AV 336- Digital Signal Processing Lab

Labsheet 2

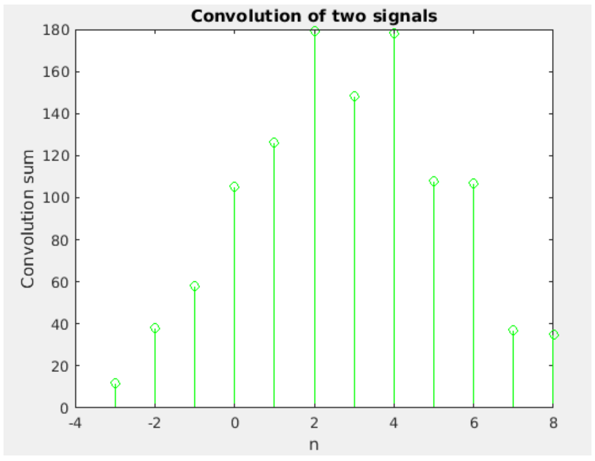
Submitted by: K.V.N.G.Vikram
SC15B148
E.P Roll No- 8

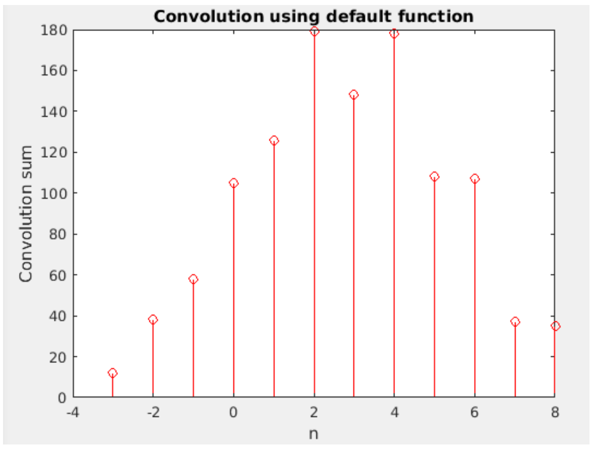
<u>% Q1</u>

1. Suppose x_1 and x_2 are two finite sequences defined as
$x_1[n] = [4; 2; 6; 3; 8; 1; 5]$
$x_2[n] = [3; 8; 6; 9; 6; 7]$
Let the starting index of $x_1[n]$ be -1 (i.e. $x_1[-1] = 4; x_1[0] = 2 : : :$) and the starting index of $x_2[n]$ be -2. Obtain the convolution of $x_1[n]$ and $x_2[n]$ using Matlab
(a) You should first try an implementation on your own without using any inbuilt functions.
(b) Now find out whether there are any inbuilt Matlab functions that can be used to compute convolution and use that.
Ans:

```
clear;
clc;

x = [ 4 2 6 3 8 1 5 ] % x1
h = [ 3 8 6 9 6 7 ]   % x2
ax = -1      % starting point of x1
ah = -2      % starting point of x2

% selecting the final start point
a = ax+ah;

% a) first part of solution

m = length(x);
n = length(h);
X = [x,zeros(1,n)];
H = [h,zeros(1,m)];

% number of terms after convolution n + m - 1
for i = 1:m+n-1
    Y(i)=0; % defining the i'th term 0 initially
    for j = 1:m % for all the terms in x
```

```
        if(i-j+1>0) % but not the terms which are not overlapped
            Y(i) = Y(i) + X(j)*H(i-j+1); % summation to Y and H is flipped
        end
    end
end

% defining the x axis for the the convolution solution
x1 = a:a+numel(Y)-1;

% plotting

stem(x1,Y,'g')
ylabel('Convolution sum')
xlabel('n')
title('Convolution of two signals')

% b) second part of solution

% conv(X,H) is the default function for the convolution of two signals
figure()
stem(x1,conv(x,h),'r')
ylabel('Convolution sum')
xlabel('n')
title('Convolution using default function')
```



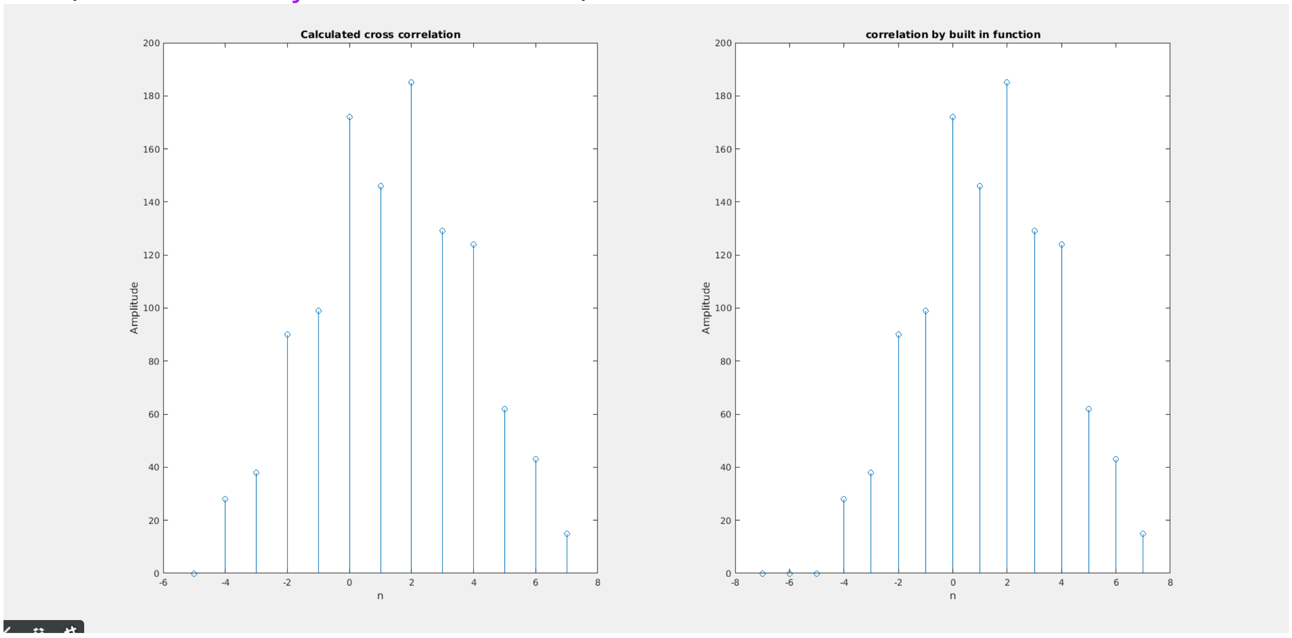


<u>% Q2</u>

2. Find out what is meant by auto-correlation of a discrete time signal x[n]. Find out the auto-correlation of x1[n] and x2[n] defined above using your own code. Find out whether there is a builtin Matlab function for finding out the auto-correlation. Find out the auto-correlation
of x1[n] and x2[n] using the builtin function(s). Find out whether auto-correlation can be implemented using convolution.

Ans)

```
clear;
clc;

x = [ 4 2 6 3 8 1 5 ] % x1
h = [ 3 8 6 9 6 7  ]   % x2
ax = -1      % starting point of x1
ah = -2      % starting point of x2

% padding zeros for adjusting origen of two signals at same index
if ax>ah
    x = [zeros(1,ax-ah) x];
elseif ax<ah
    h = [zeros(1,ah-ax) h];
end

% defining the final cross correlation solution
Y = zeros(1,numel(x)+numel(h)-1);

% x axis variable of final solution
s = -numel(h)+1:numel(x)-1;

% padding zeros to x
xa = [x , zeros(1,(numel(Y)-numel(x)))];


for i = 1:numel(Y) % caluclating for each element of Y

    Y(i) = 0; % initially assigned zero

    for j = 1:i

        if (j-s(i))>0  % not considering non overlapping elements of shifted h
            % h is shifted and multiplied to x element wise by the
            % corresponding shift and added to Y(i)
            Y(i) = Y(i)+xa(j)*h(j-s(i));
        end

    end

end

% plotting the calculated values
subplot(1,2,1)
stem(s,Y)
xlabel('n')
ylabel('Amplitude')
title('Calculated cross correlation')

% by inbuilt function 'xcorr(x,h)'
% xcorr() has different dimensions and shift as of Y
% so corresponding shift 'r' is used adjust x axis vector 's'
if numel(x)==numel(h)
    r = 0;
else
    r = numel(x)-numel(h)
end

 v = [s(1)-r:s(1)+numel(xcorr(x,h))-r-1];
 u = xcorr(x,h);

subplot(1,2,2)
stem(v,u)
xlabel('n')
ylabel('Amplitude')
title('correlation by built in function')
```



% Q3

3. Find out what is meant by cross-correlation between two discrete time signals x1[n] and x2[n]. Find out whether cross-correlation is implemented as a builtin function in Matlab. Find the cross-correlation between x1[n] and x2[n] using your own code as well as builtin functions.
Find out whether cross-correlation can be implemented using convolution.

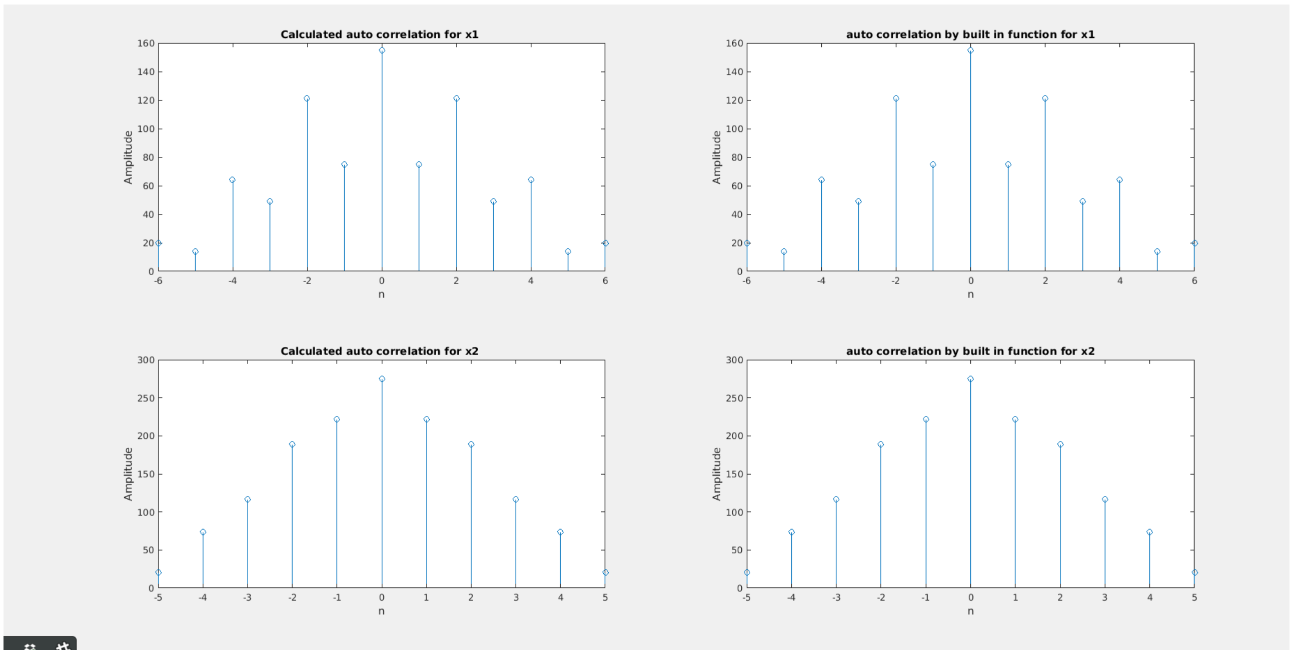
Ans)

```
clear;
clc;
myx = [ 4 2 6 3 8 1 5 ] % x1
myh = [ 3 8 6 9 6 7  ]  % x2
myax = -1      % starting point of x1
myah = -2      % starting point of x2
```

```
% for auto correlation is same as cross correlation with both the signals same

% for x1
h = myx;
x = myx;
ah = myax;
ax = myax;

% rest of the code is same as for cross correlation

% padding zeros for adjusting origen of two signals at same index
if ax>ah
    x = [zeros(1,ax-ah) x];
elseif ax<ah
    h = [zeros(1,ah-ax) h];
end

% defining the final cross correlation solution
Y = zeros(1,numel(x)+numel(h)-1);

% x axis variable of final solution
s = -numel(h)+1:numel(x)-1;

% padding zeros to x
xa = [x , zeros(1,(numel(Y)-numel(x)))];


for i = 1:numel(Y) % caluclating for each element of Y

    Y(i) = 0; % initially assigned zero

    for j = 1:i

        if (j-s(i))>0  % not considering non overlapping elements of shifted h
            % h is shifted and multiplied to x element wise by the
            % corresponding shift and added to Y(i)
            Y(i) = Y(i)+xa(j)*h(j-s(i));
        end

    end

end

% plotting the calculated values
subplot(2,2,1)
stem(s,Y)
xlabel('n')
ylabel('Amplitude')
title('Calculated auto correlation for x1 ')

% by inbuilt function 'xcorr(x,h)'

% xcorr() has different dimensions and shift as of Y
% so corresponding shift 'r' is used adjust x axis vector 's'
if numel(x)==numel(h)
```

```
    r = 0;
else
    r = numel(x)-numel(h)
end

 v = [s(1)-r:s(1)+numel(xcorr(x,h))-r-1];
 u = xcorr(x,h);

subplot(2,2,2)
stem(v,u)
xlabel('n')
ylabel('Amplitude')
title('auto correlation by built in function for x1')


% for x2
x = myh;
ax = myah;
h = myh;
ah = myah;

% rest of the code is same as for cross correlation

% padding zeros for adjusting origen of two signals at same index
if ax>ah
    x = [zeros(1,ax-ah) x];
elseif ax<ah
    h = [zeros(1,ah-ax) h];
end

% defining the final cross correlation solution
Y = zeros(1,numel(x)+numel(h)-1);

% x axis variable of final solution
s = -numel(h)+1:numel(x)-1;

% padding zeros to x
xa = [x , zeros(1,(numel(Y)-numel(x)))];


for i = 1:numel(Y) % caluclating for each element of Y

    Y(i) = 0; % initially assigned zero

    for j = 1:i

        if (j-s(i))>0  % not considering non overlapping elements of shifted h
            % h is shifted and multiplied to x element wise by the
            % corresponding shift and added to Y(i)
            Y(i) = Y(i)+xa(j)*h(j-s(i));
        end

    end

end
```

```
% plotting the calculated values
subplot(2,2,3)
stem(s,Y)
xlabel('n')
ylabel('Amplitude')
title('Calculated auto correlation for x2 ')

% by inbuilt function 'xcorr(x,h)'

% xcorr() has different dimensions and shift as of Y
% so corresponding shift 'r' is used adjust x axis vector 's'
if numel(x)==numel(h)
    r = 0;
else
    r = numel(x)-numel(h)
end

 v = [s(1)-r:s(1)+numel(xcorr(x,h))-r-1];
 u = xcorr(x,h);

subplot(2,2,4)
stem(v,u)
xlabel('n')
ylabel('Amplitude')
title('auto correlation by built in function for x2')
```

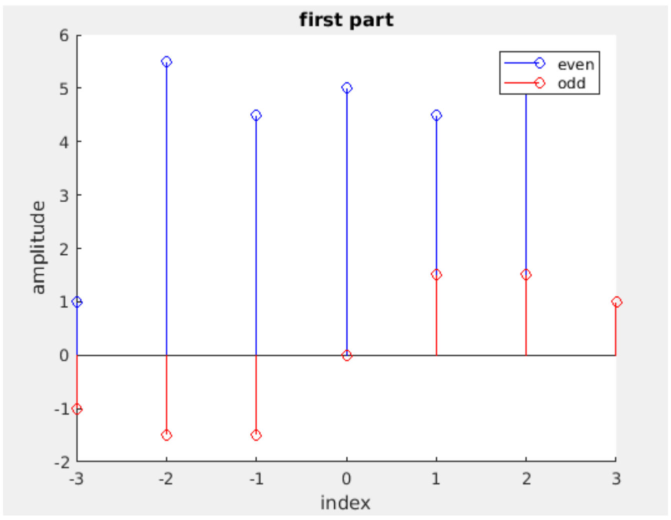


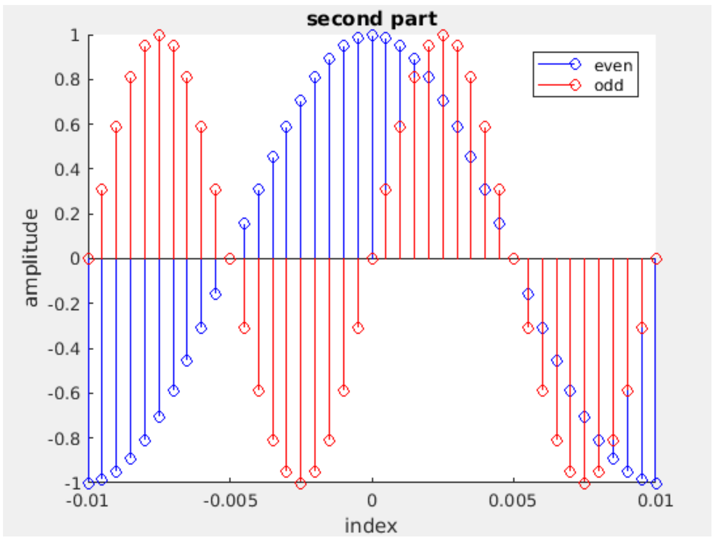
% Q4

4. Define what odd and even signals are. Write Matlab code to find out the odd and even parts of the following signals:
(a) x1[n] = [4; 3; 5; 6; 7; 2]; starting index is -2

(b) x2[n] = sin(2_100n)+cos(_100n) for all n (but choose an appropriate finite time extent for implementation)
Ans:

```
clc
clear

%a
x1 = [0 4 3 5 6 7 2];
n1 = -(numel(x1)-1)/2 : (numel(x1)-1)/2

figure()
hold()
title('first part')
even1 = (x1 + flip(x1))/2;
odd1  = (x1 - flip(x1))/2;

stem(n1,even1,'b')
stem(n1,odd1,'r')
legend('even','odd')
xlabel('index')
ylabel('amplitude')


%b
n2 = -0.01:0.0005:0.01;
x2 = sin(2*pi*100*n2)+cos(pi*100*n2);

figure()
hold()
title('second part')
even = (x2 + flip(x2))/2;
odd  = (x2 - flip(x2))/2;
stem(n2,even,'b')
stem(n2,odd,'r')
legend('even','odd')
xlabel('index')
ylabel('amplitude')
```





% Q5

5. Suppose x1 and x2 are two finite sequences defined as
x1[n] = [4; 2; 6; 3; 8; 1]
x2[n] = [3; 8; 6; 9; 6; 7]
Write a Matlab program to compute the circular convolution of x1[n] and x2[n]. Modify your Matlab program such that it can compute the circular convolution of any x1 and x2 given as input - include in your program logic to check whether the circular convolution can be
computed and return appropriate error messages.

```
% circular convolution

x1 = [4 2 6 3 8 1]
x2 = [3 8 6 9 6 7]

% checking if length of arrays are equal
% if not then zeros are padded at the end
if numel(x1) == numel(x2)
    disp('array length matching..')
else
    disp('error !! array legths not equal')
    if numel(x1)>numel(x2)
        disp('padding zeros on the right of X2')
        x2 = padarray(x2,[0 (numel(x1)-numel(x2))],0,'post')
    else
        disp('padding zeros on the right of X1')
        x1 = padarray(x1,[0 (numel(x2)-numel(x1))],0,'post')
    end
end

c = zeros(size(x1));
x1 = flip(x1)';
x2=x2';
for i = 1 : numel(x1)
    c(i) = sum(x2.*circshift(x1,i));
end
disp('Final array after circular convolution is ')
c
```

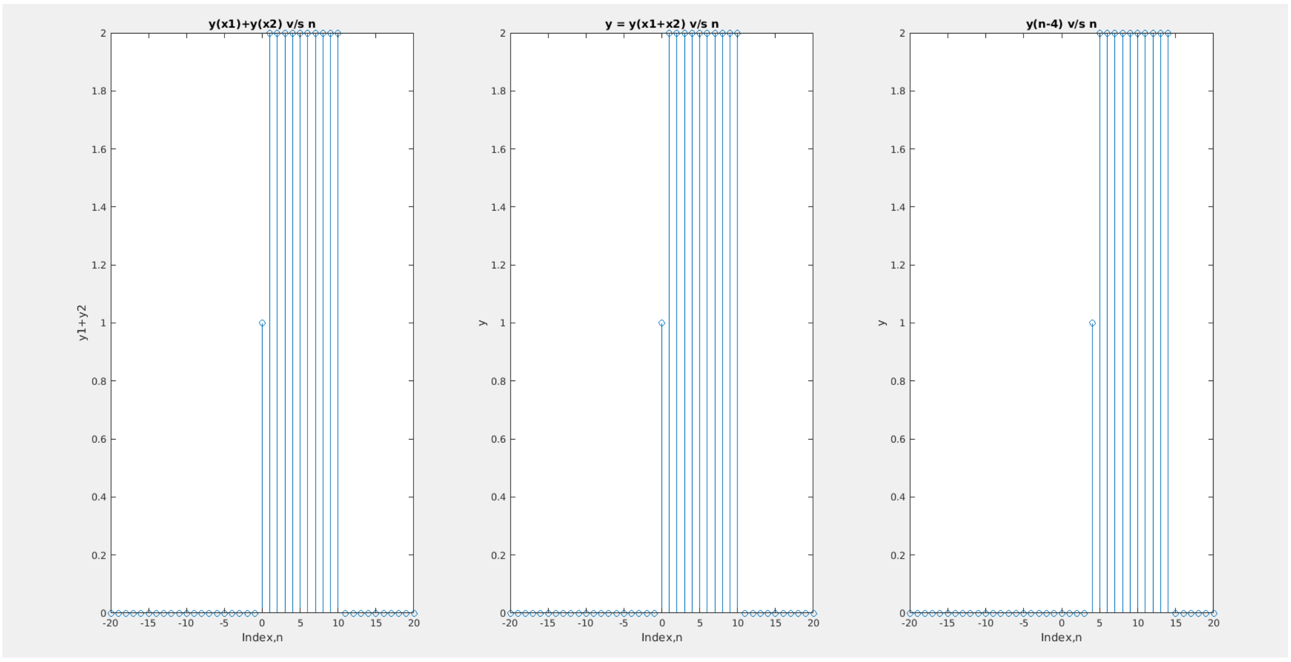
% Q7

7. Let x1[n] and x2[n] be two signals defined as
x1[n] = u[n] - u[n -10];
x2[n] =n; for n = 0; 1; : : : ; 10;
0; otherwise.

Here u[n] is the standard step function. State whether the following systems (the input output relationship are given) are linear and time invariant. Using Matlab check whether the systems satisfy the linearity and time invariance property for the above candidate input signals and
time delays of -1 and 1.
(a) y[n] = x[n - 3] _ x[n - 2]
(b) y[n] = x[n + 2]
(c) y[n] = sin(x[n])
(d) y[n] = x[2n]
Ans:

```
% linear and time invarient
clear;
n=-20:20;

%Defining x1 and x2
x1 = 1*double(n>=0 & n<=10);
x2 = 1*double(n>0 & n<=10);



y1=zeros(1,length(n));

%Checking linearity

x=x1+x2;
xt = circshift(x,4);
y=x;

% following y will be the output functions
yt = xt;
y1 = x1;
y2 = x2;



subplot(1,3,1);
stem(n,y1+y2);
xlabel('Index,n');
ylabel('y1+y2');
title('y(x1)+y(x2) v/s n');

subplot(1,3,2);
stem(n,y);
xlabel('Index,n');
ylabel('y');
title('y = y(x1+x2) v/s n');
```

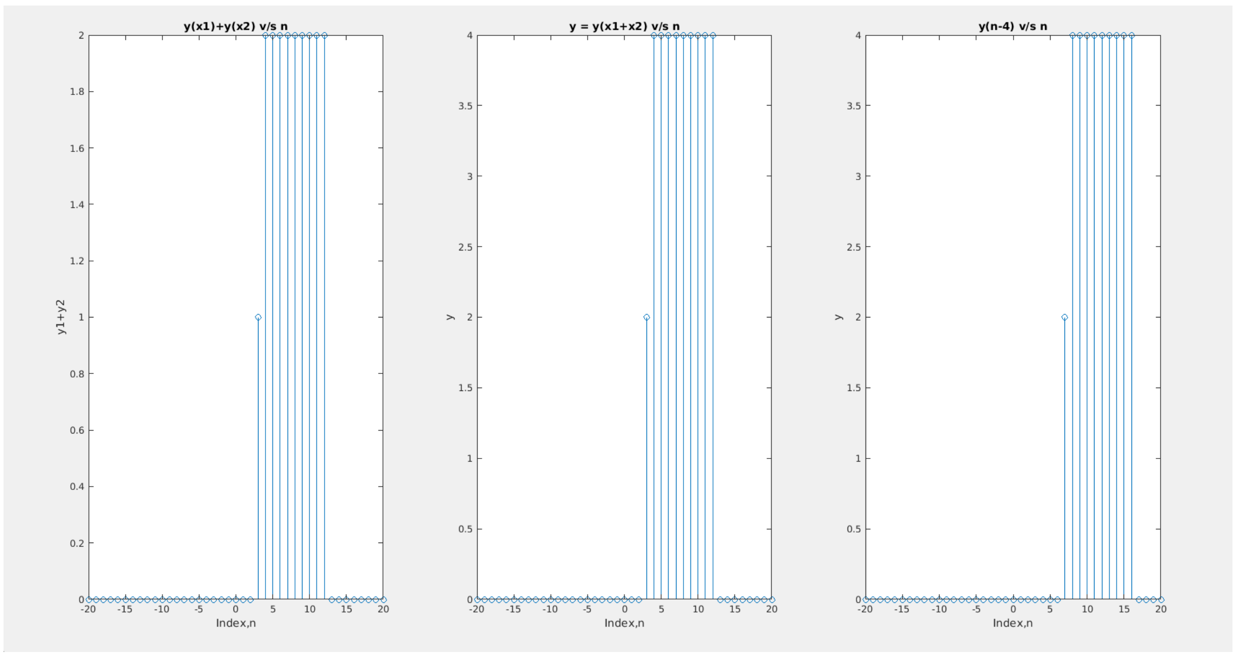
```
subplot(1,3,3);
stem(n,yt);
xlabel('Index,n');
ylabel('y');
title('y(n-4) v/s n')
```



<u>% a)</u>

```
% a is non linear and time invarient
clear;
n=-20:20;

%Defining x1 and x2
x1 = 1*double(n>=0 & n<=10);
x2 = 1*double(n>0 & n<=10);


%Checking linearity and time invarience

x=x1+x2;
xt = circshift(x,4); % 4 is arbitary number to check time invarience


% output functions
y = circshift(x,3).*circshift(x,2);
yt = circshift(xt,3).*circshift(xt,2);
y1 = circshift(x1,3).*circshift(x1,2);
y2 = circshift(x2,3).*circshift(x2,2);


subplot(1,3,1);
```

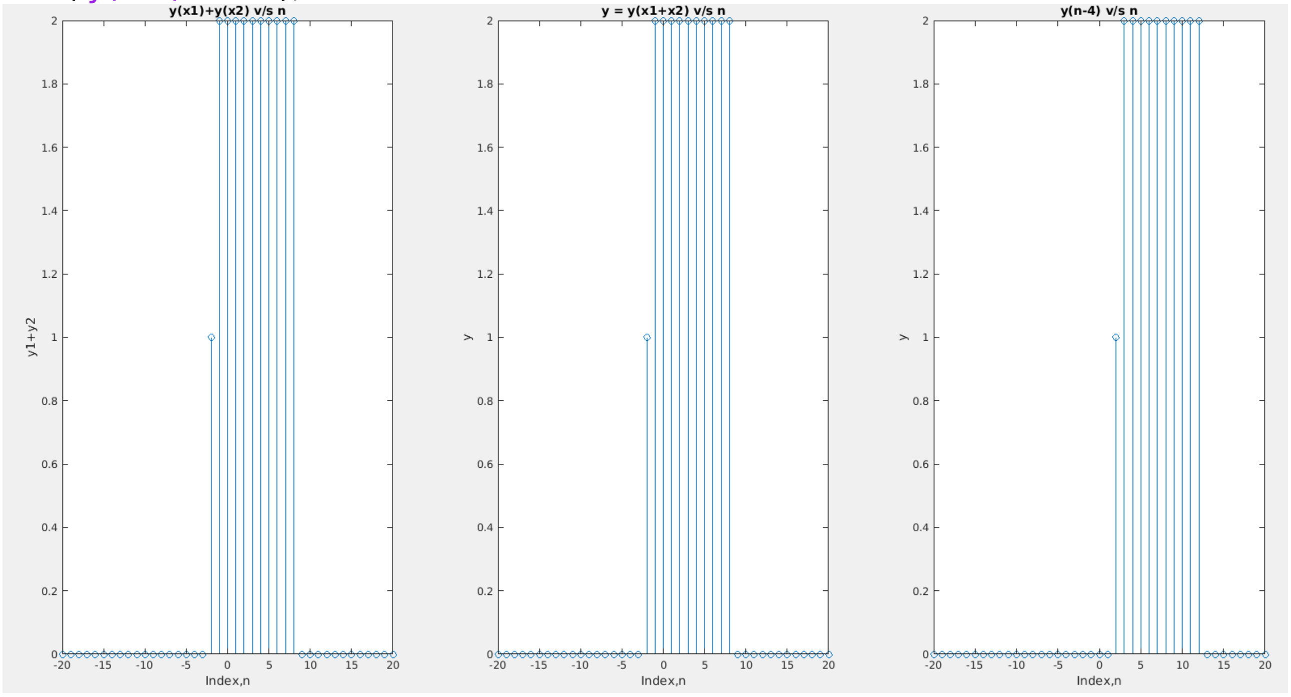
```
stem(n,y1+y2);
xlabel('Index,n');
ylabel('y1+y2');
title('y(x1)+y(x2) v/s n');

subplot(1,3,2);
stem(n,y);
xlabel('Index,n');
ylabel('y');
title('y = y(x1+x2) v/s n');


subplot(1,3,3);
stem(n,yt);
xlabel('Index,n');
ylabel('y');title('y(n-4) v/s n');
```



%
b)

```
% b is linear and time invarient
clear;
n=-20:20;

%Defining x1 and x2
x1 = 1*double(n>=0 & n<=10);
x2 = 1*double(n>0 & n<=10);


%Checking linearity and time invarience

x=x1+x2;
xt = circshift(x,4); % 4 is arbitary number to check time invarience
```

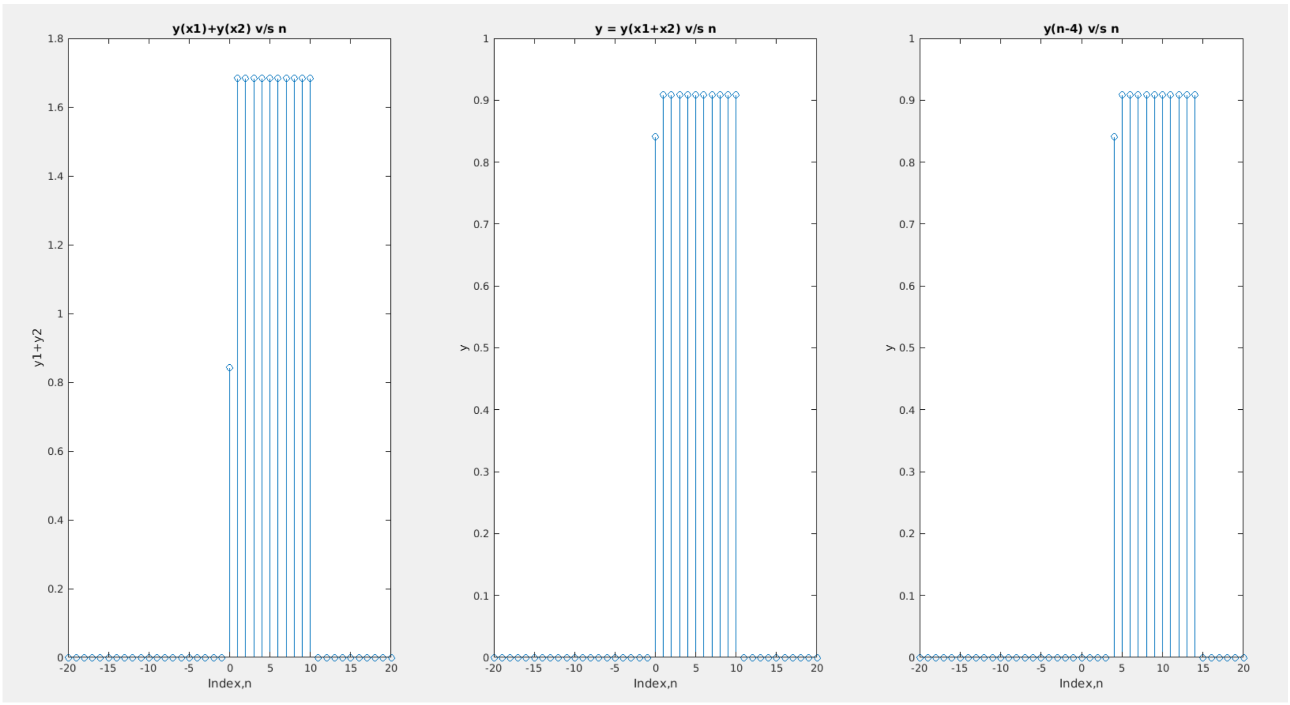
```
% output functions
y = circshift(x,-2);
yt = circshift(xt,-2);
y1 = circshift(x1,-2);
y2 = circshift(x2,-2);


subplot(1,3,1);
stem(n,y1+y2);
xlabel('Index,n');
ylabel('y1+y2');
title('y(x1)+y(x2) v/s n');

subplot(1,3,2);
stem(n,y);
xlabel('Index,n');
ylabel('y');
title('y = y(x1+x2) v/s n');

subplot(1,3,3);
stem(n,yt);
xlabel('Index,n');
ylabel('y');
title('y(n-4) v/s n');
```



<u>% c)</u>

```
% c is non linear and time invarient
clear;
n=-20:20;

%Defining x1 and x2
x1 = 1*double(n>=0 & n<=10);
```

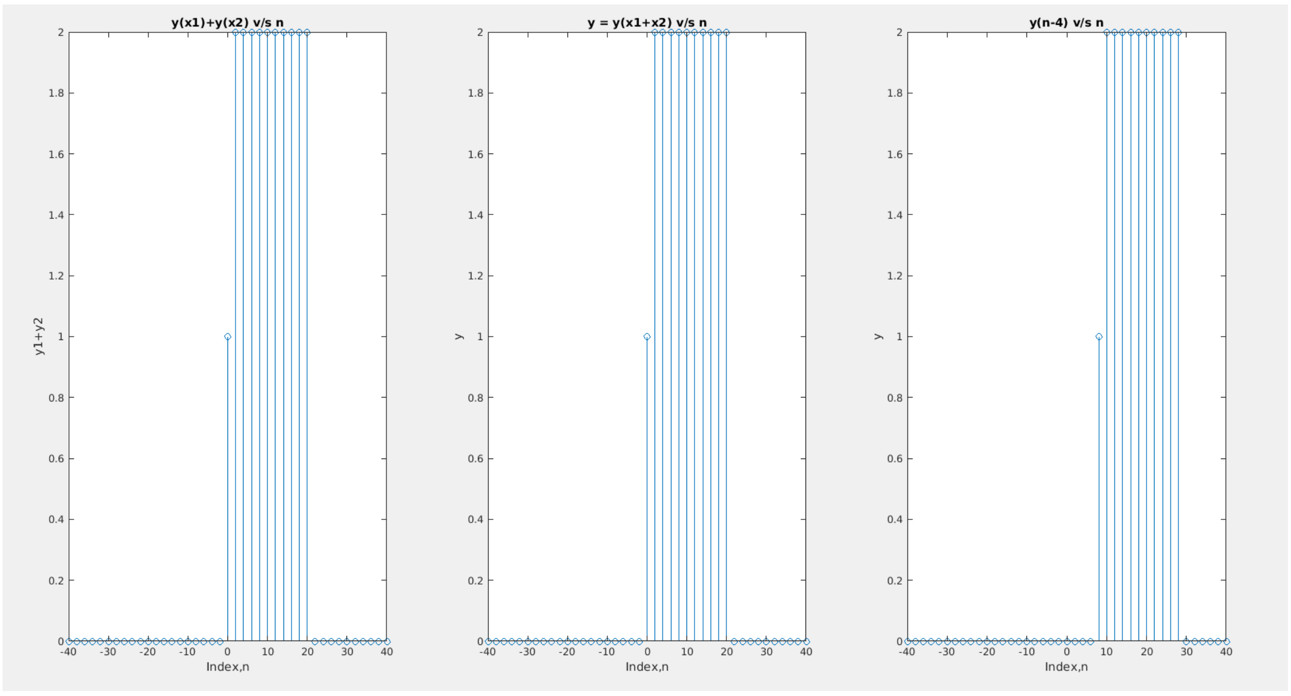
```matlab
x2 = 1*double(n>0 & n<=10);


%Checking linearity and time invarience

x=x1+x2;
xt = circshift(x,4); % 4 is arbitary number to check time invarience


% output functions
y  = sin(x) ;
yt = sin(xt);
y1 = sin(x1);
y2 = sin(x2);


subplot(1,3,1);
stem(n,y1+y2);
xlabel('Index,n');
ylabel('y1+y2');
title('y(x1)+y(x2) v/s n');

subplot(1,3,2);
stem(n,y);
xlabel('Index,n');
ylabel('y');
title('y = y(x1+x2) v/s n');


subplot(1,3,3);
stem(n,yt);
xlabel('Index,n');
ylabel('y');
title('y(n-4) v/s n');
```



% d)

```
% d is linear and not time invarient
clear;
n=-20:20;

%Defining x1 and x2
x1 = 1*double(n>=0 & n<=10);
x2 = 1*double(n>0 & n<=10);


%Checking linearity and time invarience

x=x1+x2;
xt = circshift(x,4); % 4 is arbitary number to check time invarience


% output functions
y  = x ;
yt = xt;
y1 = x1;
y2 = x2;
n = 2*n;

subplot(1,3,1);
stem(n,y1+y2);
xlabel('Index,n');
ylabel('y1+y2');
title('y(x1)+y(x2) v/s n');

subplot(1,3,2);
stem(n,y);
```

```
xlabel('Index,n');
ylabel('y');
title('y = y(x1+x2) v/s n');


subplot(1,3,3);
stem(n,yt);
xlabel('Index,n');
ylabel('y');
title('y(n-4) v/s n');
```

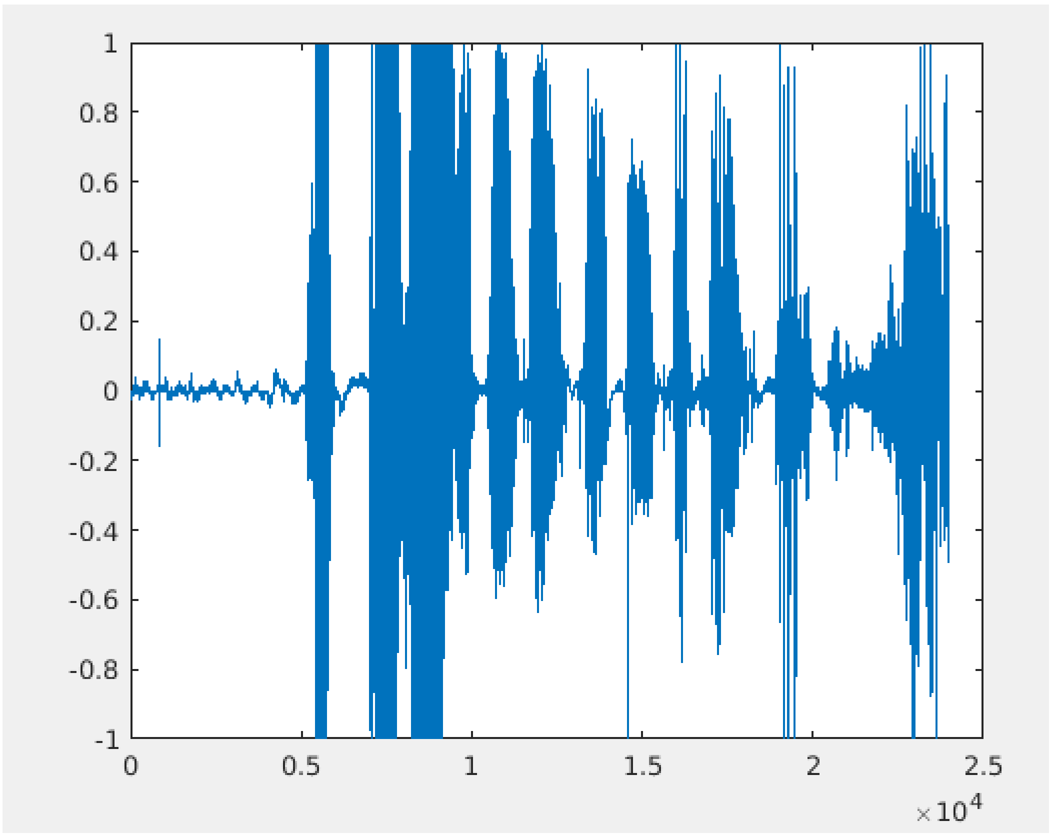


% Q8

8. Find out what \audiorecorder" and \audioplayer" functions of MATLAB do. Use these functions to record and play the audio activity of your surroundings.
Ans:

```
% Record your voice for 3 seconds.
recObj = audiorecorder;
disp('Start speaking.')
recordblocking(recObj,3)
```

```
disp('End of Recording.');

% Play back the recording.
play(recObj);

% Store data in double-precision array.
myRecording = getaudiodata(recObj);
% Plot the waveform.
plot(myRecording);
```



% Q9

9. Generate and play the following signals as audio
(a) sine wave of length 2 seconds with 500 Hz frequency and sampled at 22100 Hz.
(b) chirp signal using Matlab's chirp function - find out what exact various parameters of the chirp function has
(c) dual tone signal - consisting of two sine waves of two different frequencies - you are free to choose the different frequencies, but comment on what you hear as a function of the two frequencies.
Ans:

```
% a)
clear;

t = 2;  % time
sf = 22100; %sample frequency
f = 500; % frequency
```

```
n=0:t/sf:t;
x=sin(2*pi*f*n);   % defining the signal
p=audioplayer(x,sf); % making a audioplayer object with signal and samplerate
play(p); % playing the audio


% b)
clear;

t=0:0.001:30;
y=chirp(t,40,10,400,'logarithmic');
% 40 is initial frequency
% 10 is the frequency steps in which frequency will be increased
% 400 is maximum frequency
% 'logerithmic' is a parameter for particular kind of chirp
sf=1000;
p=audioplayer(y,sf);
play(p);
disp('Please be patient')


% c)

t=-1:0.0001:0.4;
z=sin(2*pi*217*t)+sin(2*pi*200*t);
f=22100;
p=audioplayer(z,f);
%play(p);
for j=1:3
for i = 1:2
pause(0.8);
play(p);
end
pause(0.65)
end


% the peaks will be the overtones of two frequencies
% peaks can be identified only if freq difference is less
% frequency of peaks = f1~f2

disp('Hello!')
```