (2,1,1);
stem (n,x);
grid on;
axis ([0 14 -1 2]);
xlabel ('n');
ylabel ('x(n)');
title ('Figure 2.4(a)');
xtilde = [x,x,x];
length_xtilde = length (xtilde);
n_new = [0:length_xtilde-1];
subplot (2,1,2);
stem (n_new,xtilde,'r');
grid on;
xlabel ('n');
ylabel ('perodic x(n)');
title ('Periodic Signal');
```

Graph:

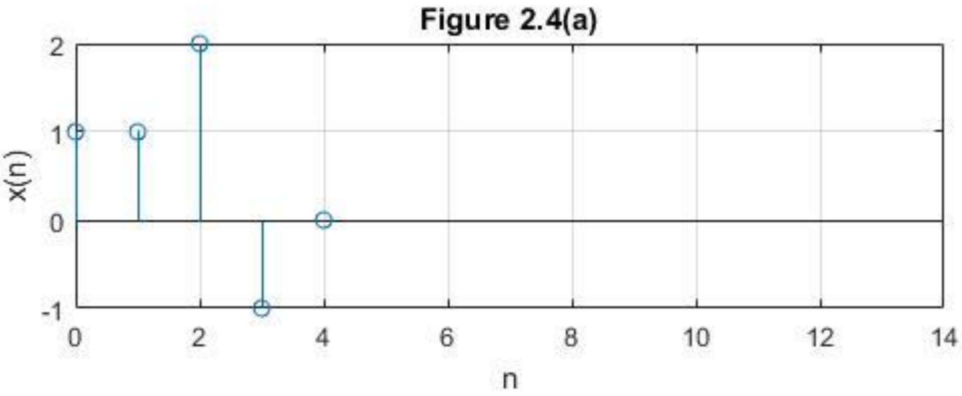
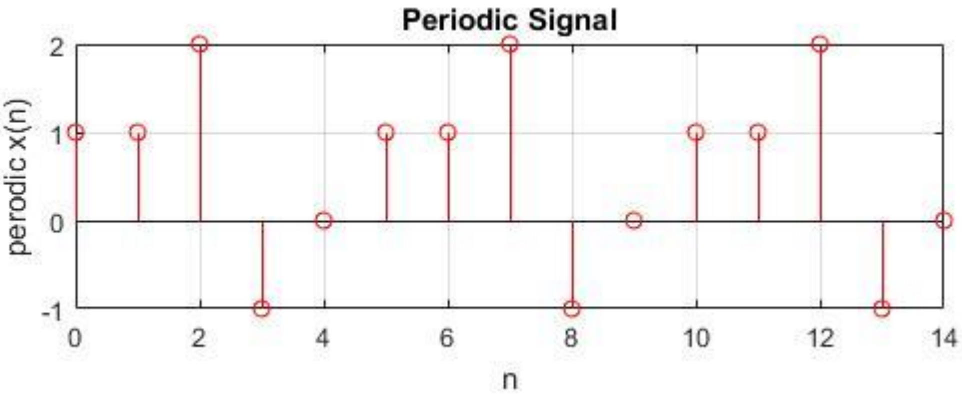


Figure 2.4(a)
x(n)
n
2
1
0
-1
0
2
4
6
8
10
12
14



Periodic Signal
perodic x(n)
n
2
1
0
-1
0
2
4
6
8
10
12
14