Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN3551- Digital Signal Processing

Assignment 1

Ranasingha A.S.N. 200507N

1. Harmonics Detection

First the signal is loaded as **"xn_test"** and the sampling frequency (Fs) is defined. Here Fs is 128 Hz.

```
%loading the corrupted signal
load('signal507.mat','xn_test');

Fs = 128; % sampling frequency
```

Five subsets named S1, S2, S3, S4 and S5 are formed using the original signal.

```
%Creating subsets
S1 = xn_test(1:128);
S2 = xn_test(1:256);
S3 = xn_test(1:512);
S4 = xn_test(1:1024);
S5 = xn_test(1:1792);

%Obtain DFT of each subset of samples
DFT_S1 = fft(S1);
DFT_S2 = fft(S2);
DFT_S3 = fft(S3);
DFT_S4 = fft(S4);
DFT_S5 = fft(S5);
```

Obtain DFT of each subset

Magnitude response of DFT of each signal is plotted against frequency to identify the harmonics. In a signal corrupted by noise, harmonics can be identified where there are spikes in the magnitude plot of DFT.

```
for i=1:5
    DFT_S = eval(['DFT_S', int2str(i)]);
    [P1, index] = findpeaks(abs(DFT_S), 'NPeaks', 8, 'SortStr', 'descend'); % Get the eight maximum peak points
    N = length(DFT_S); % number of DFT points
    f = (-N/2:N/2-1)*Fs/N; % Frequency vector
    freq = f(index);

    subplot(5,1,i);
    stem(f,abs(fftshift(DFT_S)),'MarkerSize', 3); %Plot magntude responses of DFT of each signal
    ylabel("| DFT (S"+ int2str(i) + ") |");

    %Display outputs
    disp("Subset " + int2str(i));
    fprintf('Prominent peaks :');
    disp(P1);
    fprintf('frequencies     :');
    disp(freq)

end
sgtitle('Magnitude Response of subsets');
xlabel("frequency");
ax = gca;
```

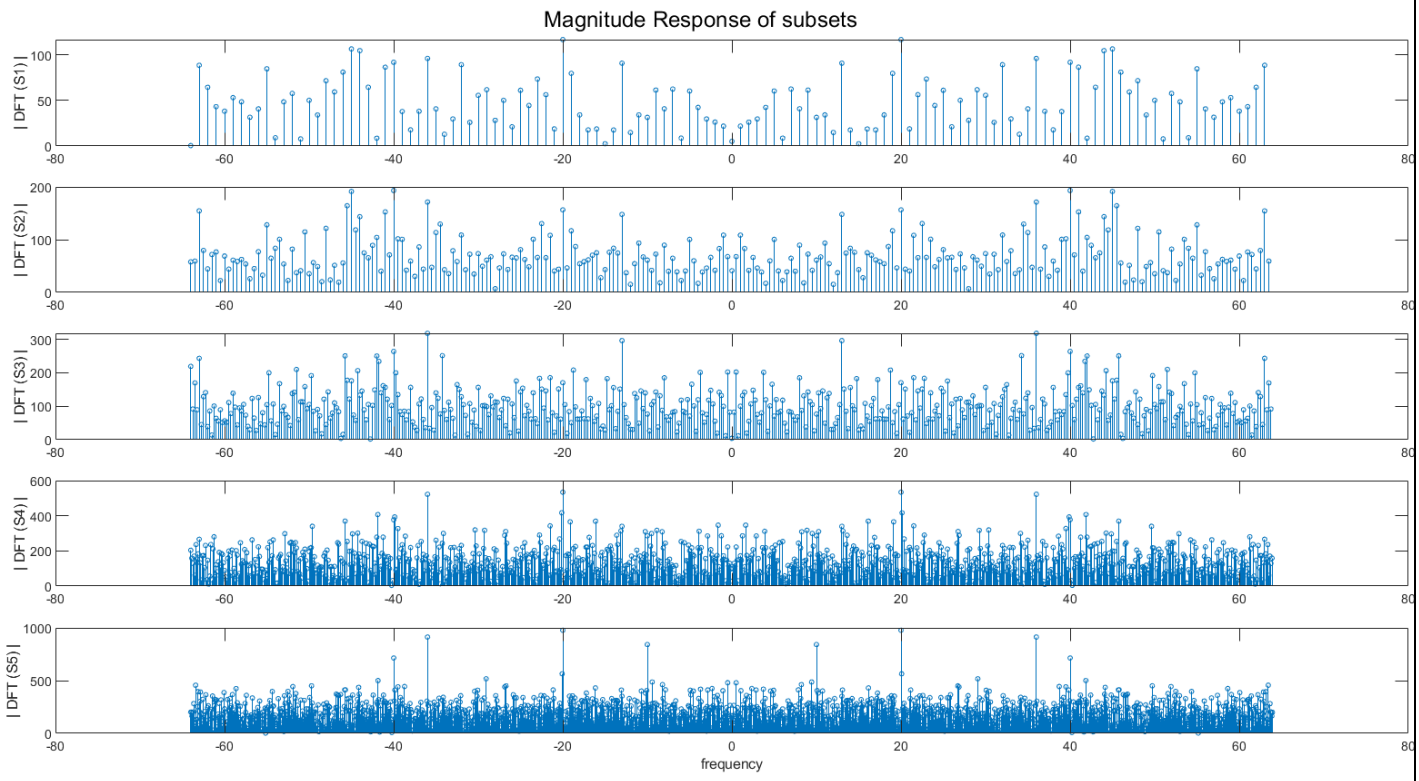
Since our time domain signals are real-valued, we get their DFTs symmetric around some mid frequency. In other terms if the discrete time domain signal is $x[n]$ then DFT of $x[n]$ *holds the relation*,

$$X[N-k] \;=\; X^*[k]$$

To get the mid frequency to the origin, we call **fftshift** function in MATLAB. It has mentioned that the original signal contains 4 harmonics. Therefore, here we are taking the first 8 maximum peak points as 4 harmonics would result in 8 spikes in magnitude plot od DFT. (Since symmetric, there are four different negative and its positive frequency pairs resulting in 8 spikes)

The magnitude response correspond to each subset is as follows.

Magnitude Responses



Observations:

- In magnitude responses of DFTs of S1 and S2, significant spikes cannot be observed clearly.
- In |DFT (S3)| plot, there are more than 8 spikes and still we cannot identify harmonics clearly.
- In |DFT (S4)| plot only two pairs of significant spikes are clearly visible.
- In the magnitude response of DFT of S5, we can clearly identify four positive negative frequency pairs having significant spikes. These frequencies correspond to the four harmonics in the original signal.

Since when forming subsets S1 to S4, we have taken only a part of whole samples of the original signal. As a result, the frequency resolution of these signals is lower than that of S5. The frequency resolution varies as follows.

$$S1 < S2 < S3 < S4 < S5$$

Signals with low frequency resolution have higher impact from noise making it unable to distinguish between harmonics and noise. S1 has the lowest resolution and therefore no significant spike can be identified in the magnitude response of its DFT.

S5 has the highest frequency resolution giving a clear image of the harmonics.

Displayed Outputs:

```
Subset 1
Prominent peaks :  116.9053  116.9053  106.6807  106.6807   96.2453   96.2453   92.0513   92.0513

frequencies     :   -20    20   -45    45   -36    36   -40    40

Subset 2
Prominent peaks :  193.2338  193.2338  191.5153  191.5153  171.3343  171.3343  156.5590  156.5590

frequencies     :   -40    40   -45    45   -36    36   -20    20

Subset 3
Prominent peaks :  318.2962  318.2962  296.4322  296.4322  264.0049  264.0049  251.4537  251.4537

frequencies     :  -36.0000   36.0000  -13.0000   13.0000  -40.0000   40.0000  -34.2500   34.2500

Subset 4
Prominent peaks :  533.7317  533.7317  522.2423  522.2423  406.7896  406.7896  393.1578  393.1578

frequencies     :  -20.0000   20.0000  -36.0000   36.0000  -41.8750   41.8750  -39.8750   39.8750

Subset 5
Prominent peaks :  977.6880  977.6880  913.0787  913.0787  841.8572  841.8572  714.6065  714.6065

frequencies     :   -20    20   -36    36   -10    10   -40    40
```

Harmonics

∴ The 4 harmonic frequencies are:

$$10\ Hz,\ \ 20\ Hz,\ \ 36\ Hz,\ \ 40\ Hz$$

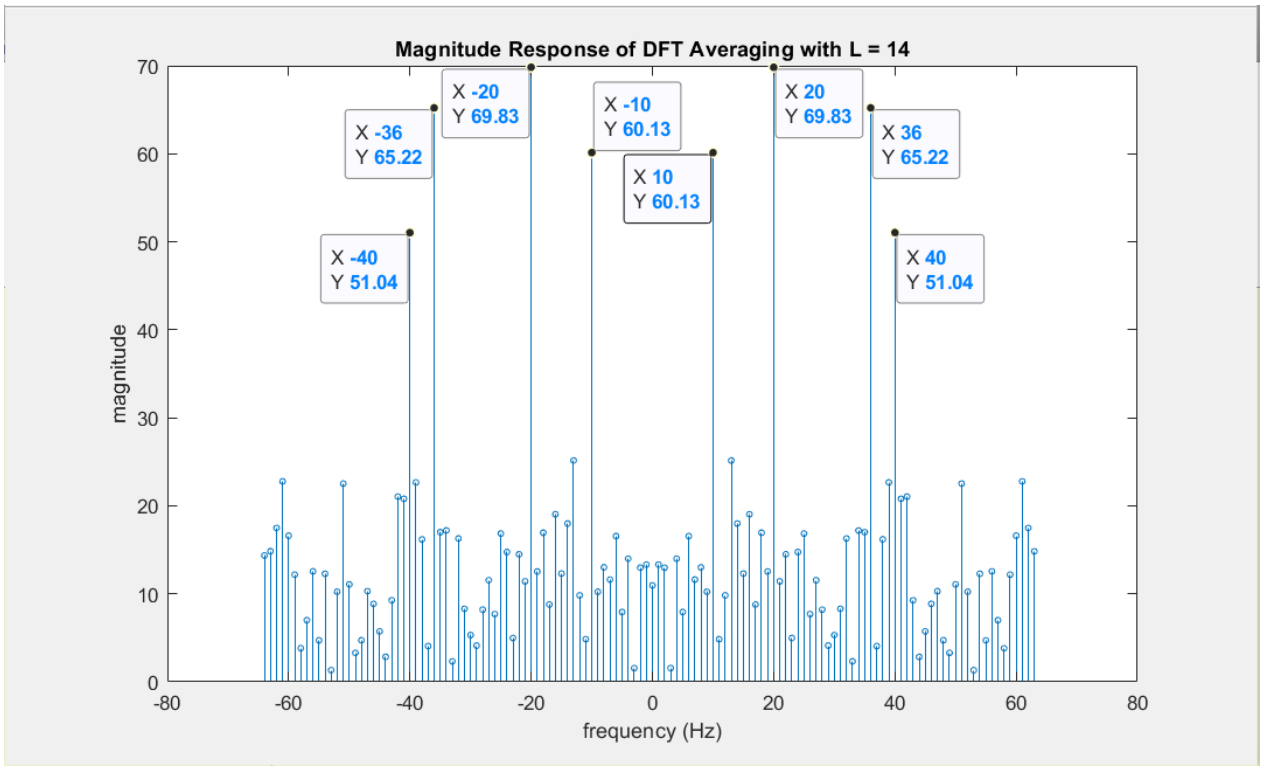
1.1 DFT Averaging Method

Now we split the original signal into 14 subsets each of size equal to 128 samples.

L =14, K =128

```
%% Part 2: DFT averaging method
N = 1792;
L = 14;
K = N/L;
sum_DFT = zeros(1, K);

for i=1:L
    subset = xn_test(((i-1)*K + 1):i*K); % Take subsets
    DFT = fft(subset);
    sum_DFT = sum_DFT + DFT; % Adding DFTs of subsets together
end

average_DFT = sum_DFT/L; % calculating average DFT

[Peak, index] = findpeaks(abs(fftshift(average_DFT)), 'NPeaks', 8, 'SortStr', 'descend');

N = length(average_DFT); % number of DFT points
f = (-N/2:N/2-1)*Fs/N; % Frequency vector
freq = f(index);

%Plot magnitude response
figure;
stem(f, abs(fftshift(average_DFT)),'MarkerSize', 3);
xlabel("frequency (Hz)");
ylabel("magnitude");
title(['Magnitude Response of DFT Averaging with L = ' , int2str(L) ]);

%detect and display harmonics
Harmonics_list = abs(freq); % eliminate negative frequency components
Harmonics = unique(Harmonics_list); % eleminating repeated frequencies
fprintf("The four harmonic frequencies are : ");
disp(sort(Harmonics)); % display harmonics
```

Magnitude Response of the Average DFT



From the above we can conclude that the four harmonic frequencies should be, 10 Hz, 20 Hz, 36 Hz and 40 Hz.

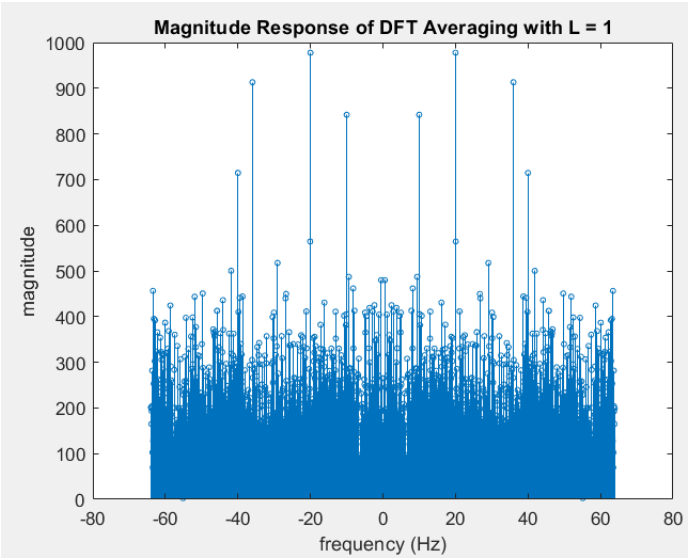
Displayed Output:

```
The four harmonic frequencies are :     10    20    36    40
```

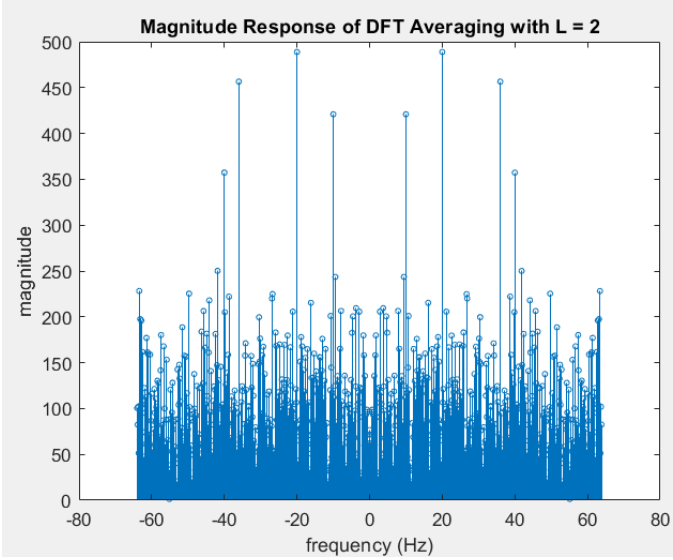
Q5. Finding Minimum L

To see what the minimum value of L is such that four harmonics remain clearly visible, we increase the value of L from 1 up to 14. We can do that by simply changing the value of L in the above code. Also, since number of samples must be an integer, we should make sure to assign values to L which are factors of total number of samples, N (Here 1792). (Possible values less than 14 are: 1,2,4,7,8)

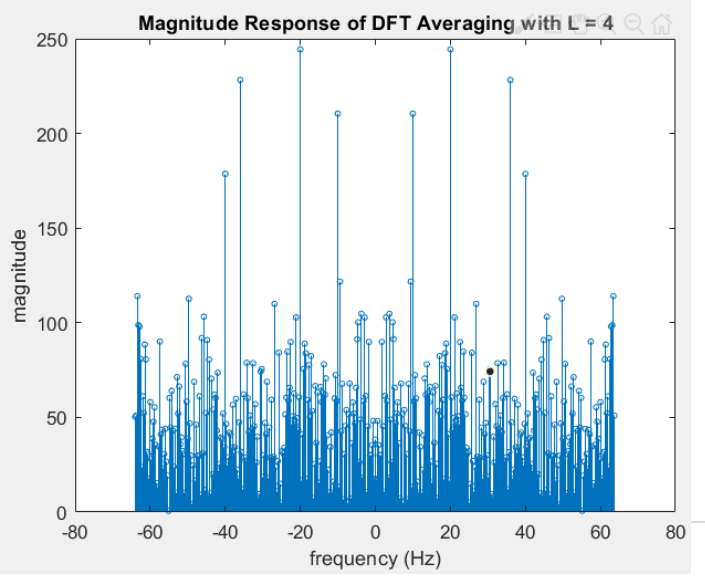
(a) L = 1



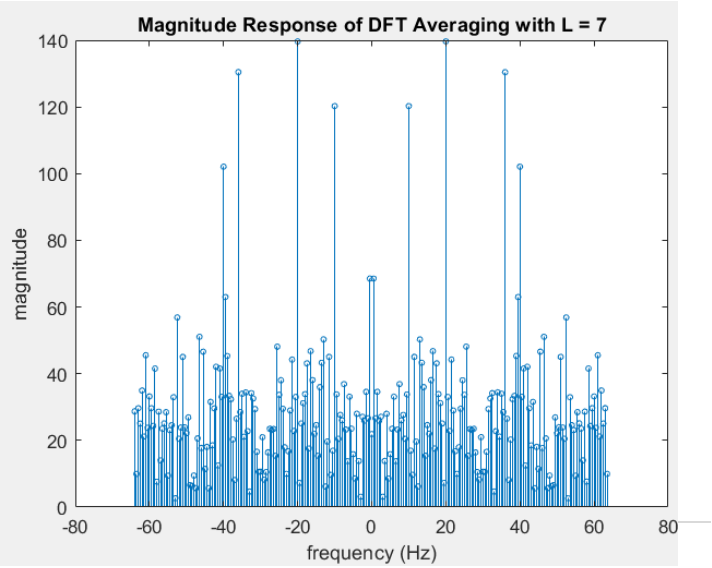
(b) L = 2



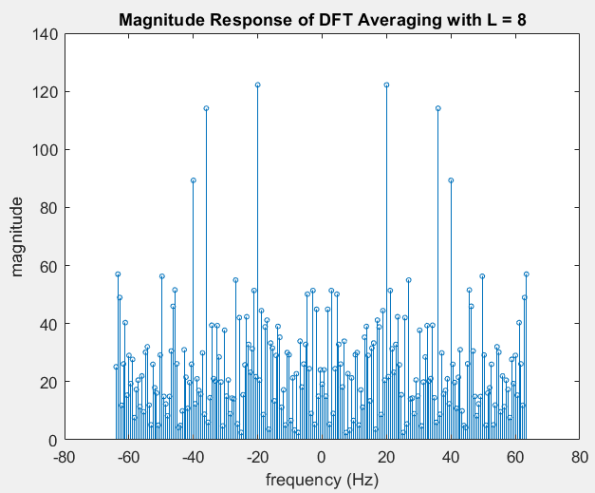
(c) L = 4



(d) L = 7



(e) L = 8



In L = 8 case, all the harmonics are not visible. Only 3 harmonics are visible.

From the above results, the minimum value of L that gives clear image of harmonics is when **L =1**. That means when the subset overlaps with the complete signal.

K = 100 or K = 135

In DFT averaging method, we need subsets to be same in size. Here our total sample size, N is 1792. To get the use of all the samples in DFT averaging, we need K to be a factor of N. Neither 100 nor 135 is a factor of 1792. Therefore, this would result in size imbalance between subsets.

If we use one of these K values keeping the same subset size then, some sample data will not be considered during the calculations.

Say K = 135,
the maximum number of samples we can get such that it is divisible by 135 is 1755. Then 37 samples will be left unused. This is not an effective solution.
On the other hand, lower K values result in a low frequency resolution which can affect the visibility of the harmonics in the DFT.

2. Interpolation with Zero Insertion

First, four signals named x, x2, x3, and x4 are created by extracting samples from music clip y as follows. Then their DFTs are obtained.

```
%% Part 3: Interpolation

load handel; %the audio file is by default saved to a file named y

N = 20000;
%creating signals from the original
x = y(1 : N);
x2 = x(1 : 2 : N);
x3 = x(1 : 3 : N);
x4 = x(1 : 4 : N);

%calculation DFT of signals
DFT_X = fft(x);
DFT_X2 = fft(x2);
DFT_X3 = fft(x3);
DFT_X4 = fft(x4);
```

A function is defined to interpolate any given signal S with a given K value.

```
%Interpolation with zero insertion
function result = zeroInterpolation (S,K)
    n = length(S); %signal length
    if mod(n , 2) == 1 %case 1: odd length signal
        n1 = (n+1)/2;
        result = [S(1 : n1); zeros(K*n, 1);S((n1 + 1) : n)];

    else %case 2: even length signal
        n1 = n/2;
        result = [S(1 : n1) ; S(n1 + 1)/2 ; zeros(K*n - 1, 1) ; S(n1 + 1)/2 ; S((n1 + 2) : n)];

    end

end
```

Another function is defined to calculate the difference between a given signal and its interpolated signal in 2-norm.

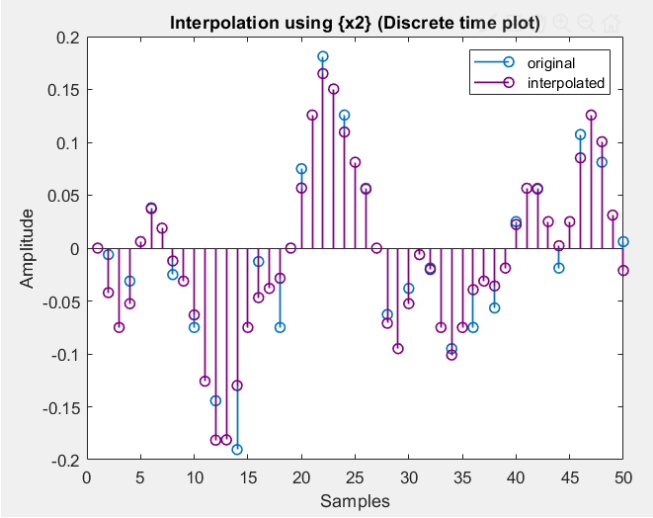
```
%Original and interpolated signal difference in 2-norm
function result = norm_calc(original, interpolated)
    reshape_orig = [original; zeros(length(interpolated) - length(original),1)];
    result = norm(reshape_orig - interpolated);
end
```

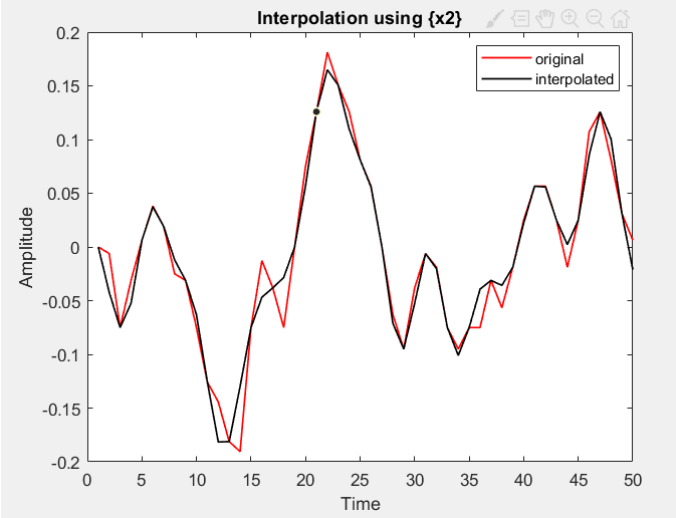
a) Here we are interpolating the signal x2 with K=1. Finally, the original signal and the interpolated signal are plotted on the same graph.

```
%%% Question(a) : Interpolation of signal x2

fprintf("Interpolation using x2 signal \n");
K = 1;
XZ2 = zeroInterpolation (DFT_X2 ,K);                    %interpolation by zero insertion
reconstructed_x2 = ifft(XZ2);                           %inverse DFT
scaled_reconstructed_x2 = (1+K)* reconstructed_x2;      %scaling interpolated signal
difference_X2 = norm_calc (x, scaled_reconstructed_x2); %calculating the difference between original and interpolated signal
fprintf("\tDifference in 2 norm: %.4f\n", difference_X2);

%plot two signals on the same figure
figure;
stem(x(1:50), "LineWidth", 1); %stem plot of the original signal
hold on;
stem(scaled_reconstructed_x2(1:50),'color', [0.5, 0, 0.5], "LineWidth", 1);  %stem plot of the interpolated signal
hold off
ylabel("Amplitude");
xlabel("Samples");
title("Interpolation using \{x2\} (Discrete time plot)")
legend("original", "interpolated");

%plot two signals on the same figure
figure;
plot(x(1:50),'color', [1, 0, 0], "LineWidth", 1); %plot original signal
hold on;
plot(scaled_reconstructed_x2(1:50), 'color', [0, 0, 0], "LineWidth", 1); %plot interpolated signal
hold off
ylabel("Amplitude");
xlabel("Time");
title("Interpolation using \{x2\}")
legend("original", "interpolated");
```

After running the above code, the following outputs were obtained.

```
Interpolation using x2 signal
    Difference in 2 norm: 6.1448
```





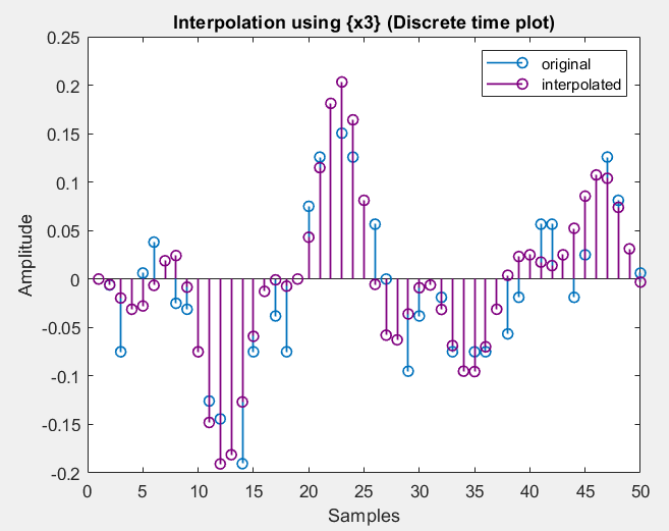
The interpolated signal is very close to the shape of the original signal. The deviation is very small.

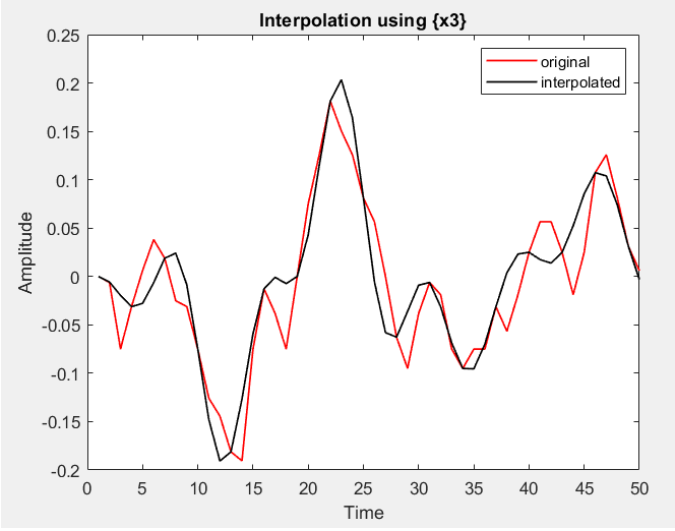
b) As the second case, we interpolate using x3 with K=2. Finally, the original signal and the interpolated signal are plotted on the same graph as previous.

```
%%% Question(b) : Interpolation of signal x3

fprintf("\nInterpolation using x3 signal \n");
K = 2;
XZ3 = zeroInterpolation (DFT_X3 ,K);                        %interpolation by zero insertion
reconstructed_x3 = ifft(XZ3);                               %inverse DFT
scaled_reconstructed_x3 = (1+K)* reconstructed_x3;          %scaling interpolated signal
difference_X3 = norm_calc (x, scaled_reconstructed_x3);     %calculating the difference between original and interpolated signal
fprintf("\tDifference in 2 norm: %.4f\n", difference_X3);


%plot two signals on the same figure
figure;
stem(x(1:50), "LineWidth", 1); %stem plot of the original signal
hold on;
stem(scaled_reconstructed_x3(1:50),'color', [0.5, 0, 0.5], "LineWidth", 1);  %stem plot of the interpolated signal
hold off
ylabel("Amplitude");
xlabel("Samples");
title("Interpolation using \{x3\} (Discrete time plot)")
legend("original", "interpolated");

%plot two signals on the same figure
figure;
plot(x(1:50),'color', [1, 0, 0], "LineWidth", 1); %plot original signal
hold on;
plot(scaled_reconstructed_x3(1:50), 'color', [0, 0, 0], "LineWidth", 1); %plot interpolated signal
hold off
ylabel("Amplitude");
xlabel("Time");
title("Interpolation using \{x3\}")
legend("original", "interpolated");
```

Outputs:

```
Interpolation using x3 signal
    Difference in 2 norm: 8.3652
```





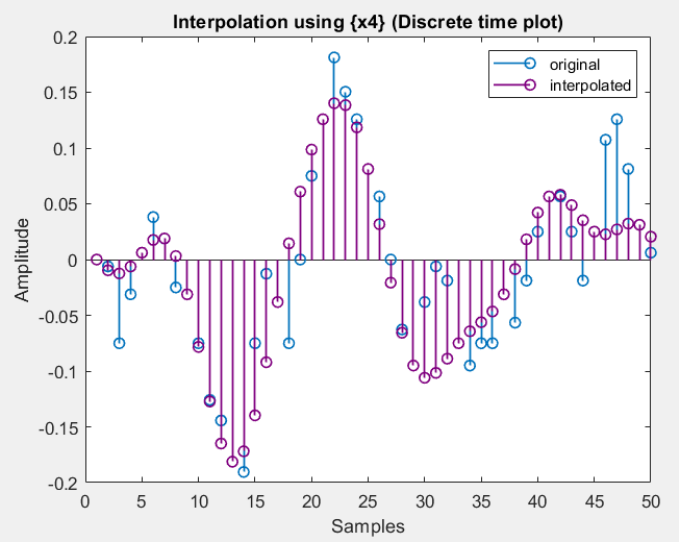
Here, deviation is higher than the previous case.

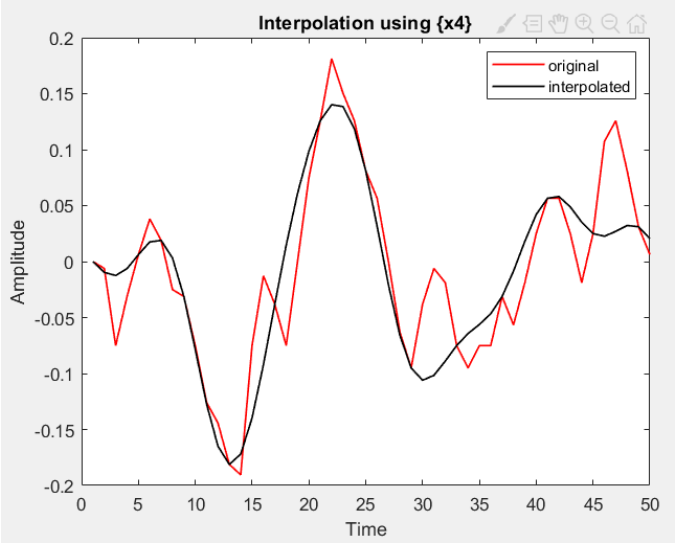
c) As the third case, we interpolate using x4 with K=3. Finally original signal and the interpolated signal is plotted on the same graph as previous.

```
%%% Question(c) : Interpolation of signal x4

fprintf("\nInterpolation using x4 signal \n");
K = 3;
XZ4 = zeroInterpolation (DFT_X4 ,K);                        %interpolation by zero insertion
reconstructed_x4 = ifft(XZ4);                               %inverse DFT
scaled_reconstructed_x4 = (1+K)* reconstructed_x4;          %scaling interpolated signal
difference_X4 = norm_calc (x, scaled_reconstructed_x4);     %calculating the difference between original and interpolated signal
fprintf("\tDifference in 2 norm: %.4f\n", difference_X4);


%plot two signals on the same figure
figure;
stem(x(1:50), "LineWidth", 1); %stem plot of the original signal
hold on;
stem(scaled_reconstructed_x4(1:50),'color', [0.5, 0, 0.5], "LineWidth", 1);  %stem plot of the interpolated signal
hold off
ylabel("Amplitude");
xlabel("Samples");
title("Interpolation using \{x4\} (Discrete time plot)")
legend("original", "interpolated");

%plot two signals on the same figure
figure;
plot(x(1:50),'color', [1, 0, 0], "LineWidth", 1); %plot original signal
hold on;
plot(scaled_reconstructed_x4(1:50), 'color', [0, 0, 0], "LineWidth", 1); %plot interpolated signal
hold off
ylabel("Amplitude");
xlabel("Time");
title("Interpolation using \{x4\}")
legend("original", "interpolated");
```

Outputs:

```
Interpolation using x4 signal
    Difference in 2 norm: 23.4998
```





Here deviation is higher than both previous cases.

Observations:

The deviation of interpolated signal from the original signal is getting higher from x2 to x4. Among the three signals (x2, x3 and x4) used to interpolate the original signal, x2 gives minimum deviation making it closer to the actual signal while x4 gives the highest deviation. When using x3, deviation is higher than that of x2 but lower than x4.

Our orginal signal $x[n]$ consists of first 20, 000 sample points of the music clip.

- $x2[n]: \ interpolated \ with \ K \ = 1$
- $x3[n]: \ interpolated \ with \ K \ = 2$
- $x4[n]: \ interpolated \ with \ K \ = 3$

Among three signals, $x2[n]$ has the highest number of samples and $x4[n]$ has the minimum number of samples from the original signal $x[n]$. Therefore, when we perform frequency domain interpolation with zero padding, we are particularly adding higher number of zeros to $x3[n]$ and $x4[n]$. As a result, we lose more high frequency components in $x3[n]$ and $x4[n]$ than $x2[n]$ leading to higher deviations.

∴ From the results, we can say that when K increases, the deviation of interpolated signal from the original also increases. (Less sample data from the original signal, more zeros)

3. Appendix

Code

```
%loading the corrupted signal
load('signal507.mat','xn_test');

Fs = 128; % sampling frequency
%% Part 1: Harmonic Detection using peaks

%Creating subsets
S1 = xn_test(1:128);
S2 = xn_test(1:256);
S3 = xn_test(1:512);
S4 = xn_test(1:1024);
S5 = xn_test(1:1792);

%Obtain DFT of each subset of samples
DFT_S1 = fft(S1);
DFT_S2 = fft(S2);
DFT_S3 = fft(S3);
DFT_S4 = fft(S4);
DFT_S5 = fft(S5);


for i=1:5
    DFT_S = eval(['DFT_S', int2str(i)]);
    [P1, index] = findpeaks(abs(fftshift(DFT_S)), 'NPeaks', 8, 'SortStr', 'descend'); % Get the eight maximum peak points
    N = length(DFT_S); % number of DFT points
    f = (-N/2:N/2-1)*Fs/N; % Frequency vector
    freq = f(index);

    subplot(5,1,i);
    stem(f,abs(fftshift(DFT_S)),'MarkerSize', 3); %Plot magntude responses of DFT of each signal
    ylabel("| DFT (S"+ int2str(i) + ") |");

    %Display outputs
    disp("Subset " + int2str(i));
    fprintf('Prominent peaks :');
    disp(P1);
    fprintf('frequencies     :');
    disp(freq)

end
sgtitle('Magnitude Response of subsets');
xlabel("frequency");
ax = gca;
```

```
%% Part 2: DFT averaging method
N = 1792;
L = 4;
K = N/L;
sum_DFT = zeros(1, K);

for i=1:L
    subset = xn_test(((i-1)*K + 1):i*K); % Take subsets
    DFT = fft(subset);
    sum_DFT = sum_DFT + DFT; % Adding DFTs of subsets together
end

average_DFT = sum_DFT/L; % calculating average DFT

[Peak, index] = findpeaks(abs(fftshift(average_DFT)), 'NPeaks', 8, 'SortStr', 'descend');

N = length(average_DFT); % number of DFT points
f = (-N/2:N/2-1)*Fs/N; % Frequency vector
freq = f(index);
```

```
%Plot magnitude response
figure;
stem(f, abs(fftshift(average_DFT)),'MarkerSize', 3);
xlabel("frequency (Hz)");
ylabel("magnitude");
title(['Magnitude Response of DFT Averaging with L = ' , int2str(L) ]);

%detect and display harmonics
Harmonics_list = abs(freq); % eliminate negative frequency components
Harmonics = unique(Harmonics_list); % eleminating repeated frequencies
fprintf("The four harmonic frequencies are : ");
disp(sort(Harmonics)); % display harmonics
```

```
%% Part 3: Interpolation

load handel; %the audio file is by default saved to a file named y

N = 20000;
%creating signals from the original
x = y(1 : N);
x2 = x(1 : 2 : N);
x3 = x(1 : 3 : N);
x4 = x(1 : 4 : N);

%calculation DFT of signals
DFT_X = fft(x);
DFT_X2 = fft(x2);
DFT_X3 = fft(x3);
DFT_X4 = fft(x4);


%%% Question(a) : Interpolation of signal x2

fprintf("Interpolation using x2 signal \n");
K = 1;
XZ2 = zeroInterpolation (DFT_X2 ,K);                        %interpolation by zero insertion
reconstructed_x2 = ifft(XZ2);                               %inverse DFT
scaled_reconstructed_x2 = (1+K)* reconstructed_x2;          %scaling interpolated signal
difference_X2 = norm_calc (x, scaled_reconstructed_x2);     %calculating the difference between original and interpolated signal
fprintf("\tDifference in 2 norm: %.4f\n", difference_X2);

%plot two signals on the same figure
figure;
stem(x(1:50), "LineWidth", 1); %stem plot of the original signal
hold on;
stem(scaled_reconstructed_x2(1:50),'color', [0.5, 0, 0.5], "LineWidth", 1);  %stem plot of the interpolated signal
hold off
ylabel("Amplitude");
xlabel("Samples");
title("Interpolation using \{x2\} (Discrete time plot)")
legend("original", "interpolated");

%plot two signals on the same figure
figure;
plot(x(1:50),'color', [1, 0, 0], "LineWidth", 1); %plot original signal
hold on;
plot(scaled_reconstructed_x2(1:50), 'color', [0, 0, 0], "LineWidth", 1); %plot interpolated signal
hold off
ylabel("Amplitude");
xlabel("Time");
title("Interpolation using \{x2\}")
legend("original", "interpolated");
```

```
%%% Question(b) : Interpolation of signal x3

fprintf("\nInterpolation using x3 signal \n");
K = 2;
XZ3 = zeroInterpolation (DFT_X3 ,K);                        %interpolation by zero insertion
reconstructed_x3 = ifft(XZ3);                               %inverse DFT
scaled_reconstructed_x3 = (1+K)* reconstructed_x3;          %scaling interpolated signal
difference_X3 = norm_calc (x, scaled_reconstructed_x3);     %calculating the difference between original and interpolated signal
fprintf("\tDifference in 2 norm: %.4f\n", difference_X3);


%plot two signals on the same figure
figure;
stem(x(1:50), "LineWidth", 1); %stem plot of the original signal
hold on;
stem(scaled_reconstructed_x3(1:50),'color', [0.5, 0, 0.5], "LineWidth", 1);  %stem plot of the interpolated signal
hold off
ylabel("Amplitude");
xlabel("Samples");
title("Interpolation using \{x3\} (Discrete time plot)")
legend("original", "interpolated");

%plot two signals on the same figure
figure;
plot(x(1:50),'color', [1, 0, 0], "LineWidth", 1); %plot original signal
hold on;
plot(scaled_reconstructed_x3(1:50), 'color', [0, 0, 0], "LineWidth", 1); %plot interpolated signal
hold off
ylabel("Amplitude");
xlabel("Time");
title("Interpolation using \{x3\}")
legend("original", "interpolated");
```

```
%%% Question(c) : Interpolation of signal x4

fprintf("\nInterpolation using x4 signal \n");
K = 3;
XZ4 = zeroInterpolation (DFT_X4 ,K);                        %interpolation by zero insertion
reconstructed_x4 = ifft(XZ4);                               %inverse DFT
scaled_reconstructed_x4 = (1+K)* reconstructed_x4;          %scaling interpolated signal
difference_X4 = norm_calc (x, scaled_reconstructed_x4);     %calculating the difference between original and interpolated signal
fprintf("\tDifference in 2 norm: %.4f\n", difference_X4);


%plot two signals on the same figure
figure;
stem(x(1:50), "LineWidth", 1); %stem plot of the original signal
hold on;
stem(scaled_reconstructed_x4(1:50),'color', [0.5, 0, 0.5], "LineWidth", 1);  %stem plot of the interpolated signal
hold off
ylabel("Amplitude");
xlabel("Samples");
title("Interpolation using \{x4\} (Discrete time plot)")
legend("original", "interpolated");

%plot two signals on the same figure
figure;
plot(x(1:50),'color', [1, 0, 0], "LineWidth", 1); %plot original signal
hold on;
plot(scaled_reconstructed_x4(1:50), 'color', [0, 0, 0], "LineWidth", 1); %plot interpolated signal
hold off
ylabel("Amplitude");
xlabel("Time");
title("Interpolation using \{x4\}")
legend("original", "interpolated");
```

```
%Interpolation with zero insertion
function result = zeroInterpolation (S,K)
    n = length(S); %signal length
    if mod(n , 2) == 1 %case 1: odd length signal
        n1 = (n+1)/2;
        result = [S(1 : n1); zeros(K*n, 1);S((n1 + 1) : n)];

    else %case 2: even length signal
        n1 = n/2;
        result = [S(1 : n1) ; S(n1 + 1)/2 ; zeros(K*n - 1, 1) ; S(n1 + 1)/2 ; S((n1 + 2) : n)];

    end

end

%Original and interpolated signal difference in 2-norm
function result = norm_calc(original, interpolated)
    reshape_orig = [original; zeros(length(interpolated) - length(original),1)];
    result = norm(reshape_orig - interpolated);
end
```