UNITED INTERNATIONAL UNIVERSITY

Lab Report-(01)

Course Title – Digital Signal Processing Laboratory

Course Code- EEE 3301

Submitted To

Dr. Ahmed, Khawza Iftekhar Uddin
Professor
Department of Electrical and Electronic Engineering
United International University, Dhaka, Bangladesh

Submitted By

Group leader

Asikur Rahaman

ID: 021 191 004

Section: A

Members

Ayman Zafar; ID: 021 191 058

Nafiul Islam; ID: 021 182 004

Sagor Paul; ID: 021 191 070

Nafi Ul Kaysar Buruz; ID: 021 221 101

Objectives

The objectives of this lab session were to generate, manipulate and plot discrete time signals using MATLAB. Also, to show different types of operation on discrete time signals by using elementary functions like, Unit sample function, unit step function and real valued exponential function and some custom functions for signal addition, signal multiplication and folding.

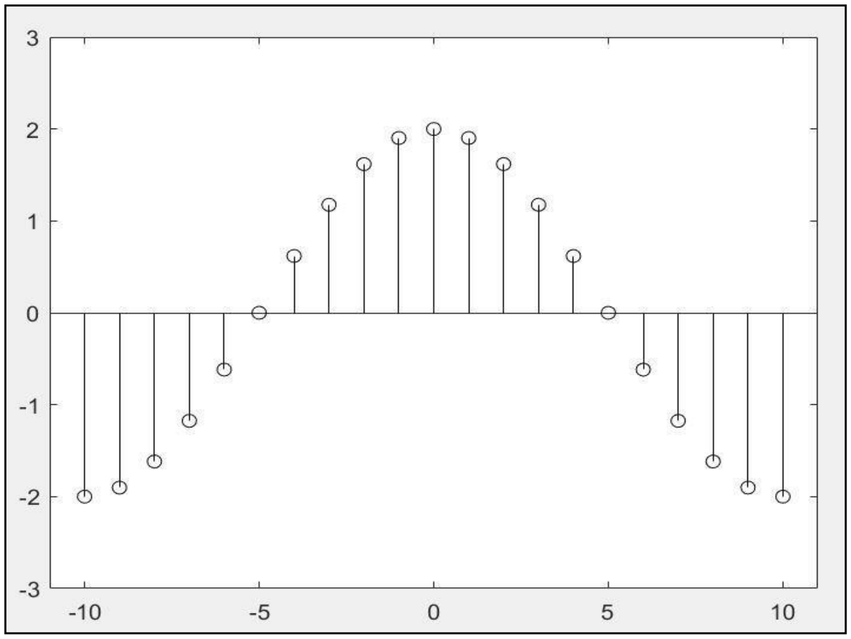
Literature Survey

At the beginning of this session the instructor discussed how to generate a discrete time signal in MATLAB,

Example 1.1

```
n=[-10:10];
x=2*cos(2*pi*0.05*n); stem(n,x,'k'); axis([-11
11 -3 3]);
```

Here, a discrete cosine graph is generated in respect with n which has an array of integer values from -10 to 10 where the sampling frequency is 0.05. Later a built-in code ‘stem’ is used to plot this function in discrete time domain and axis is used to have a zoomed-out view of the graph.

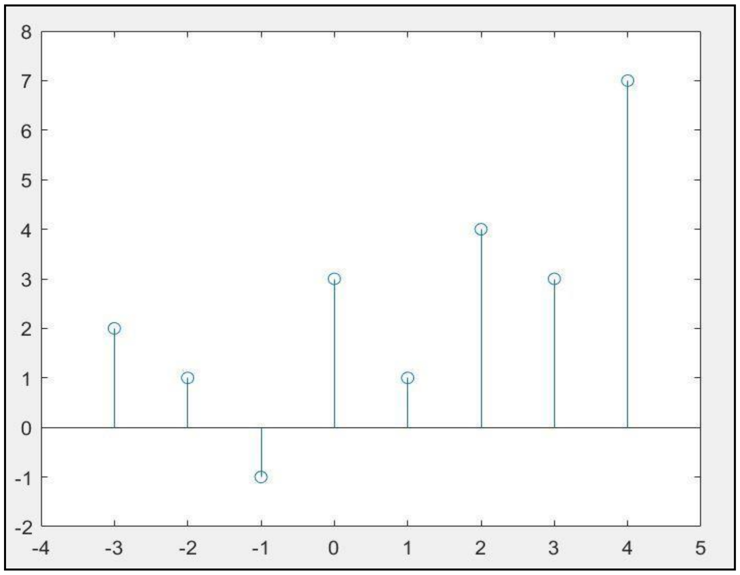


After that, the class moved on to generation of a discrete

sequence Example 1.2

```
n= [-3 -2 -1 0 1 2 3 4];
x= [2 1 -1 3 1 4 3 7];
stem(n, x);
axis([-4 5 -2 8]);
```

In this example, x is a given sequence x(n) = [2, 1, -1, 3, 1, 4, 3, 7] where an arrow is placed on 4th element. Which indicates that the origin of the discrete time axis is placed on the 4th element of that sequence. Therefore, value of n is set to -3 to 4 and then stem is again used to plot the sequence in discrete time domain.



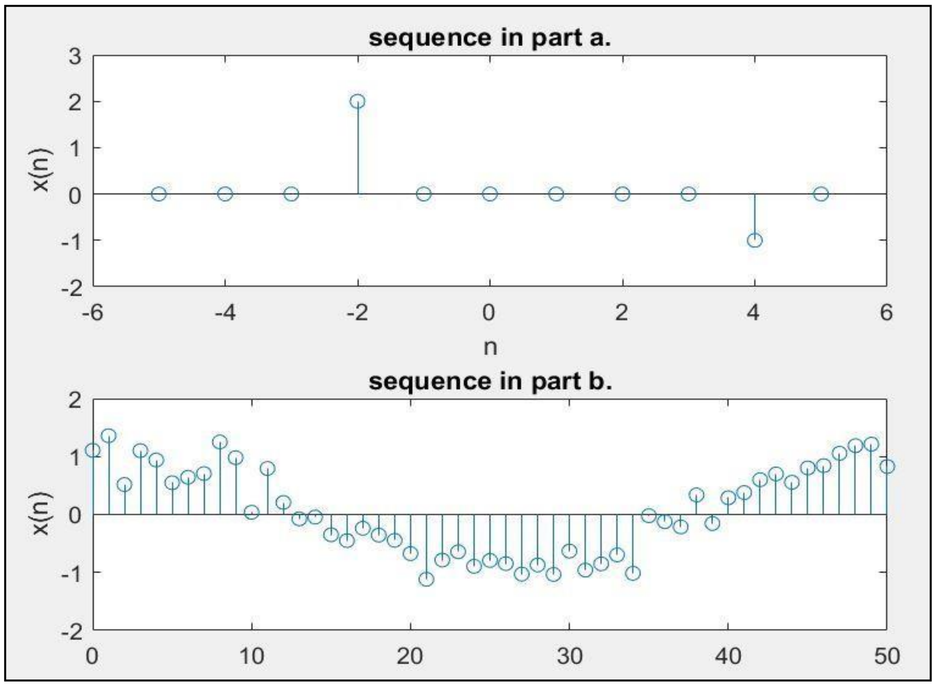
Progressively, the focus moved to another topic where the class learned about using the elementary functions in operation of discrete time signals.

Example-1.3

```
%for part a
n=[-5:5];
x=2*delta(-2,-5,5)-delta(4,-5,5);
subplot(2,1,1);stem(n,x); axis ([-6 6 -2 3]);
title('sequence in part a.');xlabel('n');ylabel('x(n)');

%for part b
n=[0:50];
x=cos(0.04*pi*n)+0.2*randn(size(n));
subplot(2,1,2);stem(n,x);
title('sequence in part b.');xlabel('n');ylabel('x(n)');
```

This syntax is divided in two parts where in first part, delta function is used to generate a shifted impulse signal from -5 to 5 interval as declared in 1st two lines. Then subplot is used to plot multiple signals in one figure, again stem is used to plot the graph in discrete and axis is used to plot a zoomed-out view of the figure. At last, title is used to label the axis of the plotted graph. At second part, a random cosine signal is plotted with respect to a discrete time domain from 0 to 50 range. Again subplot, stem and title are used for plot the graph in discrete informative way.



Next, a new built-in function is introduced which is used to plot periodic signals with a given period,

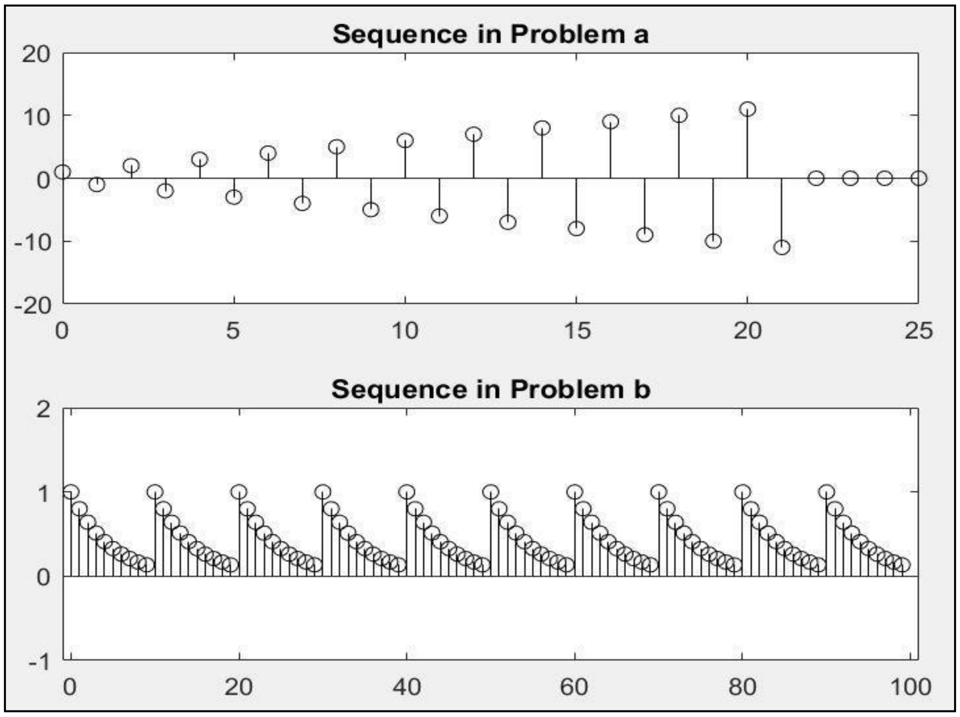
<u>Example-1.4</u>

```
% (a)
n      = [0:25];
x1 = zeros(1,26); for
m=0:10,
        x1 = x1 + (m+1)*(delta(2*m,0,25)-delta(2*m+1,0,25));
end subplot(2,1,1);
stem(n,x1,'k'); title('Sequence in
Problem a')

% (b)
```

```
n=[0:9]';
x=(0.8).^n
N=10;
y = repmat(x,N); %repmat is a library function m=0:1:size(y)-
1;
subplot(2,1,2);
stem(m,y,'k'); axis([-1 101 -1 2]);
title('Sequence in Problem b');
```

In part a, a mixed delta function is potted by initializing an array of zeroes with 27 columns then the mixed delta function is defined in third line with respect to m domain which is an integer with value from 0 to 10. The for loop is written in a manner so the value of m is added after every increment and then the values are added to have a final value of summation. In part b, a real valued exponential is plot in periodic manner with respect to integer n in 0 to 9 range. Periodic repetition is 10 time the normal period, for that repmat function is used. After that, size used to range the number of arrays should be shown in one period. At last, subplot, stem, axis and title are used for discrete plotting in both parts.



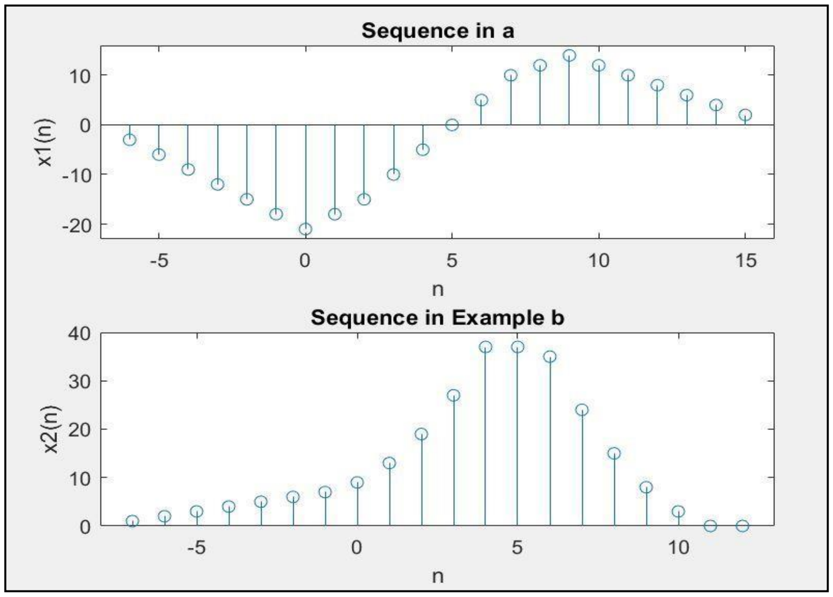
Then we progressed to the usage custom functions in operations of DT sequences, <u>Example-1.5</u>

```
n = -2:10; x = [1:7,6:-1:1];
% a) x1(n) = 2*x(n-5) - 3*x(n+4)
[x11,n11] = sigshift(x,n,5); [x12,n12] = sigshift(x,n,-4); [x1,n1] =
sigadd(2*x11,n11,-3*x12,n12);
```

```
subplot(2,1,1);
stem(n1,x1); title('Sequence in
a');
xlabel('n'); ylabel('x1(n)');
axis([min(n1)-1,max(n1)+1,min(x1)-2,max(x1)+2])

% b) x2(n) = x(3-n) + x(n)*x(n-2)
[x21,n21] = sigfold(x,n); [x21,n21] = sigshift(x21,n21,3); [x22,n22] =
sigshift(x,n,2); [x22,n22] = sigmult(x,n,x22,n22); [x2,n2] =
sigadd(x21,n21,x22,n22);
subplot(2,1,2);
stem(n2,x2);
title('Sequence in Example b')
xlabel('n');
ylabel('x2(n)');
axis([min(n2)-1,max(n2)+1,0,40])
```

For this example, we have used sigshift to shift a signal, sigadd used to add two signals, sigfold is used to fold a signal and sigmult is used to multiply two signals. These are custom functions and used in a manner so two discrete functions made from different combinations of multiple signals can be plotted in one figure. Also, a clever way of using axis is also introduced here.

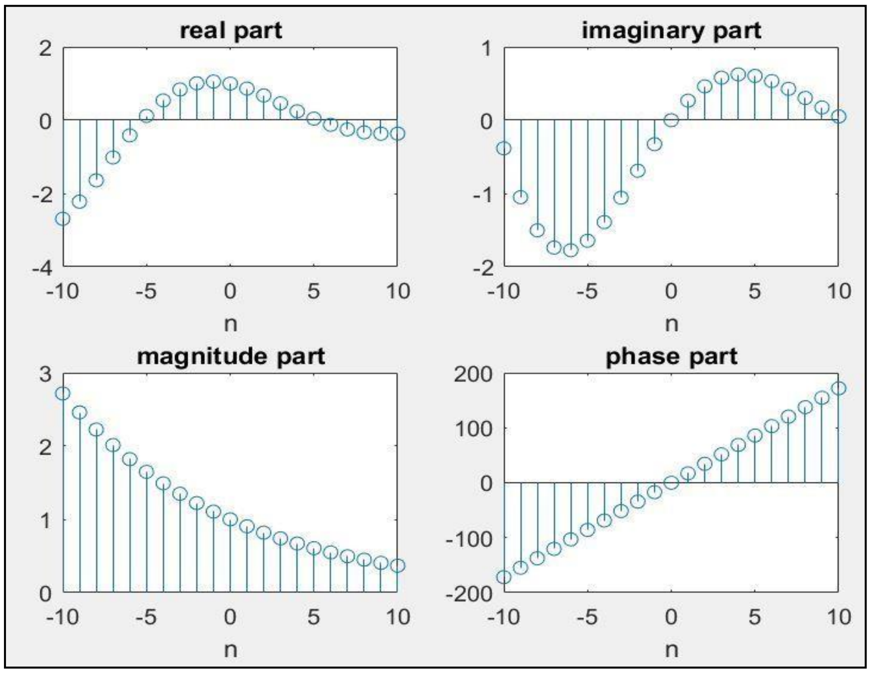


Example-1.6

```
n = [-10:1:10]; alpha = -0.1+0.3j;
x = exp(alpha*n);
subplot(2,2,1);stem(n,real(x));title('real part');xlabel('n')
subplot(2,2,2);stem(n,imag(x));title('imaginary part');xlabel('n')
subplot(2,2,3);stem(n,abs(x));title('magnitude part');xlabel('n')
subplot(2,2,4);stem(n,(180/pi)*angle(x));
title('phase part');xlabel('n')
```

After 1.5 here the session was focused about how to plot a complex graph in both polar and cartesian forms by using some built-in functions.

Figure:

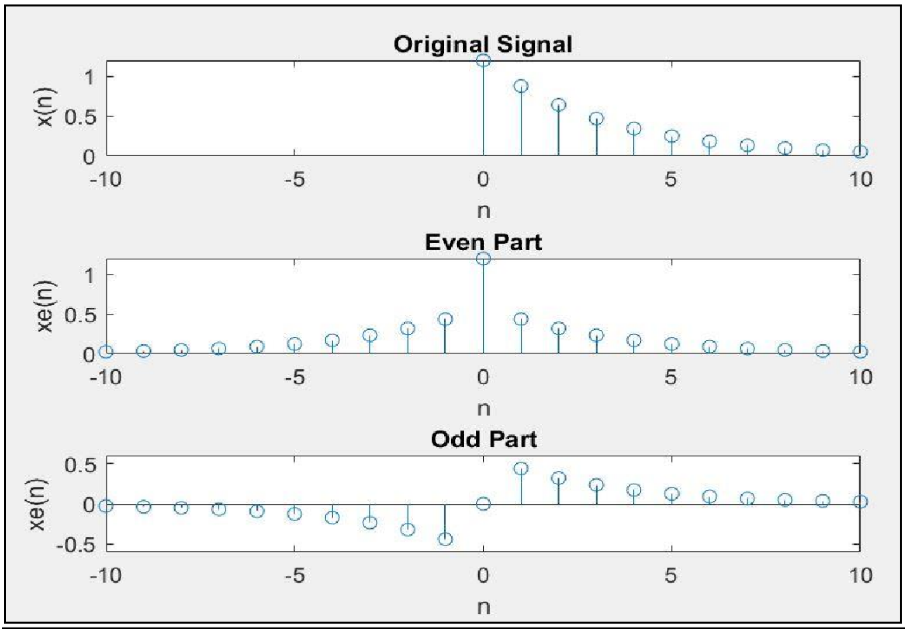


Next, the session was about representing a signal in form of even and odd function and how combining them can reform the signals plot.

Example-1.7

```
n = [0:10];
x=1.2*exp(-0.1*pi*n); [xe,xo,m]
= evenodd(x,n);
subplot(3,1,1); stem(n,x); title('Original Signal') xlabel('n');
ylabel('x(n)'); axis([-10,10,0,1.2]) subplot(312); stem(m,xe); title('Even
Part') xlabel('n'); ylabel('xe(n)'); axis([-10,10,0,1.2]) subplot(313);
stem(m,xo); title('Odd Part') xlabel('n'); ylabel('xe(n)'); axis([-10,10,-
0.6,0.6])
```

Figure:

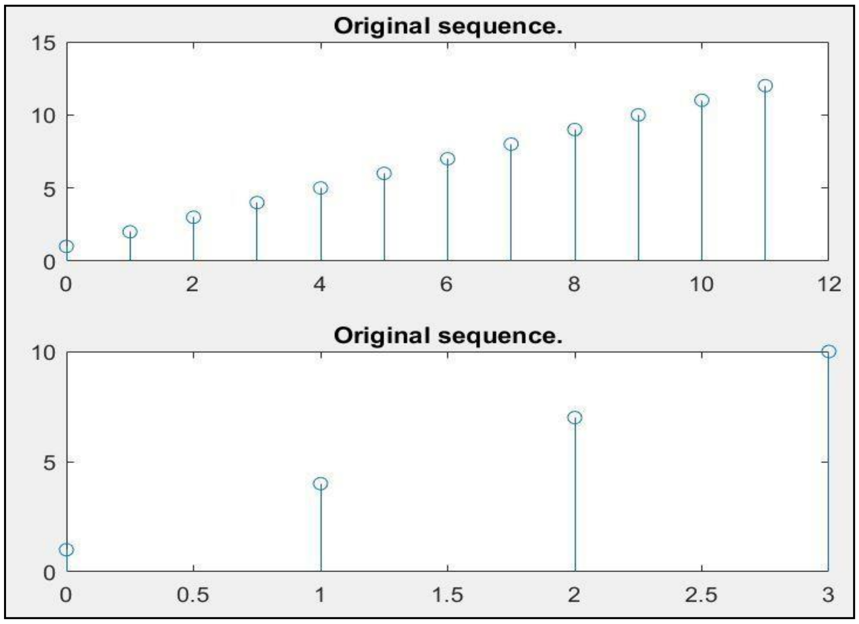


Example-1.8

```
x = [ 1 2 3 4 5 6 7 8 9 10 11 12];
k = 3;
z=downsample(x,k)
n=[0:length(x)-1];
subplot(2,1,1)
stem(n,x)
title('Original sequence.')
m=[0:length(z)-1];
subplot(2,1,2)
stem(m,z)
title('Original sequence.') stem(m,z)
title('Original sequence.')
```

Downsampling is a process where a high frequency signal with less change in peaks can be sampled in a signal with similar picture in continuous time but with a little less data so it becomes easier to manipulate the signal for desired purpose.

Figure:



Home work

Functions (used from question 1 to 5)

Signal Shifting	Signal addition

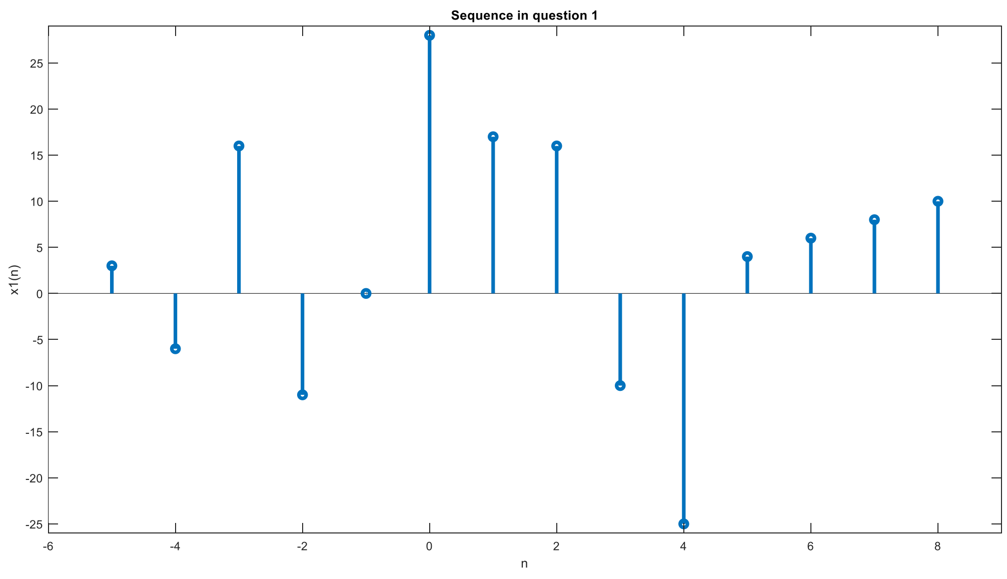
`function [y n]=sigshift(x,m,n0)`
`n=m+n0;`
`y=x;`
`end`
	`function [y n]=sigadd(x1,n1,x2,n2)`
	`n=min(min(n1),min(n2)):max(max(n1),max(n2));`
	`y1=zeros(1,length(n));y2=y1;`
	`y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;`
	`y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;`
	`y=y1+y2;`
	`end`

Answer to the question 1:

script

```
%Question 1
clc; clear all;
%x1(n) = 3x(n+2) + x(n-4) - 2x(n)
n = -3:4; x = [1,-2,6,-5,4:2:10];
[x11,n11] = sigshift(x,n,-2); [x12,n12] = sigshift(x,n,4);
[x0,n0] = sigadd(3*x11,n11,x12,n12);
[x1,n1] = sigadd(x0,n0,-2*x,n);
stem(n1,x1,'LineWidth',3); title('Sequence in question 1')
xlabel('n'); ylabel('x1(n)');
axis([min(n1)-1,max(n1)+1,min(x1)-1,max(x1)+1])
```

output numerical and figure of question 1

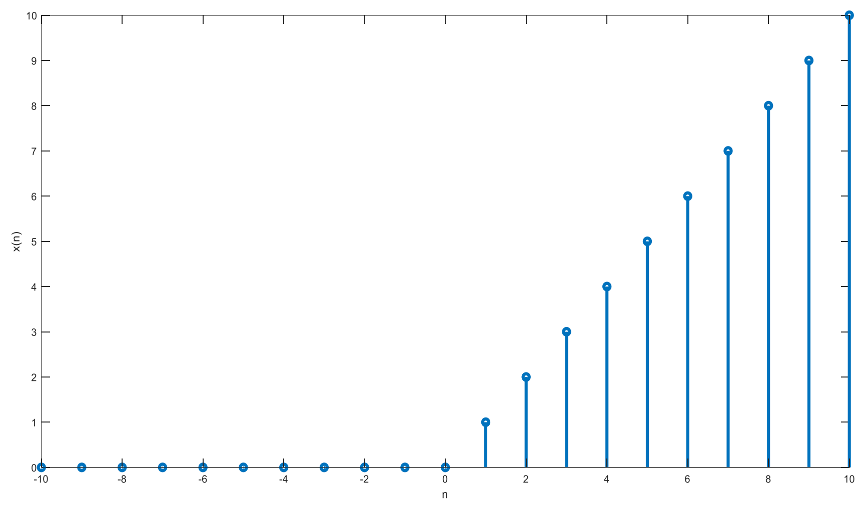


Answer to the question 2:

script

```
%Question 2
(a)
function [x n]= r(n0,n1,n2)
n=[n1:n2];
u=[n-n0]>=0;
x = n.*u;
(b)
clc; clear all;
[x n] = r(0,-10,10);
stem(n,x,'LineWidth',3)
xlabel('n'); ylabel('x1(n)');
```

output numerical and figure of question 2

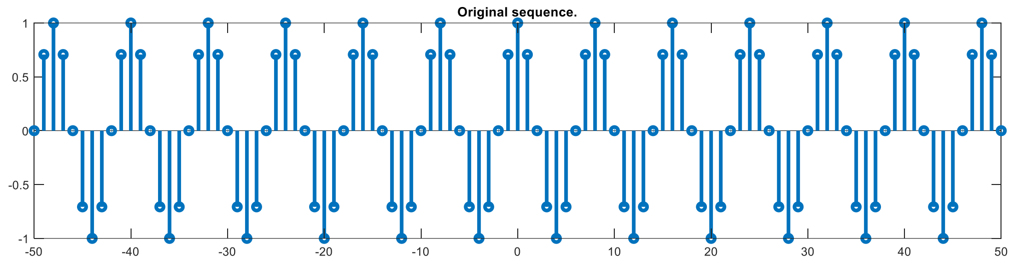


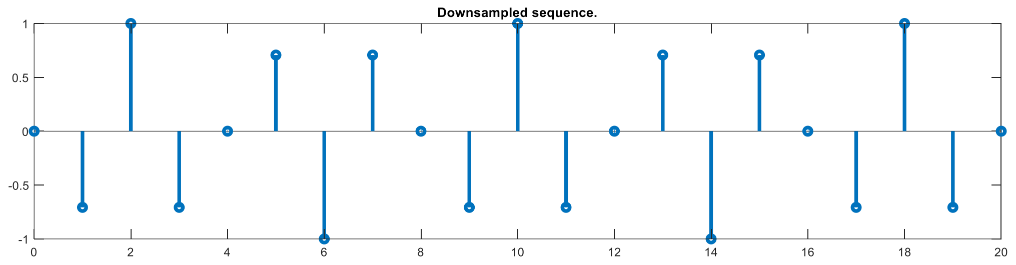
<u>Answer to the question 3:</u>

<u>script</u>

```
%Question 3
clc; clear all;
n = [-50:50];
x = cos(0.25*pi*n);
k = 5;
y = downsample(x,k);
subplot(2,1,1)
stem(n,x,'LineWidth',3)
title('Original sequence.')
m = [0:length(y)-1];
subplot(2,1,2)
stem(m,y,'LineWidth',3)
title('Downsampled sequence.')
```

<u>output numerical and figure of question 3</u>



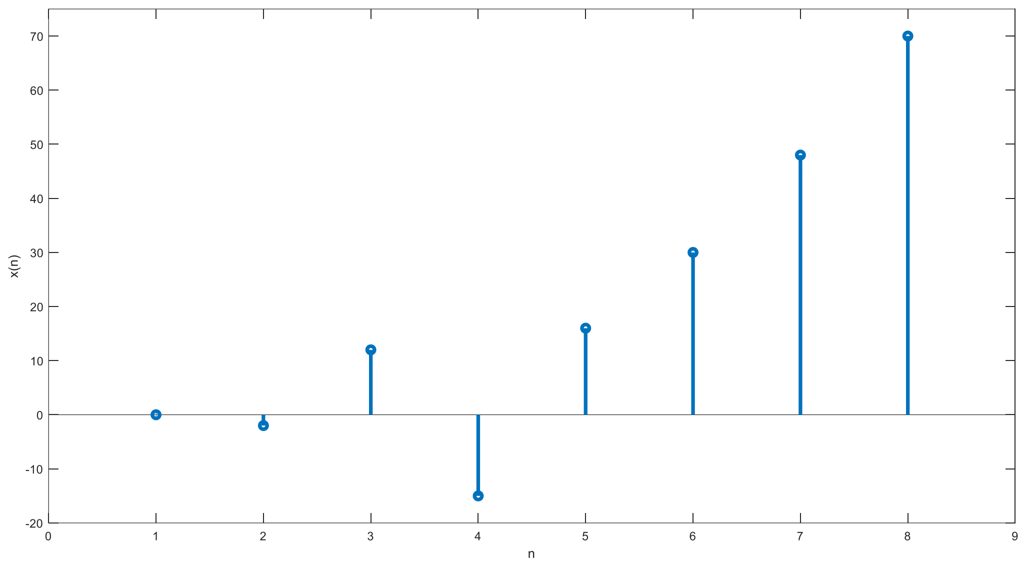


Answer to the question 4:

script

```
%Question 4
clc; clear all;
n = [0:7];
x = [1,-2,6,-5,4:2:10];
for m = 1:5
    if m == 1
        [x1,n1] = sigshift(x,n,m);
        x1 = n.*x1;
    else
        [x2,n2] = sigshift(x,n,m);
        [x3,n3] = sigadd(n.*x2,n2,x1,n1);
        x3=x1;n3=n1;
    end
end
stem(n3,x3,'LineWidth',3);
xlabel('n'); ylabel('x(n)');
axis([min(n3)-1,max(n3)+1,min(x3)-5,max(x3)+5])
```

output numerical and figure of question 4



Answer to the question 5:

script for question 5(i)

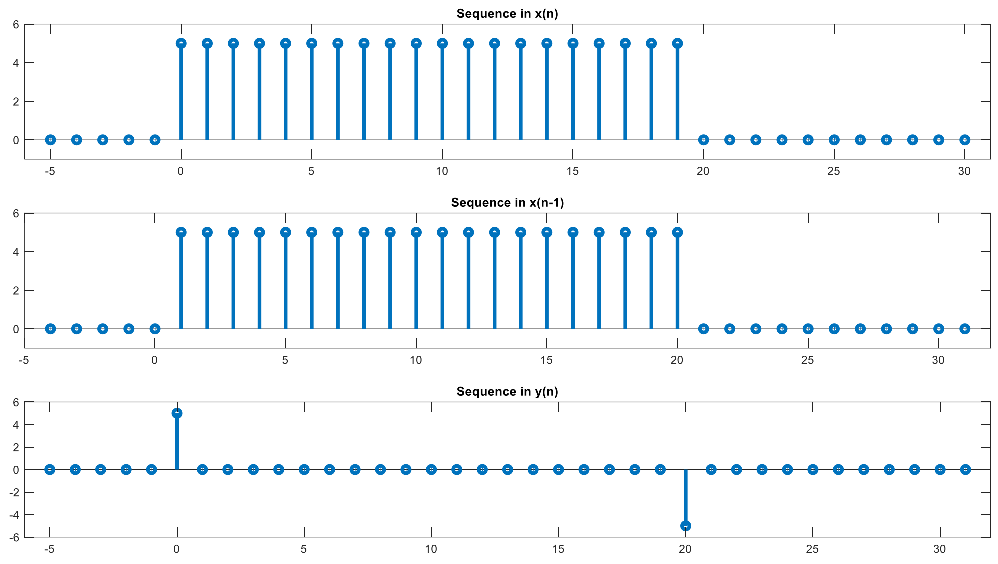
```
%Question 5
%(i)
clc; clear all;
n = [-5:30];
x = 5*(u(0,-5,30)-u(20,-5,30));
subplot(3,1,1)
stem(n,x,'LineWIdth',3)
title('Sequence in x(n)')
axis([min(n)-1,max(n)+1,min(x)-1,max(x)+1])

[x1,n1] = sigshift(x,n,1);
subplot(3,1,2)
stem(n1,x1,'LineWIdth',3)
title('Sequence in x(n-1)')
axis([min(n1)-1,max(n1)+1,min(x1)-1,max(x1)+1])

[x2,n2] = sigadd(-x1,n1,x,n);
subplot(3,1,3)
stem(n2,x2,'LineWIdth',3)
title('Sequence in y(n)')
```

```
axis([min(n2)-1,max(n2)+1,min(x2)-1,max(x2)+1])
```

output numerical and figure of question 5(i)



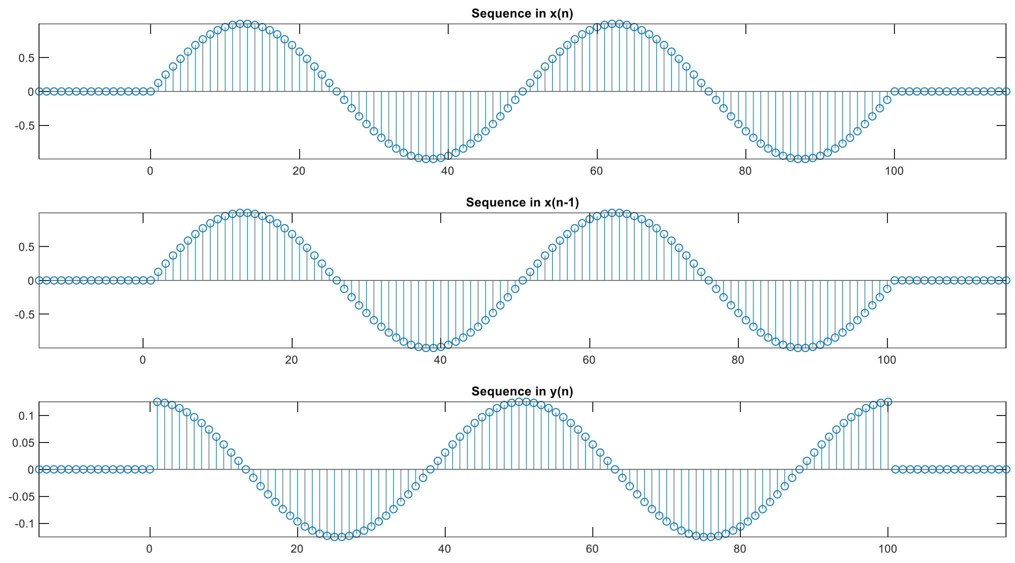
script for question 5(ii)

```
%(ii)
clc; clear all;
n = [-15:115];
x = (sin((pi/25)*n)).*(u(0,-15,115)-u(100,-15,115));
subplot(3,1,1)
stem(n,x)
title('Sequence in x(n)')
axis([min(n),max(n),min(x),max(x)])

[x1,n1] = sigshift(x,n,1);
subplot(3,1,2)
stem(n1,x1)
title('Sequence in x(n-1)')
axis([min(n1),max(n1),min(x1),max(x1)])

[x2,n2] = sigadd(-x1,n1,x,n);
subplot(3,1,3)
stem(n2,x2)
title('Sequence in y(n)')
axis([min(n2),max(n2),min(x2),max(x2)
```

output numerical and figure of question 5(ii)



Conclusion

We learned how to describe discrete-time signals mathematically and generate, manipulate, and plot discrete time signals using MATLAB. We also understood different operations on DT signals. Generation of some custom function such as unit sample function, unit step function and real valued exponential sequence can be applied to operate DT signals. Using some more custom functions, we can add, multiply, shift, reflect and decompose to even and odd part of a DT signal.