Bangladesh University of Engineering and Technology

Department of Electrical & Electronic Engineering

Course No. : EEE 312
Course Title: Digital Signal Processing I Laboratory

Experiment No.: 02
Name of the Experiment:
Time Domain Analysis of Discrete Time Signals and Systems

Part-A: Generating Basic Sequences

Assignment 2

Date of Performance: November 30, 2022
Date of Submission: December 7, 2022

Prepared by:
Tasmin Khan
Student ID: 1906055

Partner: Tasnimun Razin
Student ID: 1906044

Section: A2
Level:3, Term:1

Problem 1

Question:

1. Consider the following signals and answer the questions using plots below:

$y_1(n) = 4u(n-2) - 4u(n+3)$	$y_3(n) = \sum_{k=0}^{5} 2^{-k}\delta(n-k)$
$y_2(n) = r(n+2) - r(n) + u(n-2)$	$y_4(n) = \{2, 0, 3, -1, 0.5, -1\}$ (↑ at 3rd sample)
$s_1(n) =$ *Your Voice Signal*	$s_2(n) =$ *Your Partner's Voice Signal*

MATLAB Code for Function declaration:

Function for Signal Addition:

```
function [z,n]=sigadd(x1,n1,x2,n2)
n=min(n1(1),n2(1)):max(n1(end),n2(end));
z1=zeros(1,length(n));
z2=z1;
z1((n>=n1(1)) & (n<=n1(end)))=x1;
z2((n>=n2(1)) & (n<=n2(end)))=x2;
z=z1+z2;
end
```

Function for Signal Subtraction:

```
function [z,n]=sigsub(x1,n1,x2,n2)
n=min(n1(1),n2(1)):max(n1(end),n2(end));
z1=zeros(1,length(n));
z2=z1;
z1((n>=n1(1)) & (n<=n1(end)))=x1;
z2((n>=n2(1)) & (n<=n2(end)))=x2;
z=z1-z2;
end
```

Function for Signal Fold:

```
function [z,n]=sigflold(x,n)
z=fliplr(x);
n=-fliplr(n);
end
```

Function for Signal shift:

```
function [ys,ns]= sigshift(y,n,n0)
ns=n+n0;
ys=y;
end
```

Function for Signal upsampling:

```
function [zup,nup] = sigupsample(x,n,L)
       nup=n(1)*L:n(end)*L;
       zup=zeros(1,length(nup));
       for i=1:length(x)
           zup(i+((i-1)*(L-1)))=x(i);
       end
end
```

MATLAB main code:

```
clc
clear all
close all

%% basic sequence generation
u=@(n,n0) [(n-n0)>=0];
r=@(n,n0) ((n-n0).*((n-n0)>=0));
del=@(n,n0) [(n-n0)==0];
n=-10:10;

%% function generation

%y1 function values
y1=4*u(n,2)-4*u(n,-3);

%y2 function values
y2=r(n,-2)-r(n,0)+u(n,2);

%y3 function values
y3=zeros(1,length(n));
for k=0:5
    y3=y3+(2^(-k)*del(n,k));
end

%y4 function values
y4=[2 0 3 -1 .5 -1];
n4=-3:2;

%% S1 and S2 values
[s1dual,Fs1]=audioread('DSP audio signal.mp3');
[s2dual,Fs2]=audioread('DSP audio signal2.aac');

%Dual channel to single channel conversion
s1(1,:)=s1dual(:,1);
s2(1,:)=s2dual(:,1);
n1=1:length(s1);
n2=1:length(s2);

%%
sound(s1dual,Fs1)

%%
sound(s2dual,Fs2)
```

Problem (a):

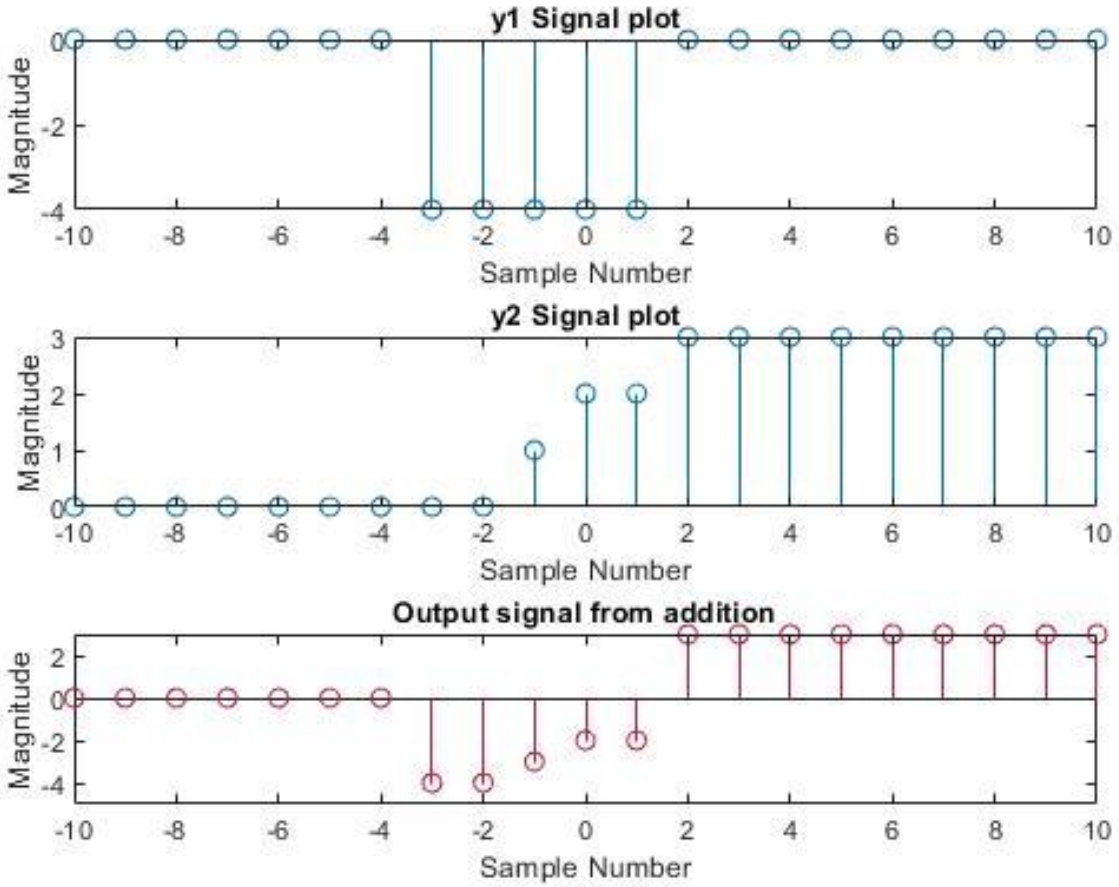
Question:

a) $y_1(n) + y_2(n) = ?$

MATLAB code:

```
%% problem a
[ya,n]=sigadd(y1,n,y2,n);

%plotting
figure(1)

subplot(3,1,1)
stem(n,y1);
title('y1 Signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(n,y2);
title('y2 Signal plot')
xlabel('Sample Number')
ylabel('Magnitude')
subplot(3,1,3)

stem(n,ya,'Color','#C41E3A');
title('Output signal from addition')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

This was a problem for adding a step signal and a ramp signal.

A function “sigadd” was created at first.
The input signals were generated too, following the conditions of unit step sequence and ramp sequence.
Then the input signals and their corresponding indices were passed through the function for generating a variable “ya”, which was the output signal.
All the index ranges of inputs and output were within our predefined range of [-10:10]; so, padding was not actually needed.

Problem (b):

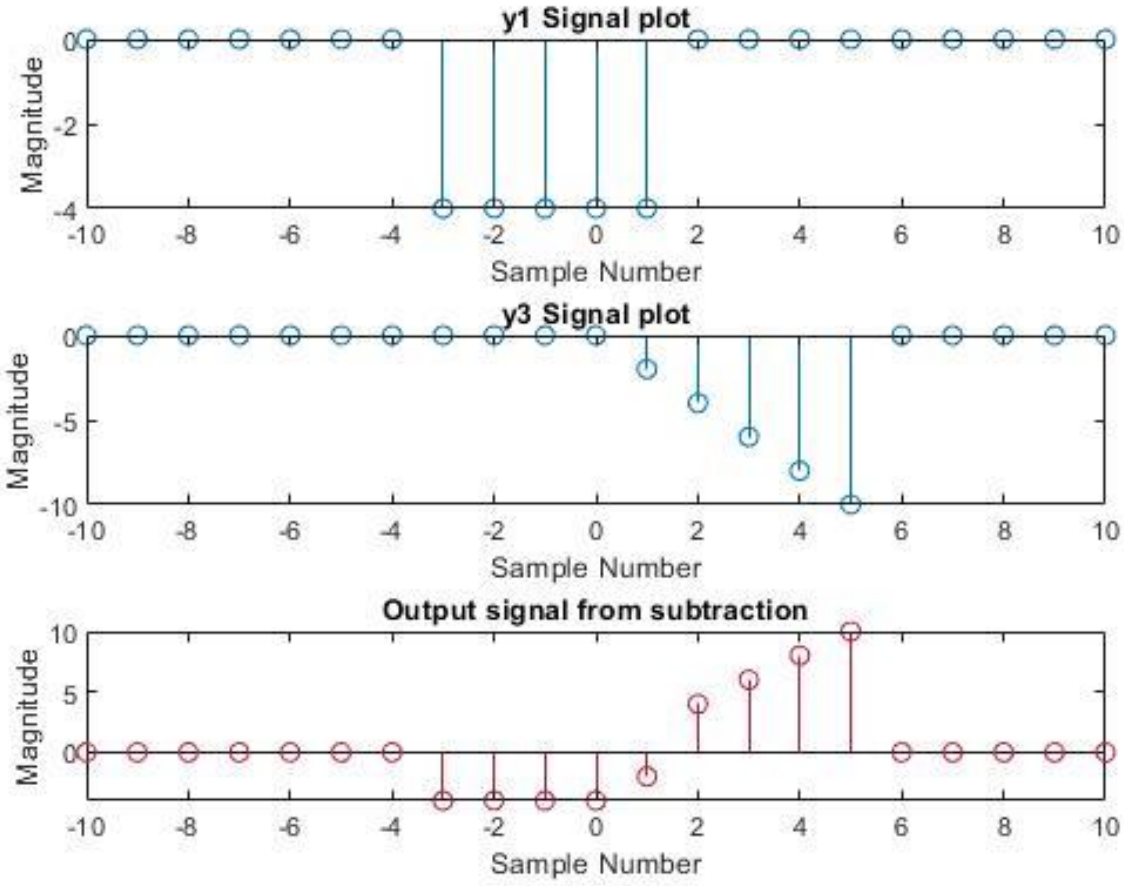
Question:

b) $y_1(n) - y_3(n) = ?$

MATLAB code:

```
%% problem b
yb=sigsub(y1,n,y3,n);

%plotting
figure(2)

subplot(3,1,1)
stem(n,y1);
title('y1 Signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(n,y3);
title('y3 Signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,3)
stem(n,yb,'Color','#C41E3A');
title('Output signal from subtraction')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

This was a problem for subtracting an impulse signal from a step signal.

A function “sigsub” was created at first.
The input signals were generated too, following the conditions of unit step sequence and unit impulse sequence.
Then the input signals and their corresponding indices were passed through the function generating a variable “yb”, which was the output signal.
All the index ranges of inputs and output were within our predefined range of [-10:10]; so, padding was not actually needed.

Problem (c):

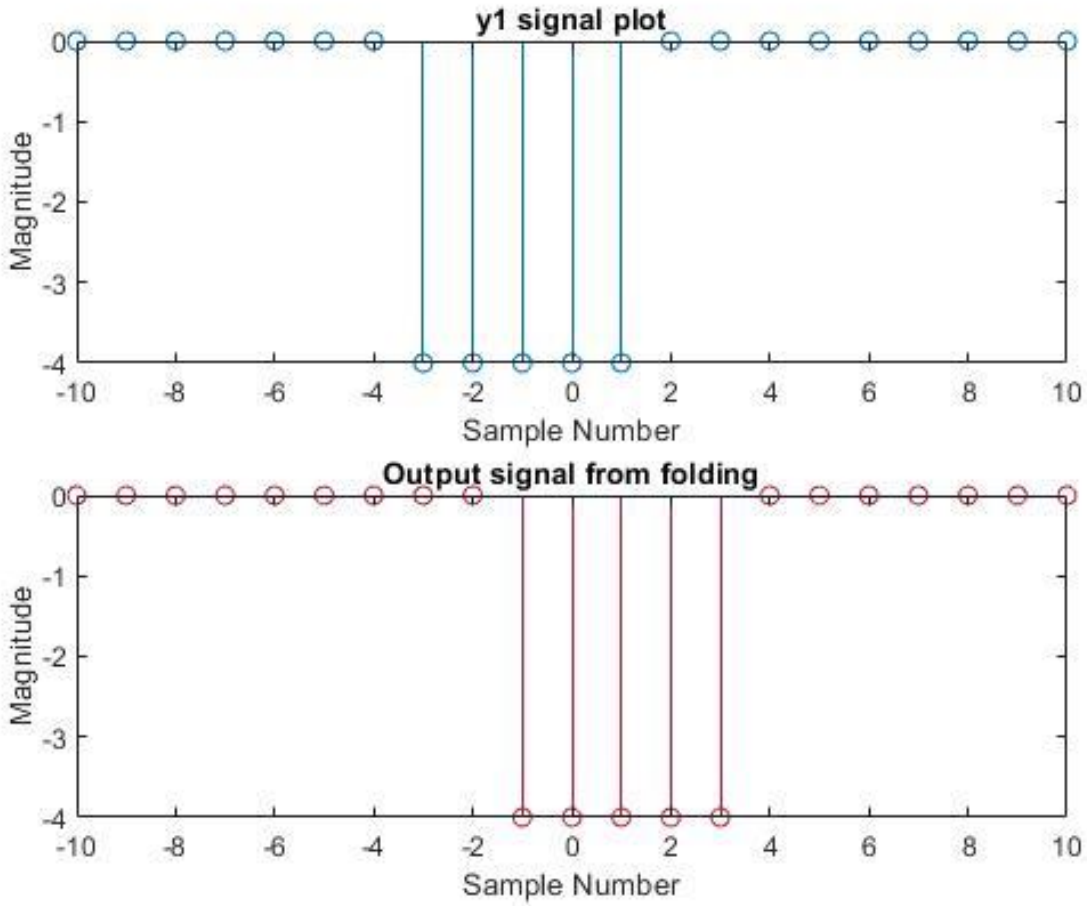
Question:

c) $y_1(-n) = ?$

MATLAB code:

```
%% problem c
[yc,nc]=sigfold(y1,n);

%plotting
figure(3)

subplot(2,1,1)
stem(n,y1);
title('y1 signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(2,1,2)
stem(nc,yc,'Color','#C41E3A');
title('Output signal from folding')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

This was a problem for folding i.e. reflecting a signal with respect to y-axis.

A function “sigfold” was created at first.
The input was a unit step signal, which were flipped on its x-axis as well as index, adjusting signs too.
Then the input signal and its index array were passed through variables “[yc,nc]”, which generated the output signal.
The index ranges of both input and output were within our predefined range of [-10:10] which is a symmetric index. Hence, in the plot we saw the same scaling for x-axis.

Problem (d):

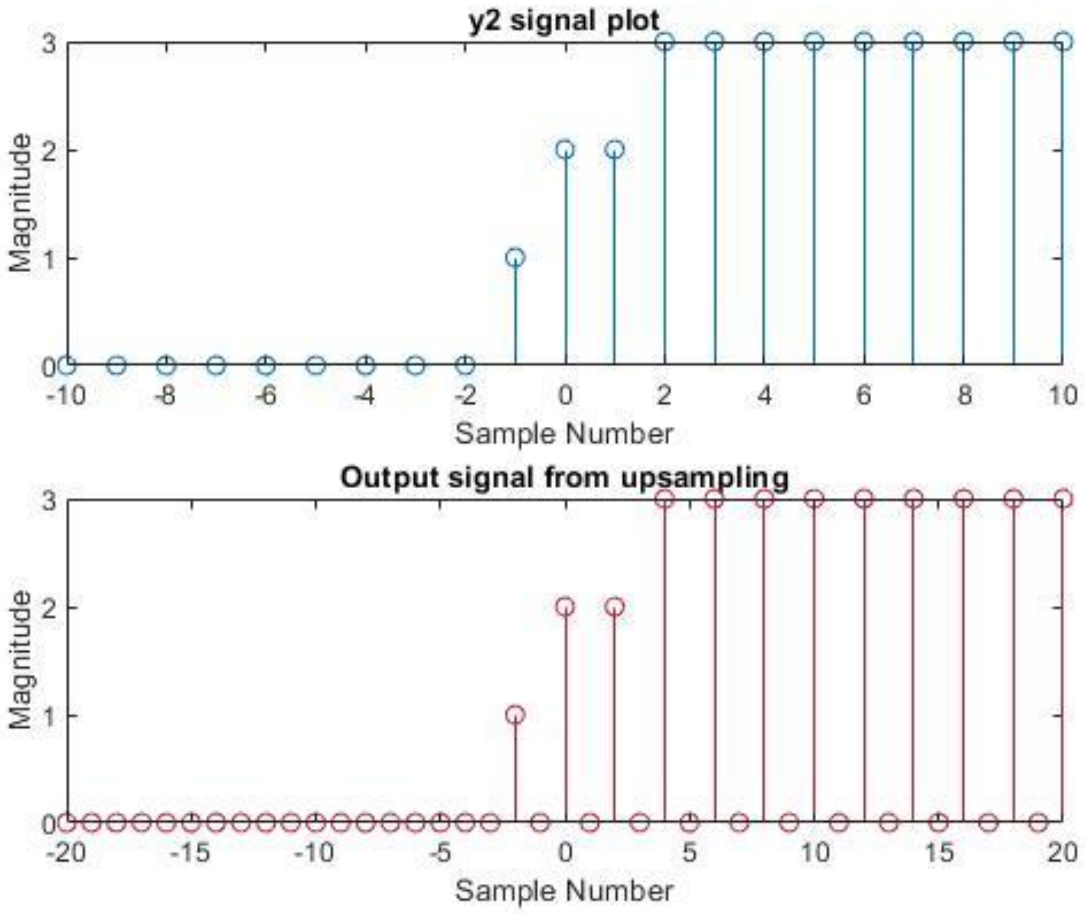
Question:

d) $y_2\left(\frac{n}{2}\right) = ?$

MATLAB code:

```
%% problem d
[yd,nd]=sigupsample(y2,n,2);

%plotting
figure(4)

subplot(2,1,1)
stem(n,y2);
title('y2 signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(2,1,2)
stem(nd,yd,'Color','#C41E3A');
title('Output signal from upsampling')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

This was a problem for up-sampling i.e. increasing the sampling rate of a signal.

A function “sigupsample” was created at first.

The input was a ramp signal, whose index range along with up-sampling factor was passed through variables “[yd,nd]” to generate the output signal of required length.
The indices of the input signal got factorized in the output, and therefore, some extra sample points were inserted of the value of zero. For a upsampling factor of L, (L-1) values were added between every two samples in the original signal.

The resulting waveshape does not differ from the original one; but the higher sampling rate increases the quality of the signal.

Problem (e):

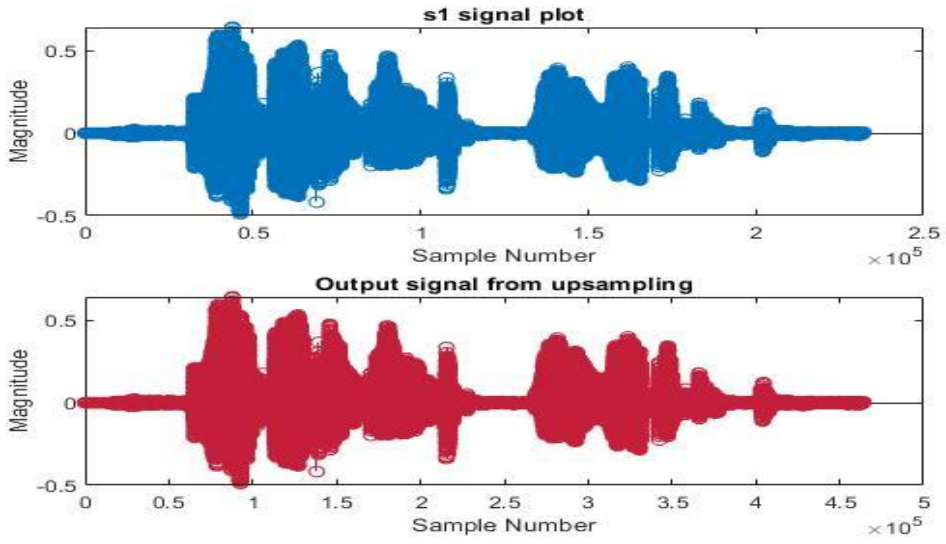
Question:

e) $s_1\left(\frac{n}{2}\right) = ?$

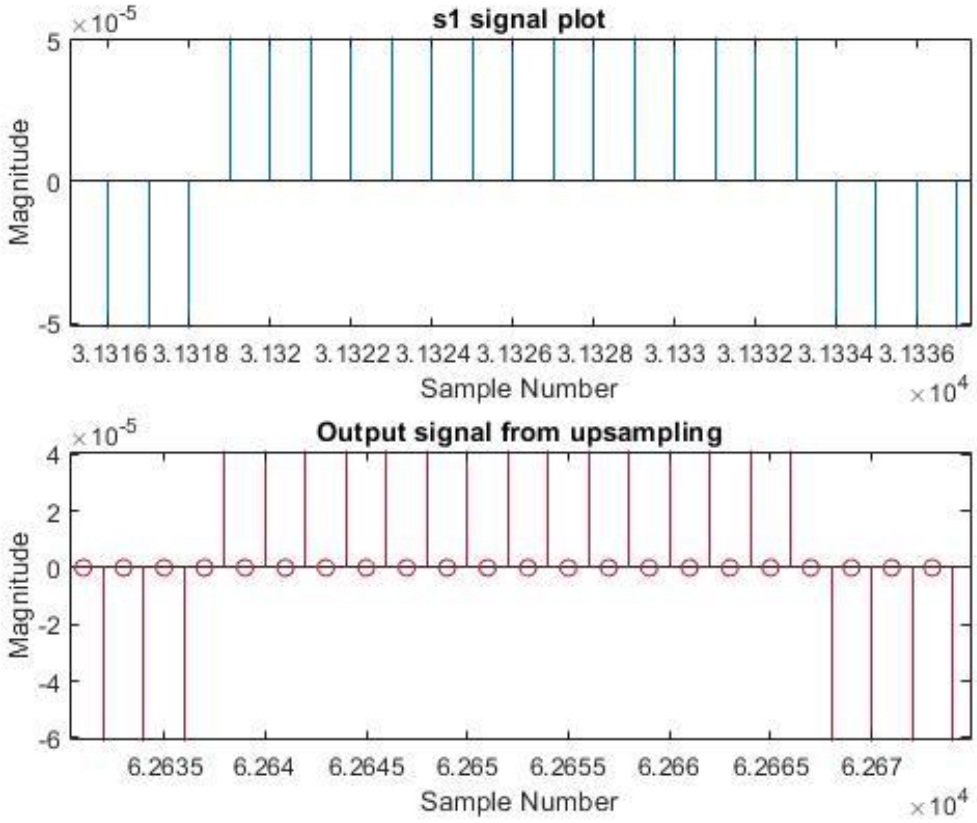
MATLAB code:

```
%% problem e

[s1e,ne]=sigupsample(s1,n1,2);

%plotting
figure(5)

subplot(2,1,1)
stem(n1,s1);
title('s1 signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(2,1,2)
stem(ne,s1e,'Color','#C41E3A');
title('Output signal from upsampling')
xlabel('Sample Number')
ylabel('Magnitude')


%%
sound(s1e,Fs1)
```

Plot:



➔ **Zoomed in:**



Discussion:

This was a problem for up-sampling i.e. increasing the sampling rate of an audio signal.
Like the previous problem, the operation was done through the function "sigupsample".
The input was a voice record, which was sampled using the "audioread" function. The "audioread" function generated a 2D array for dual audio channel with a specific sampling frequency. For our sample, the right and left stereo input was very similar and it did not make much of a difference working with any single channel. So, the input signal was extracted to be a 1D array at first. Then up-sampling was performed by the aforementioned process.
As expected, the resulting audio did not sound different from the original one; but the higher sampling rate increased the quality of the audio.

Hearing the sound at previous sampling rate Fs, it was slower. Since we upsampled the input audio using the factor 2, the sampling rate increased. So, we could hear the original audio by playing at a sampling rate of Fs*L.

Problem (f):

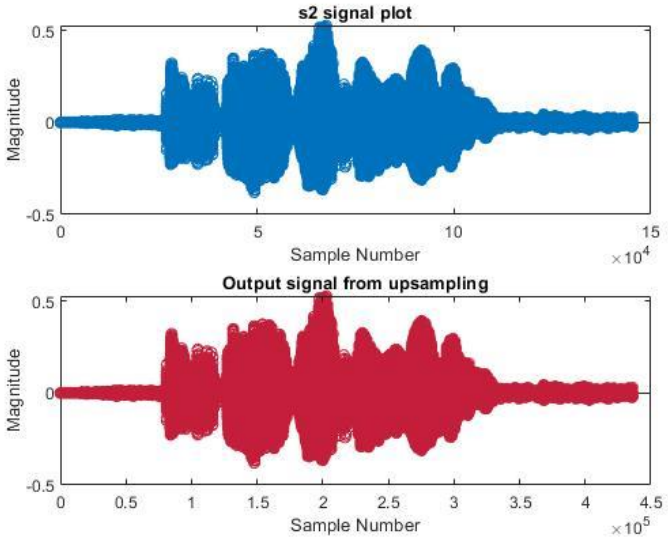
Question:

f) $s_2\left(\frac{n}{3}\right) = ?$

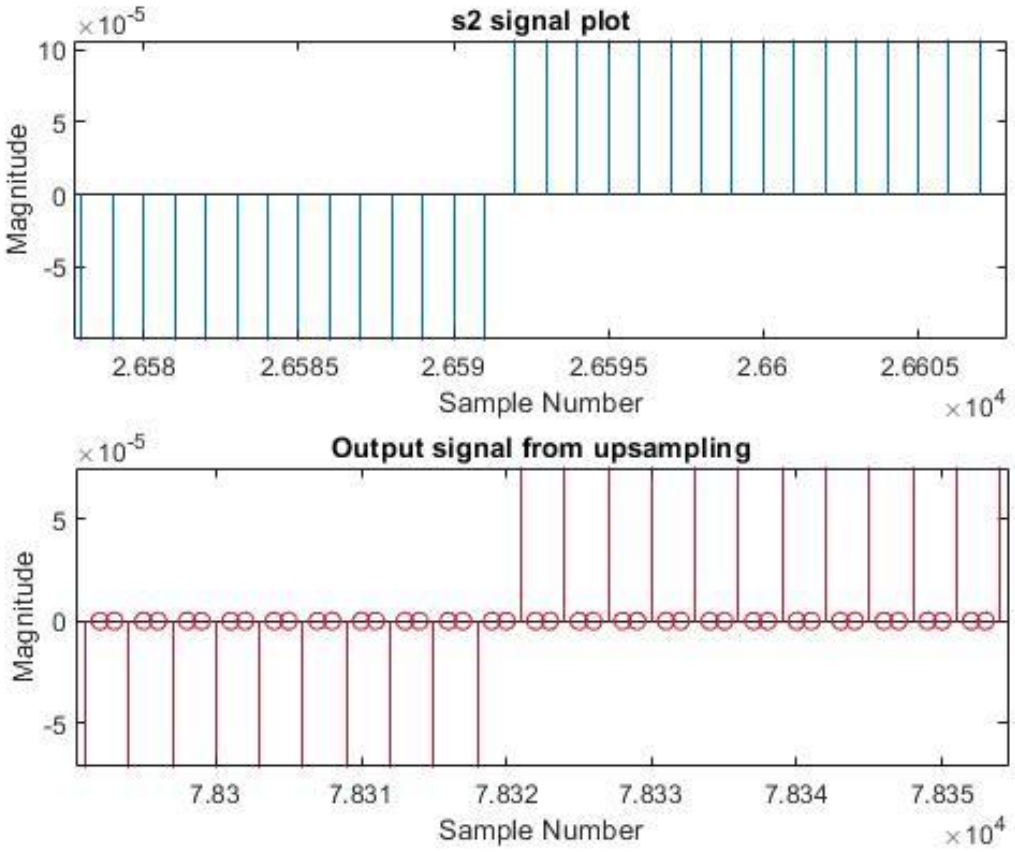
MATLAB code:

```
%% problem f

[s2f,nf]=sigupsample(s2,n2,3);

%plotting
figure(6)

subplot(2,1,1)
stem(n2,s2);
title('s2 signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(2,1,2)
stem(nf,s2f,'Color','#C41E3A');
title('Output signal from upsampling')
xlabel('Sample Number')
ylabel('Magnitude')

%%
sound(s2f,Fs2)
```

Plot:



➔ **Zoomed in:**



Discussion:

This was a similar problem for up-sampling another audio signal.
The only exception was the up-sampling factor was increased to 3. Sampling and converting the audio signal followed the same procedure as before.

Hearing the sound at previous sampling rate Fs, it was slower. Since we upsampled the input audio using the factor 3, the sampling rate increased. So, we could hear the original audio by playing at a sampling rate of Fs*L. The signal quality was improved but it did not make the audio sound any significantly better.

.

Problem (g):

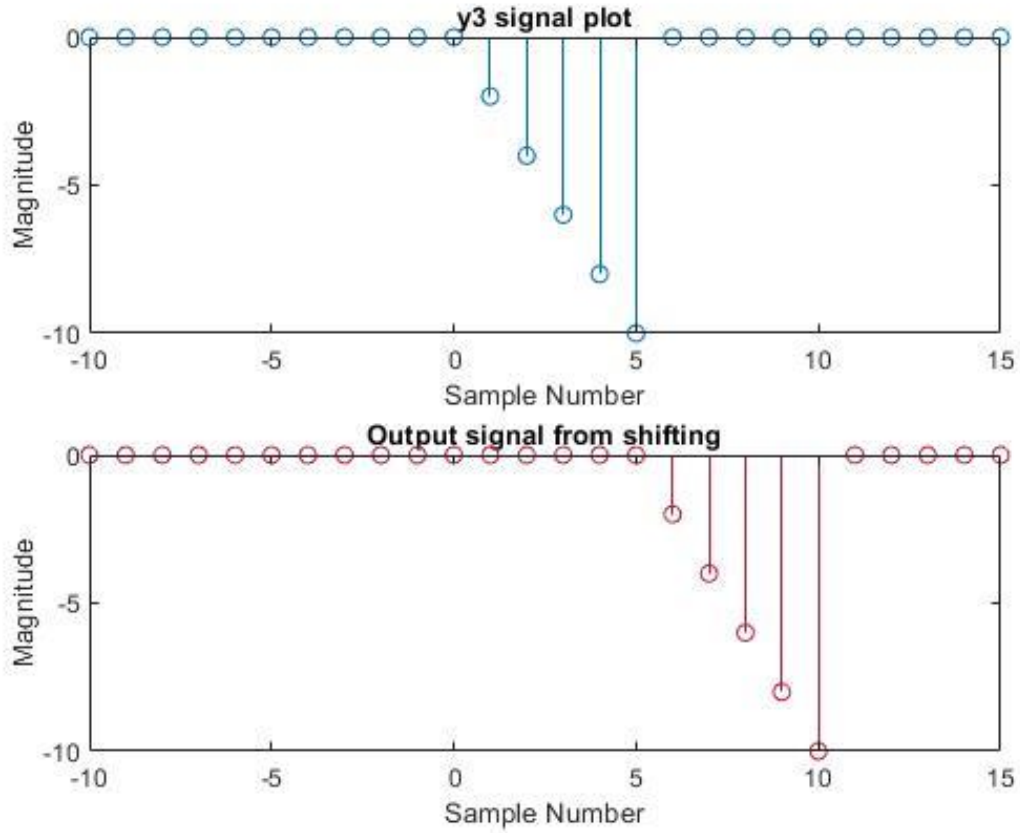
Question:

g) $y_3(n-5) = ?$

MATLAB code:

```
%% problem g
[yg,ng]=sigshift(y3,n,5);

%padding
nplotg=min(ng(1),n(1)):max(ng(end),n(end));
y3plotg=zeros(1,length(nplotg));
ygplotg=y3plotg;
y3plotg(find((nplotg>=n(1))&(nplotg<=n(end))))=y3;
ygplotg(find((nplotg>=ng(1))&(nplotg<=ng(end))))=yg;

%plotting
figure(7)

subplot(2,1,1)
stem(nplotg,y3plotg);
title('y3 signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(2,1,2)
stem(nplotg,ygplotg,'Color','#C41E3A');
title('Output signal from shifting')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

This was a problem for shifting a signal to right.

A function “sigshift” was created at first.

To resemble the index ranges of the output to the input, padding was done before plotting, covering the whole range.

Problem (h):

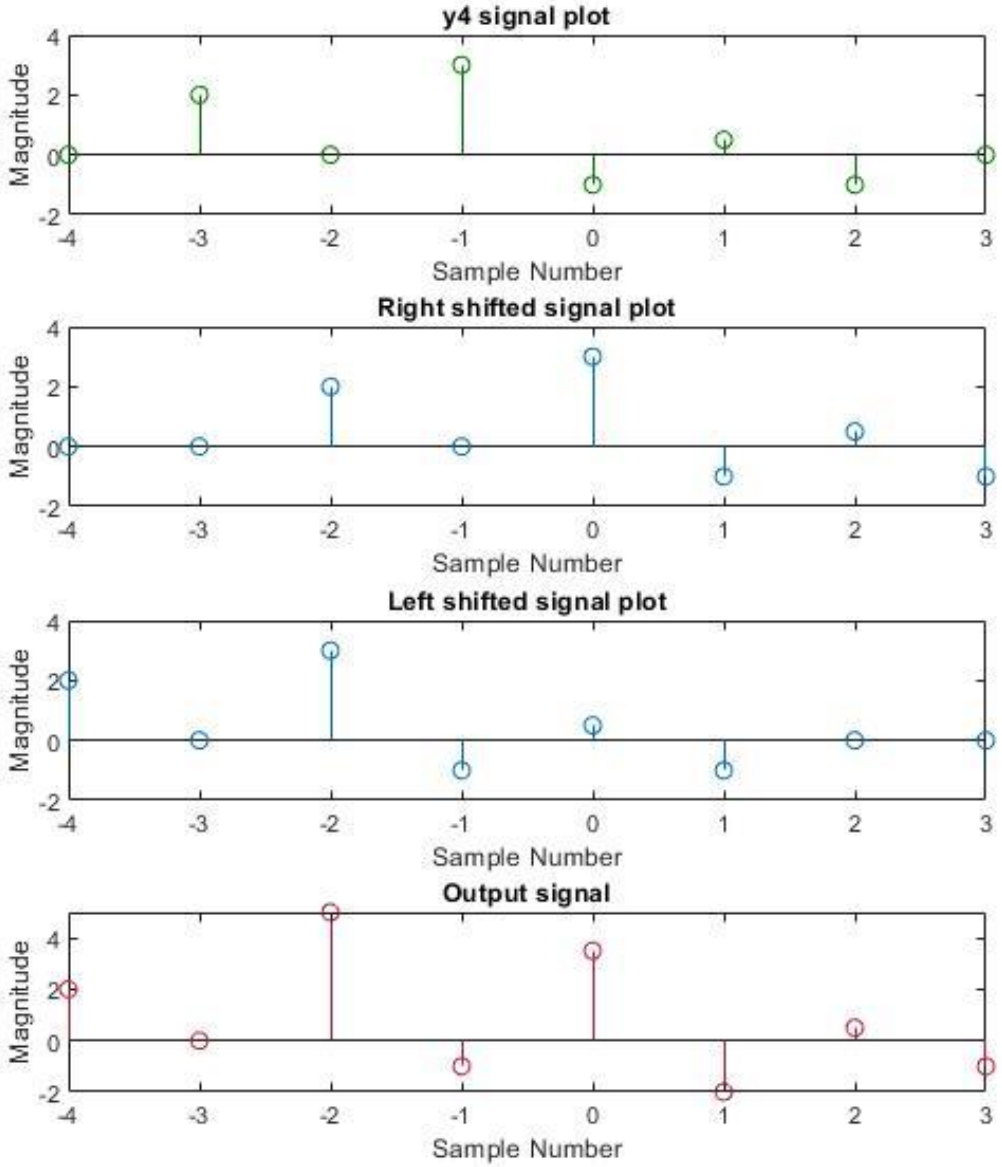
Question:

h) $y_4(n-1) + y_4(n+1) = ?$

MATLAB code:

```
%% problem h
[yh1,nh1]=sigshift(y4,n4,1);
[yh2,nh2]=sigshift(y4,n4,-1);
[yh,nh]=sigadd(yh1,nh1,yh2,nh2);

%padding
n4ploth=min([n4(1),nh1(1),nh2(1)]):max([n4(end),nh1(end),nh2(end)])
y4ploth=zeros(1,length(n4ploth));
yh1ploth=y4ploth;
yh2ploth=y4ploth;
yhploth=y4ploth;
y4ploth(find((n4ploth>=n4(1))&(n4ploth<=n4(end))))=y4;
yh1ploth(find((n4ploth>=nh1(1))&(n4ploth<=nh1(end))))=yh1;
yh2ploth(find((n4ploth>=nh2(1))&(n4ploth<=nh2(end))))=yh2;
yhploth(find((n4ploth>=nh(1))&(n4ploth<=nh(end))))=yh;

%plotting
figure(8)

subplot(4,1,1)
stem(n4ploth,y4ploth,'Color','#008000');
title('y4 signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(4,1,2)
stem(n4ploth,yh1ploth);
title('Right shifted signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(4,1,3)
stem(n4ploth,yh2ploth);
title('Left shifted signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(4,1,4)
stem(n4ploth,yhploth,'Color','#C41E3A');
title('Output signal')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

This was a problem for shifting a signal to both right and left, then adding the two.

Like the previous problem, the operation was done using the function “sigshift”; and to resemble the index ranges of all the outputs to the input in the graph, padding was done before plotting, covering the whole range.

Problem (i):

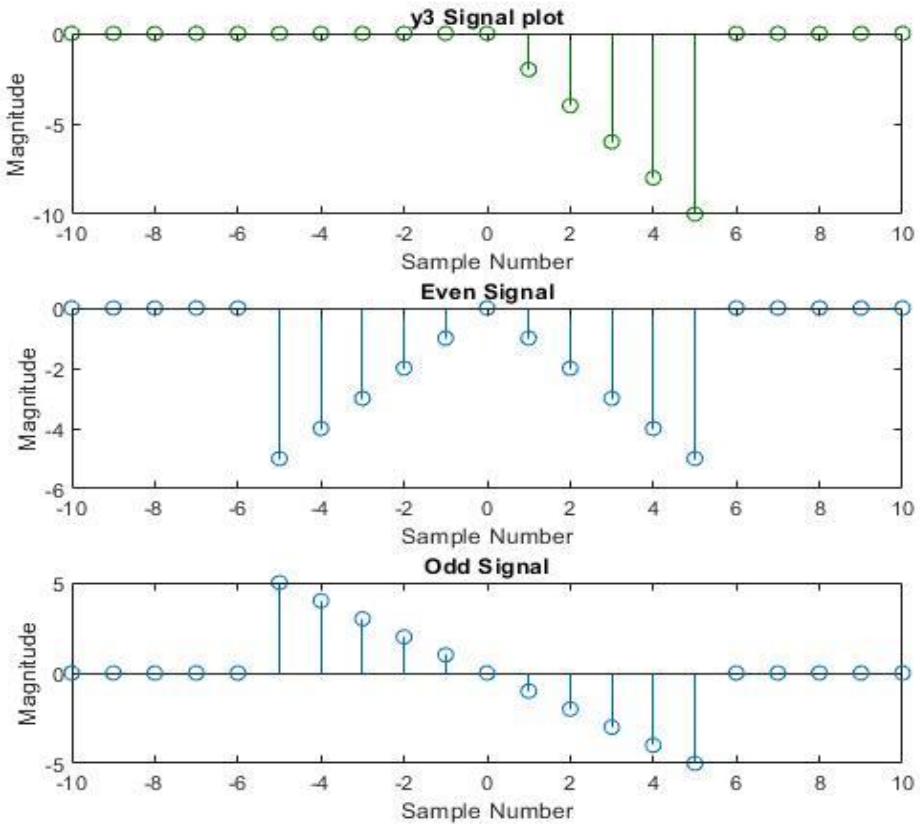
Question:

i) Odd and even part of $y_3(n)$

MATLAB code:

```
%% problem i
[yifold,ni]=sigfold(y3,n);
[yieven,nploti]=sigadd(y3,n,yifold,ni);
[yiodd,nploti]=sigsub(y3,n,yifold,ni);
yieven=yieven/2;
yiodd=yiodd/2;

%padding
y3ploti=zeros(1,length(nploti));
y3ploti(find((nploti>=n(1))&(nploti<=n(end))))=y3;

%plotting
figure(9)

subplot(3,1,1)
stem(nploti,y3ploti,'Color','#008000');
title('y3 Signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(nploti,yieven)
title('Even Signal')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,3)
stem(nploti,yiodd)
title('Odd Signal')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

This is a problem of extracting the even and the odd part of a signal.

Even part of a signal is symmetric with respect to the y-axis i.e. its graph remains unchanged after reflection about the y-axis.
Odd part of a signal is symmetric with respect to the origin i.e. its graph remains unchanged after rotation of 180° about the origin.

So, at first, the input signal was folded. Then, even part was generated by adding the folded part with the original part; odd part was generated by subtracting the folded part from the original part.

Now, adding the even and the odd part should reconstruct the original signal; so, the amplitude of both the signals were halved.

To resemble the index ranges of all the outputs to the input, padding was done before plotting, covering the whole range.

Problem (j):

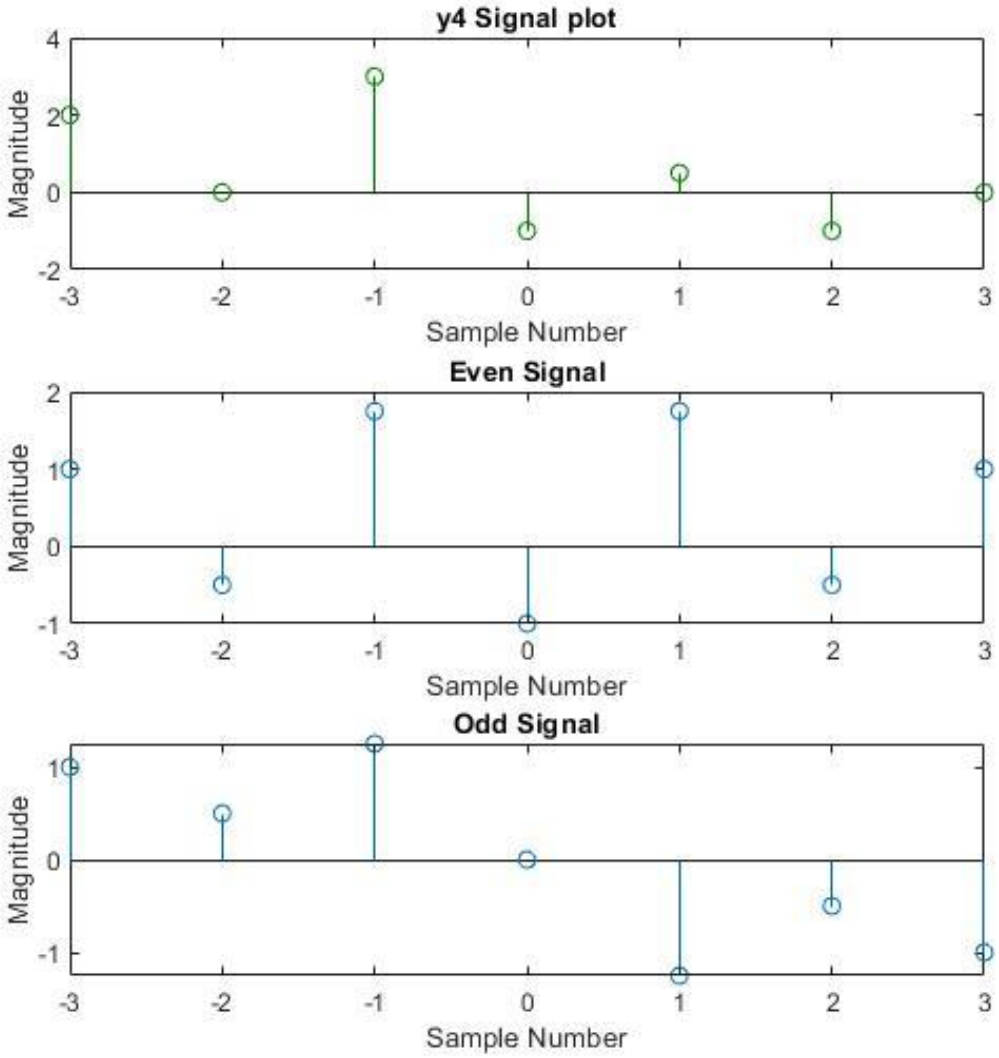
Question:

j) Odd and even part of $y_4(n)$

MATLAB code:

```
%% problem j
[yjfold,nj]=sigfold(y4,n4);
[yjeven,n4plotj]=sigadd(y4,n4,yjfold,nj);
[yjodd,n4plotj]=sigsub(y4,n4,yjfold,nj);
yjeven=yjeven/2;
yjodd=yjodd/2;

%padding
y4plotj=zeros(1,length(n4plotj));
y4plotj(find((n4plotj>=n4(1))&(n4plotj<=n4(end))))=y4;

%plotting
figure(10)

subplot(3,1,1)
stem(n4plotj,y4plotj,'Color','#008000');
title('y4 Signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(n4plotj,yjeven)
title('Even Signal')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,3)
stem(n4plotj,yjodd)
title('Odd Signal')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

This is a problem of extracting the even and the odd part of a signal.

The whole process was similar as that followed in the previous problem.

Problem (k):

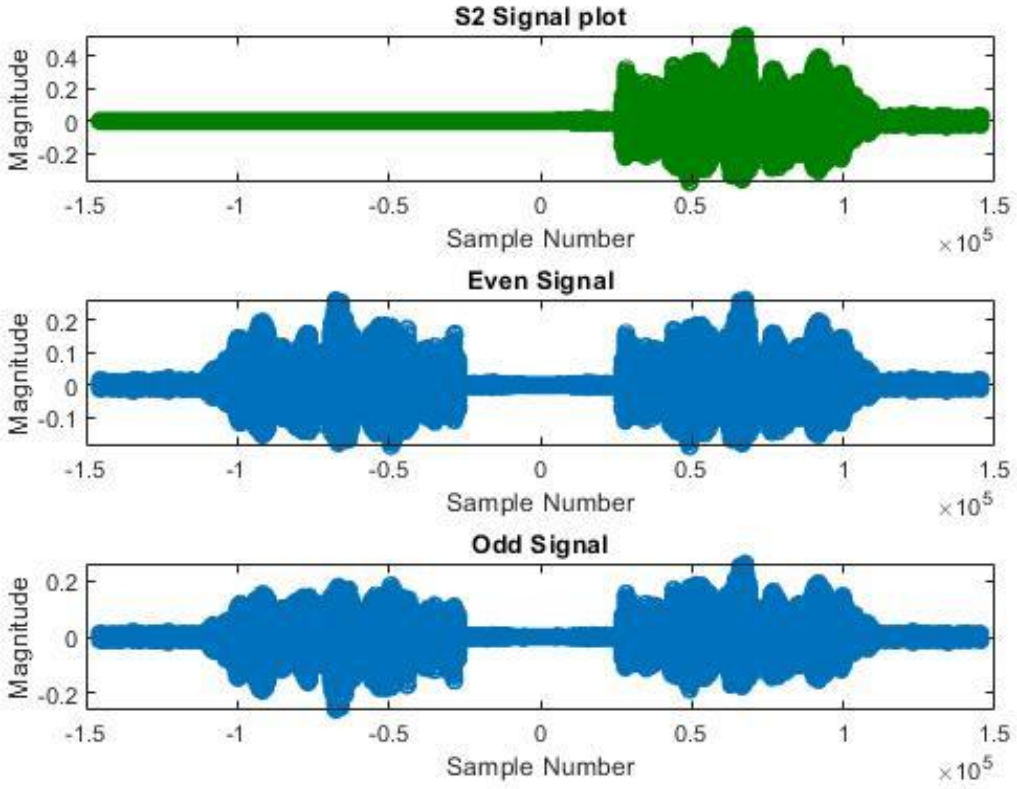
Question:

k) Odd and even part of $s_2(n)$

MATLAB code:

```
%% problem k
[s2kfold,nk]=sigfold(s2,n2);
[s2keven,n2plotk]=sigadd(s2,n2,s2kfold,nk);
[s2kodd,n2plotk]=sigsub(s2,n2,s2kfold,nk);
s2keven=s2keven/2;
s2kodd=s2kodd/2;

%padding
s2plotk=zeros(1,length(n2plotk));
s2plotk(find((n2plotk>=n2(1))&(n2plotk<=n2(end))))=s2;

%plotting
figure(11)

subplot(3,1,1)
stem(n2plotk,s2plotk,'Color','#008000');
title('S2 Signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(n2plotk,s2keven)
title('Even Signal')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,3)
stem(n2plotk,s2kodd)
title('Odd Signal')
xlabel('Sample Number')
ylabel('Magnitude')


%%
sound(s2keven,Fs2)
%%
sound(s2kodd,Fs2)
```

Plot:



Discussion:

This is a problem of extracting the even and the odd part of an audio signal.

The whole process was similar as that followed in the previous problem.

While hearing the sounds, the intensity of the sound for both the even and the odd parts was half the original sound, since the amplitudes were halved.

To test the quality of the sound, the positive parts for both the output signals sounded same as the original one.
However, in the negative part, the whole audio was reversed, so the sound was completely distorted.
The odd part having an additional phase shift of 180°, sounded more distorted that the even part with no phase shift.

So, for half the period, we heard a distorted sound with robotic screeches; for the rest half, we heard the original sound with less intensity, for both even and odd parts.

Problem (l):

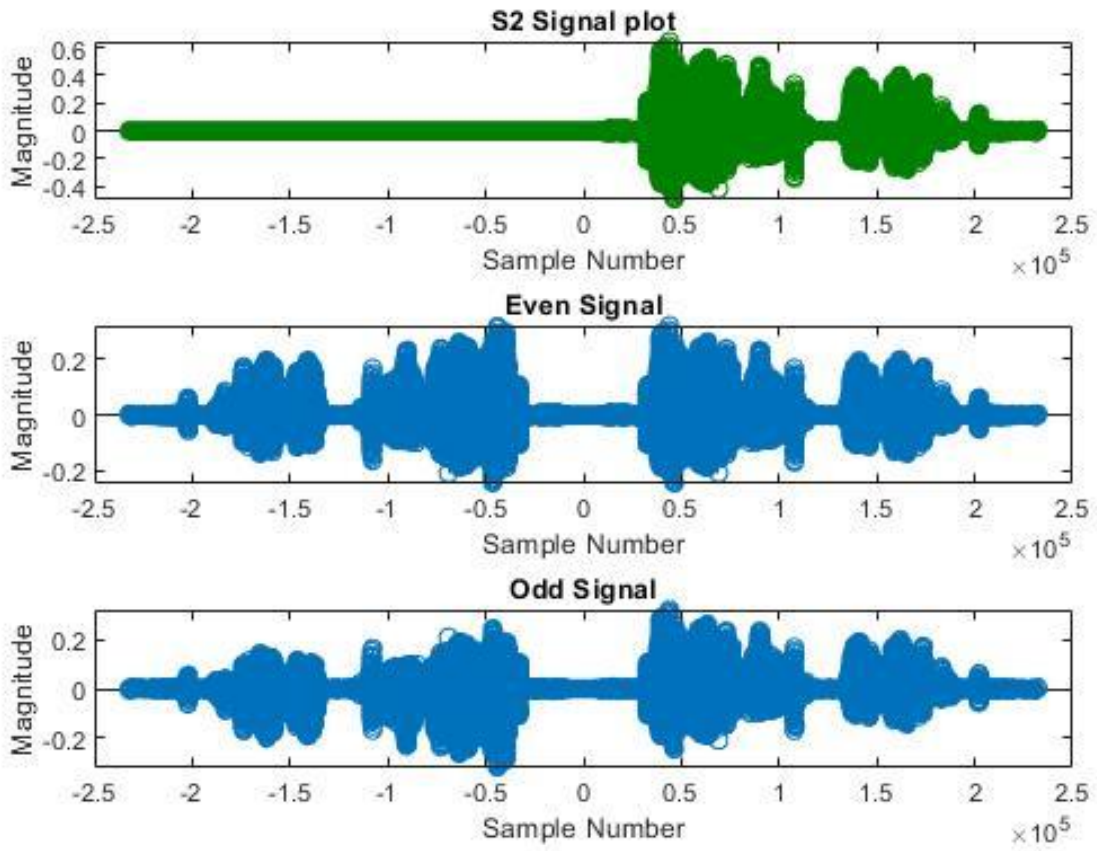
Question:

l) Odd and even part of $s_1(n)$

MATLAB code:

```
%% problem l

[s1lfold,nl]=sigfold(s1,n1);
[s1leven,n1plotl]=sigadd(s1,n1,s1lfold,nl);
[s1lodd,n1plotl]=sigsub(s1,n1,s1lfold,nl);
s1leven=s1leven/2;
s1lodd=s1lodd/2;

%padding
s1plotl=zeros(1,length(n1plotl));
s1plotl(find((n1plotl>=n1(1))&(n1plotl<=n1(end))))=s1;

%plotting
figure(12)

subplot(3,1,1)
stem(n1plotl,s1plotl,'Color','#008000');
title('S2 Signal plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(n1plotl,s1leven)
title('Even Signal')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,3)
stem(n1plotl,s1lodd)
title('Odd Signal')
xlabel('Sample Number')
ylabel('Magnitude')


%%
sound(s1leven,Fs1)
%%
sound(s1lodd,Fs1)
```

Plot:



Discussion:

This is a problem of extracting the even and the odd part of an audio signal.

The whole process, as well as the sound quality, was similar as that followed in the previous problem.

Conclusion:

In this experiment, we generated some basic sequence signals. We also learnt operations like addition, subtraction, folding, shifting and upsampling for discrete signal values. The technique of padding signal indices was very much useful to do the operations as well as to present the plots in a manner that made them visually quick to understand.

We also used real life audio signals for the experiment. Thus, the experiment can be deemed successful for learning versatile signal operations.