Name: Hemal Sharma

ID: 2221855 **Sec: 02**

Course Title : Digital Signal Processing LAB
Course Code : EEE 321L / ETE 324L (New); ECR 305L (Old)
Instructor : Dr. Kh Shahriya Zaman
Experiment No. **: 04**
Experiment Name **: Study on signal decomposition, down-sampling, folding and shifting.**

Objectives:

1. To make a MATLAB function so that signal decomposition into even and odd symmetry can be done.
2. To make a MATLAB function so that down-sampling of a signal can be done.

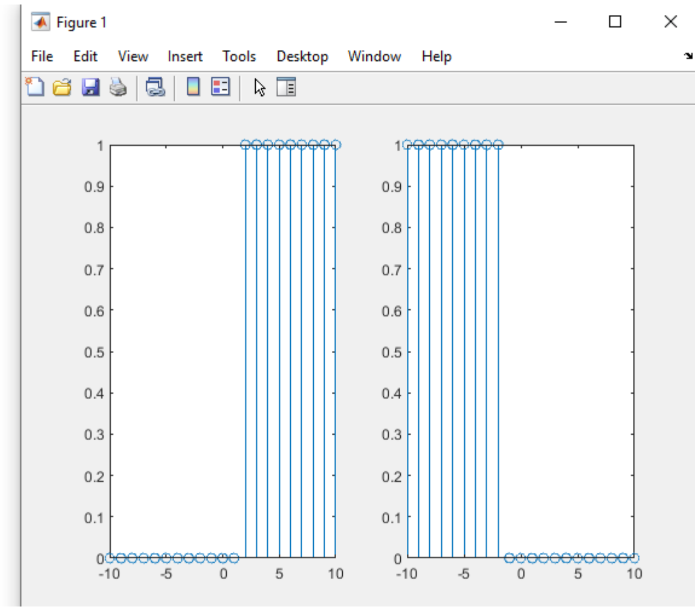
Theory

1. **Even & Odd Function:**
A function **f** is even if the graph of **f** is symmetric with respect to the **y**-axis. Algebraically, **f** is even if and only if **f(-x) = f(x)** for all **x** in the domain of **f**.
A function **f** is odd if the graph of **f** is symmetric with respect to the origin. Algebraically, **f** is odd if and only if **f(-x) = -f(x)** for all **x** in the domain of **f**. In other words, a non-imaginary valued sequence $x_e(n)$ is called even (symmetric) if
$$x_e(-n) = x_e(n)$$
Similarly, a non-imaginary valued sequence $x_e(n)$ is called odd (antisymmetric) if
$$x_o(-n) = -x_o(n)$$
Then any arbitrary real-valued sequence x(n) can be decomposed into the even and odd components
$$x(n) = x_e(n) + x_o(n)$$
where,
$$x_e(n) = \frac{1}{2}[x(n) + x(-n)]$$
$$x_o(n) = \frac{1}{2}[x(n) - x(-n)]$$
2. **Up sampling:**
To enhance the sampling rate, up sampling is the act of inserting zero-valued samples between original samples. This is sometimes referred to as "zero-stuffing."
3. **Down sampling:**
Down sampling is the process of lowering a signal's sampling rate in signal processing. This is typically done to minimize the data rate or data size.
4. **Fliplr:**
B = fliplr(A) returns A with its columns reversed from left to right. Fliplr(A) produces a vector of the same length with the order of its elements reversed if A is a row vector. Fliplr(A) just returns A If A is a column vector.

Lab work:

1. Use ***fliplr*** function to generate the signal **x[-n],** if **x[n] = u(n-2),** for **-10≤ n ≤ 10**.

```
%Hemal Sharma
%ID: 2221855
%x[-n], if x[n] = u(n-2), for -10≤ n ≤ 10.
n = -10:10;
x = stepseq(2,-10,10);
subplot(1,2,1)
stem(n,x)
subplot(1,2,2)
x2 = fliplr(x);
stem(n,x2)
```



2. Develop a **MATLAB** function named **"evenodd"** that has a form;

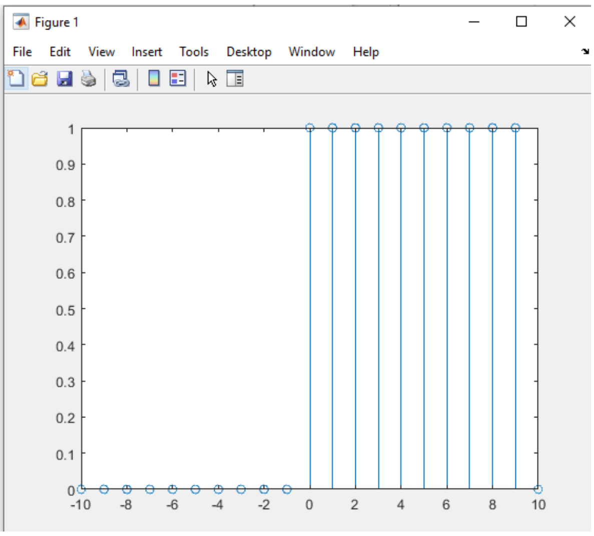
function [xe,xo,m] = evenodd(x,n)

```
%Hemal Sharma
%ID: 2221855
function [xe,xo,m] = evenodd(x,n)
    m=n;
    xe = (1/2)*(x+fliplr(x));
    xo = (1/2)*(x-fliplr(x));

end
```

3. (i) Generate **x[n] = u(n) – u(n-10)** where **-10 ≤ n ≤ 10.**

```
%Hemal Sharma
%ID: 2221855
% Generate x[n] = u(n) - u(n-10) where -10 ≤ n ≤ 10.
n = -10:10;
x = stepseq(0,-10,10)- stepseq(10,-10,10);
stem(n,x)
```



(ii) Decompose **x[n]** to generate **xe[n] and xo[n].**

```
%Hemal Sharma
%ID: 2221855
n = -10:10;
x = stepseq(0,-10,10)- stepseq(10,-10,10);
%Decompose x to odd(x0) and even(xe)
[xe,xo,m] = evenodd(x,n);
subplot(1,3,1)
stem(n,x)
subplot(1,3,2)
stem(n,xe)
subplot(1,3,3)
stem(n,xo)
```

(iii) Plot both **x[n]**, **xe[n]** and **xo[n]** and **comment** on the result.

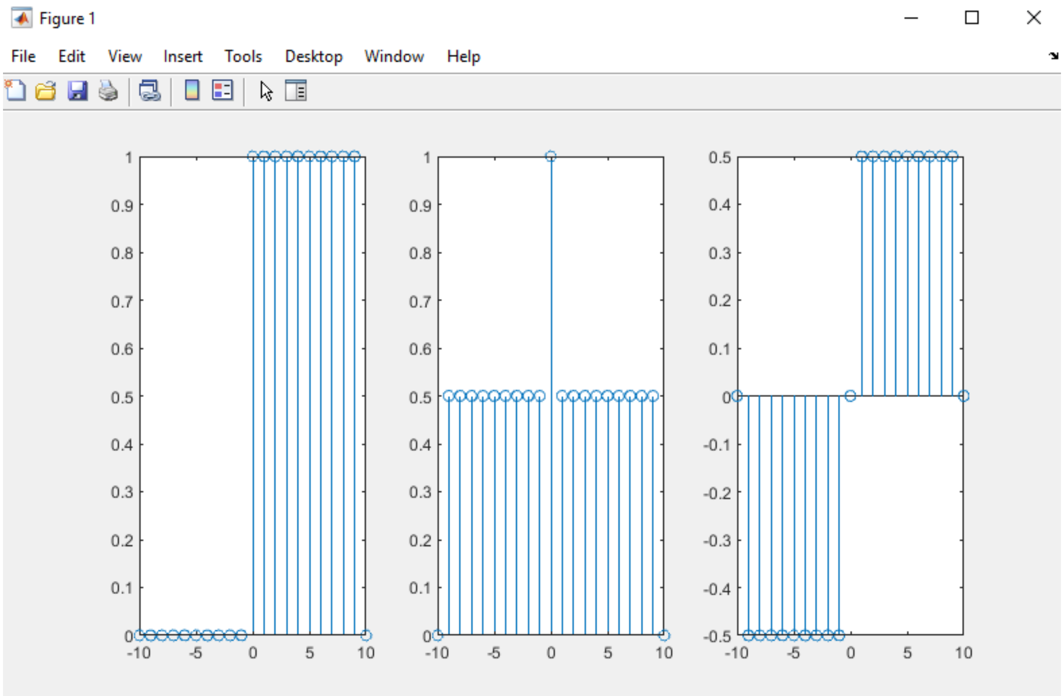


Fig 1: x[n] Fig 2: $x_e[n]$ Fig 3: $x_o[n]$

The second signal is the even signal, and the third signal is the odd signal. When we add the even and the odd signal, we get back the original signal x[n].

4. Use **"dnsample"** function to develop a function that has a form;

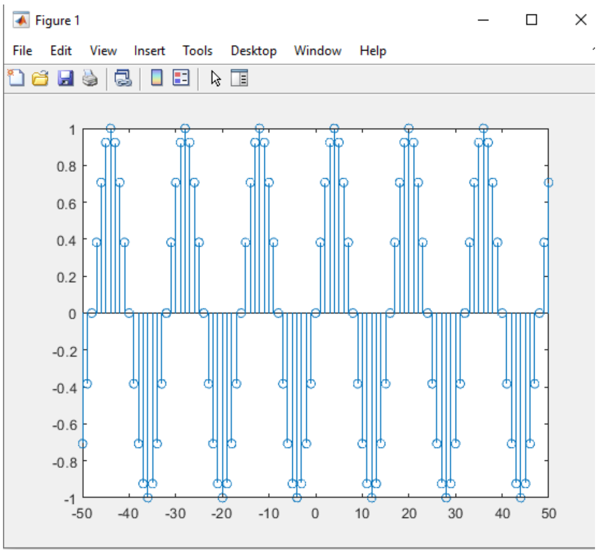
   " function [y,m] = dnsample(x, n, M)"

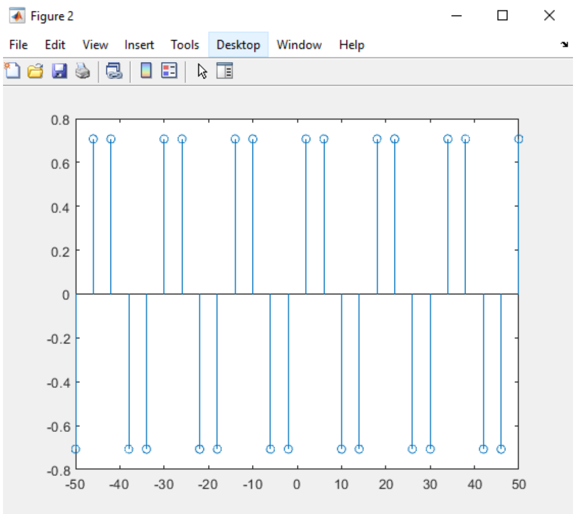
```
%Hemal Sharma
%ID:2221855
function [y,m] = dnsample(x,n,M)
    m = downsample(n,M); %horizontal coordinate
    y = downsample(x,M); %vert coord
end
```

5. (i) Generate **x[n] = sin(0.125πn)** where **-50 ≤ n ≤ 50.**

```
%Hemal Sharma
%ID:2221855
%Generate x[n] = sin(0.125πn) where -50 ≤ n ≤ 50.
n = -50:50;
x = sin(0.125*pi*n);
stem(n,x)
figure(2);
[y,m] = dnsample(x,n,4);
stem(m,y)
```

(ii) **Decimate / down-sample x[n]** by a **factor of 4** to generate **y[n].**





(iii) Here, the function is reducing the sampling rate by a factor of 4. Therefore, the number of samples in the second graph decrease by a factor of 4.

Lab Assignment-4:

(a) Develop a MATLAB function that will generate $x[n^2]$ of a signal.

```
%Hemal Sharma
%ID:2221855
% function to generate x(n^2) from x(n)
function [y, m] = n_square(x, n)
    m = [];     % domain
    y = [];     % range
    c = abs(min(n));    % to set reference value (0)
    % iterate over positive domain (>0) for square numbers
    for i=1:max(n)
        r = sqrt(i);
        if floor(r) == r    % check for perfect square
            % append m (absolute) and y (relative to reference)
            m(end+1) = r;
            y(end+1) = x(c+i+1);
        end
    end
    % correct for negative and 0 positions
    m = [-fliplr(m) 0 m];
    y = [fliplr(y) x(c) y];
end
```

Example:

```
%Hemal Sharma
%ID:2221855
% x(n) = u(n+1) + u(n-2), -5:5
n = -5:5;
x = stepseq(-1,-5,5) + stepseq(2,-5,5);
stem(n,x); grid;
[y, m] = n_square(x,n);
figure(2); stem(m,y); grid;
```

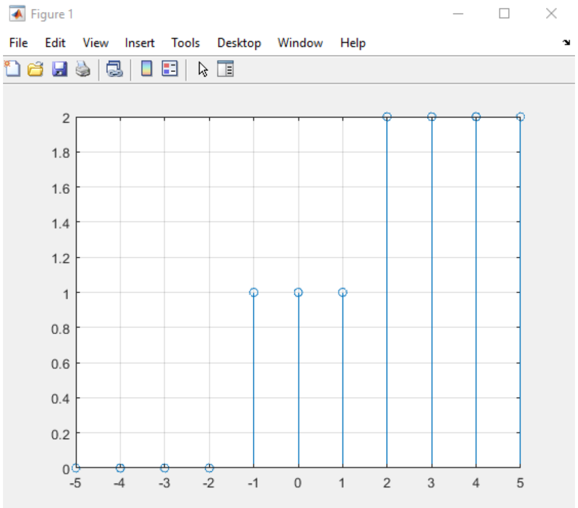


Figure 1: x[n]

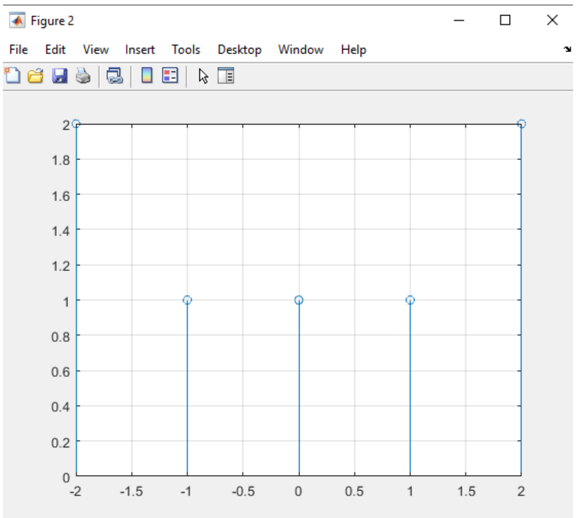


Figure 2: $x[n^2]$

(b) Develop a MATLAB function that will “upsample” a signal.

```
%Hemal Sharma
%ID: 2221855
function [y,m] = upsample(x,n,M)
    m = interp(n,M);
    y = interp(x,M);
end
```