Department of Electronic & Telecommunication Engineering, University of Moratuwa, Sri Lanka.

EN3551 Assignment 01:

Detecting Harmonics in Noisy Data and Signal Interpolation using DFT

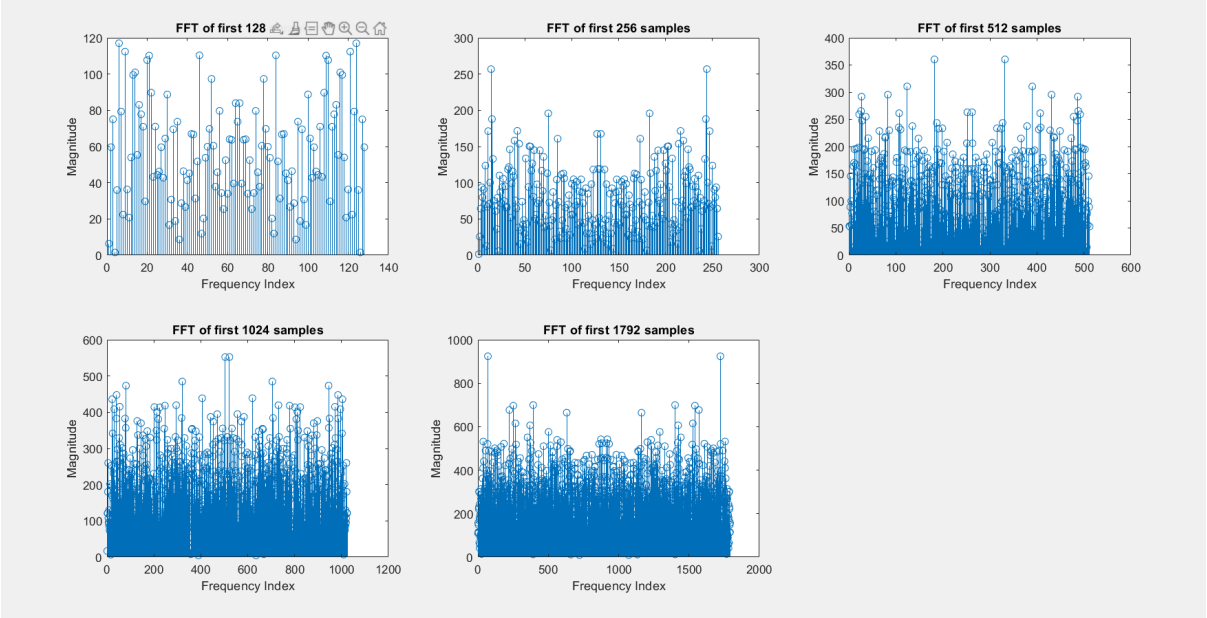
Amugoda A. S. S 210041M

10 September 2024

A. Procedure

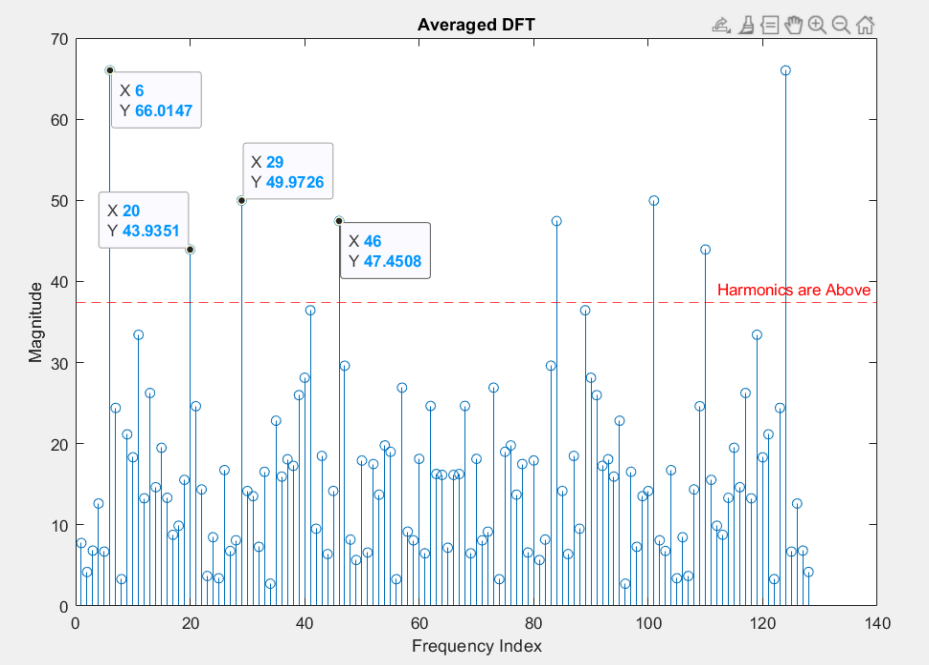
3.1 Harmonic Detection

3.1.3 Visualization of DFT of First few Samples



- Different sample sizes imply different harmonics (significant frequencies).
- Assuming *no cycle mismatches happen for multiples of 128*, we can deduce that with severe *WGN* there is no ability to directly identify harmonics even with a large number of samples.

3.1.4 DFT Averaging over L Subsets of Size K

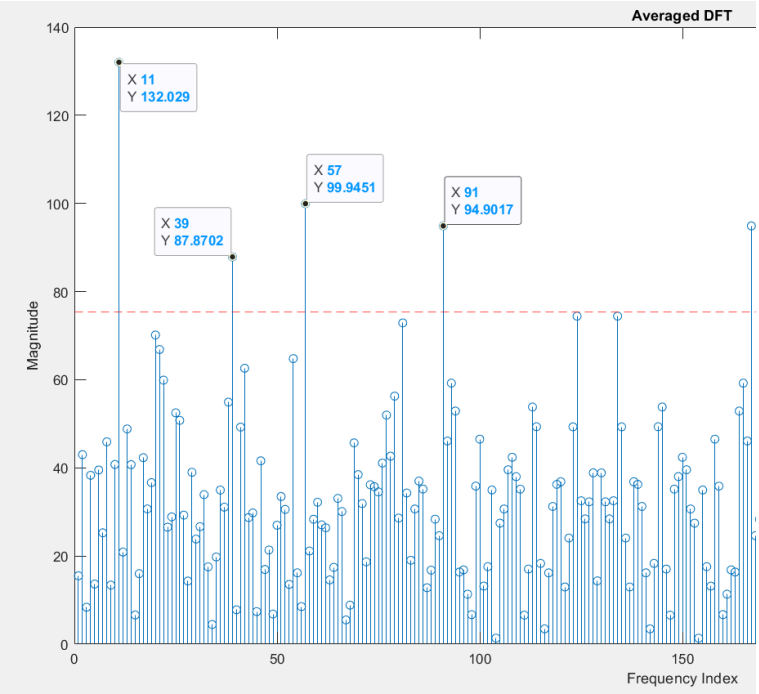


- Harmonics we can derive from this plot:
  - $5 \rightarrow \frac{5}{128} \times 128 = 5\,Hz$ [1]
  - $19\,Hz$
  - $28\,Hz$
  - $45\,Hz$

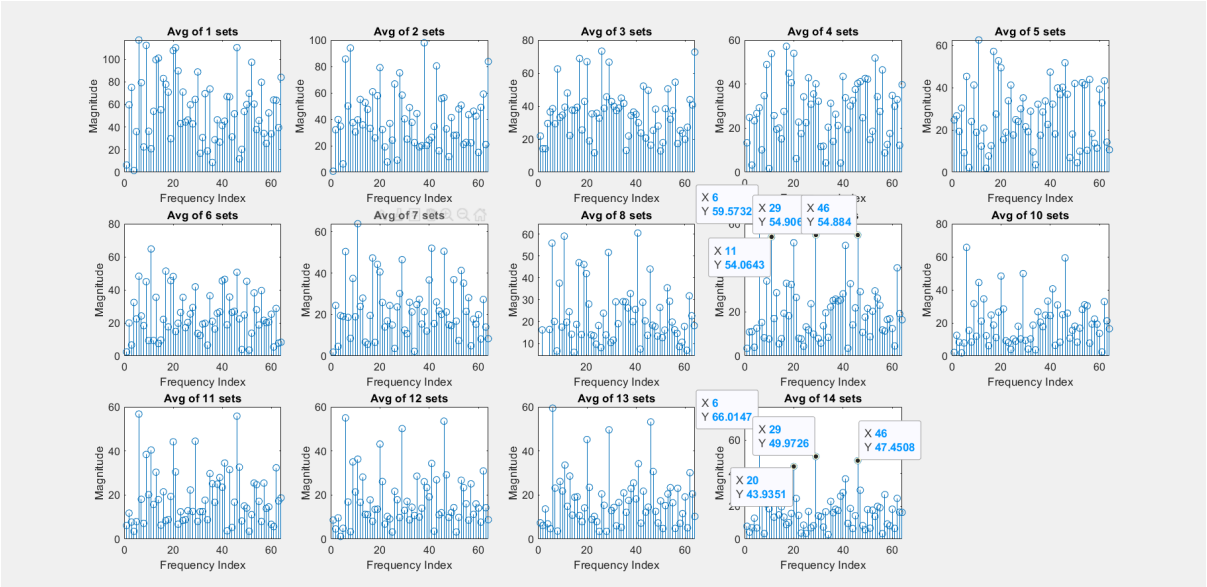
- Harmonics from subsets of 256 samples (see below) :
  - $11 \rightarrow \frac{10}{256} \times 128 = 5\,Hz$
  - $39 \rightarrow \frac{38}{256} \times 128 = 19\,Hz$
  - $57 \rightarrow \frac{56}{256} \times 128 = 28\,Hz$
  - $91 \rightarrow \frac{90}{256} \times 128 = 45\,Hz$

- Peaks from taking 7 sets of 256 samples

[1] MatLab plots start from index 1. To get the DFT index subtract 1. In this case X value 6 becomes 5.



3.1.5 Smallest Number of Subsets Required for Harmonic Identification

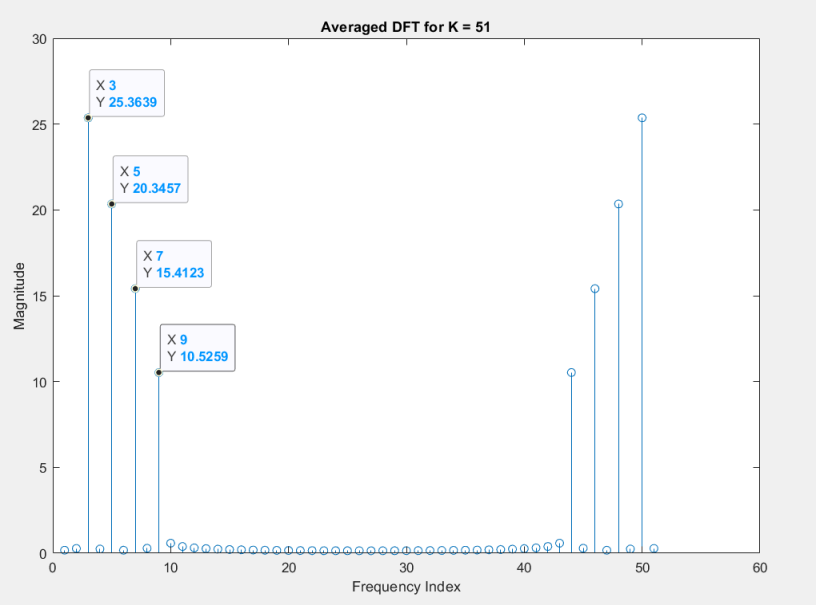


- Having only 9 sets or below leads to false harmonic/ significant frequencies.
- Averaging should happen over at least 10 subsets for the assigned signal and a sample size of 128 per subset.

3.1.6 Using a Different K Value

- To correctly identify the harmonics the number of samples should be such that the corresponding sample interval should contain only complete cycles of all the harmonics. (If complete cycles of the fundamental harmonic are in the interval.)
- When this doesn't happen unnecessary frequency components appear in the DFT.

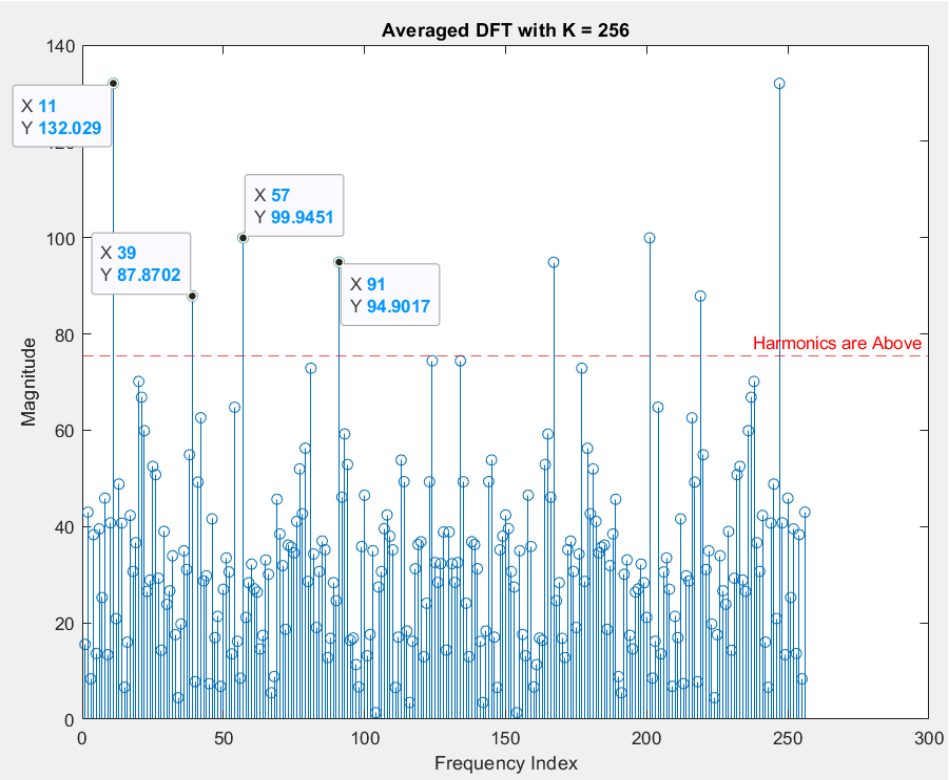
- This can clearly be seen when using a harmonic set *without noise*.
- This is the DFT of the first four harmonics of 5 *Hz* sampled at 128 *Hz.* Sample size is taken to be 51.[2]



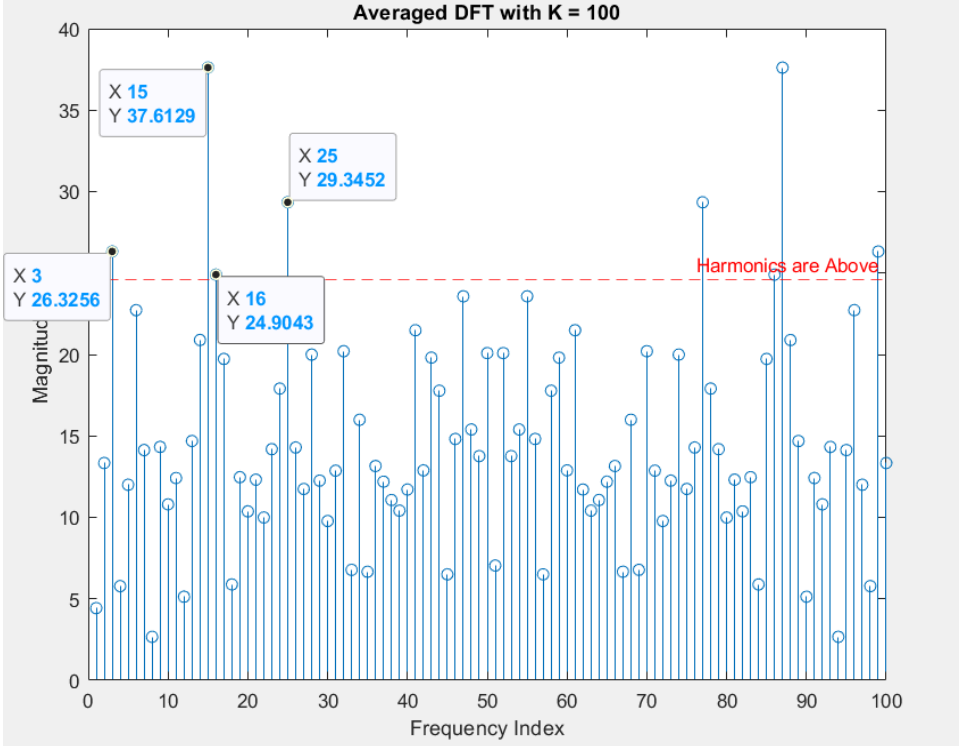
- Undesired components have appeared for incompatible sample size even without noise.

[2] Period of the fundamental is 0.2 *s.* 51 samples is one of the closest number-of-samples having complete cycles of the fundamental frequency and is not a multiple of 128. (51 =0.2 s * 128 Hz * 2)

- Also, different sample sizes for our *noisy* signal.



- For 256 (multiple of 128) harmonics can be currently identified.

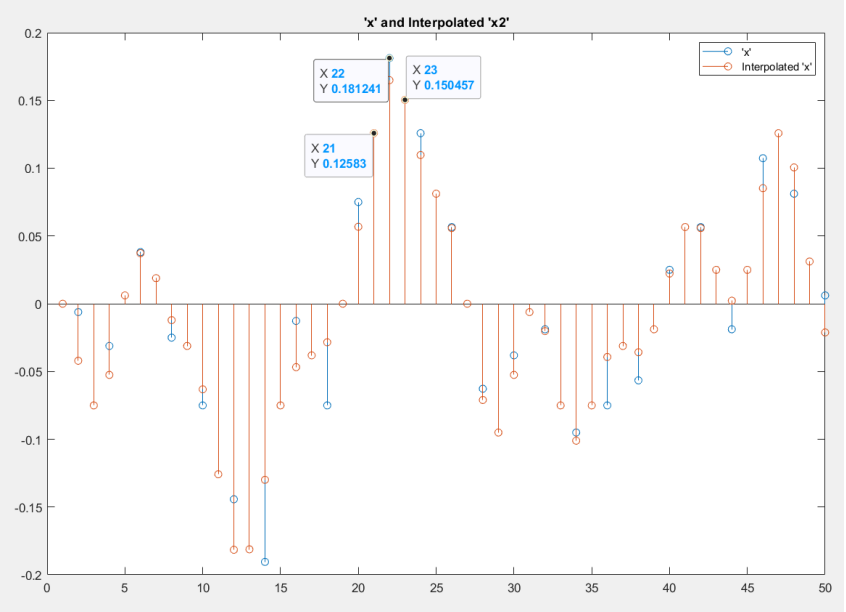


- Significant components implying wrong harmonics.

3.2 Interpolation

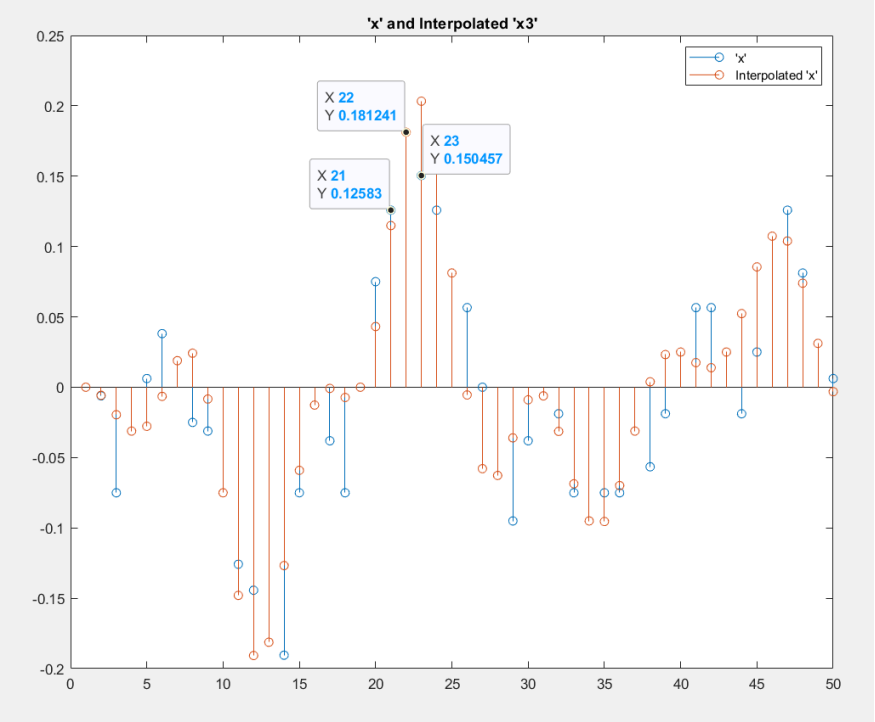
3.2. a x_2 with $K = 1$.

- $L2$ distance between x_2 and $x =$ 6.1448



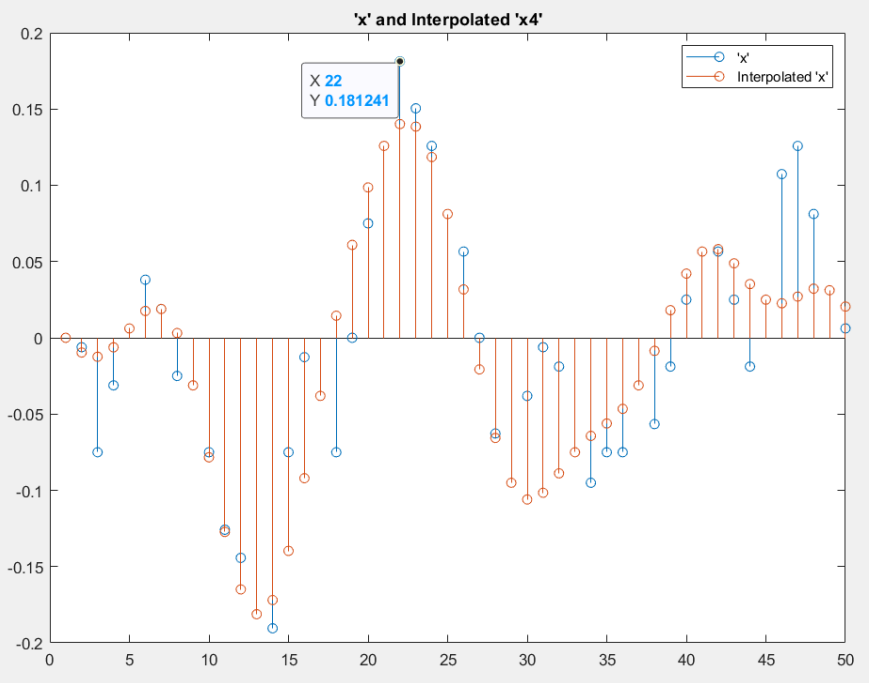
3.2. b x_3 with $K = 2$.

- $L2$ distance between x_3 and $x =$ 8.3652



3.2. c with $K = 3$.

- $L2$ distance between x_4 and $x =$ 23.4998



3.2. d Observations

- With an increased K value (more downsampled signal) the obtained interpolated signal becomes smoother.
- But, the error between the two signals becomes larger.

B. Appendix

3.1 Harmonic Detection

3.1.1 Code for instructions 3.1.1, 3.1.2, 3.1.3

```
clear;
clc;
s_N = load('signal41.mat').xn_test;% All noisy 1793 samples as contiguous array


% First we will take samples of 128, 256, 512, 1024, and 1792 and visualize their
DFT's

figure;
list = [128 256 512 1024 1792];
for i = 1:length(list)
    X = fft( s_N(1: list(i)));
    subplot(2, 3, i);
    stem(abs(X)); % Magnitude
    title(['FFT of first ', num2str(list(i)), ' samples']);
    xlabel('Frequency Index');
    ylabel('Magnitude');
    hold on;
end
```

3.1.2 Code for instructions 3.1.4, 3.1.5

```
% Change only the loaded s_N and K value for different results
clc;
Clear;

s_N = load('signal41.mat').xn_test;% All noisy 1793 samples as a contiguous array
K = 128;%sample size
L = floor(length(s_N)/ K);%number of sets of size K

% Samples divided into 'L' number of subsets of size 'K'
sgnls = transpose( reshape( s_N(1:K*L), [K L]));
% Take FFT of each row of X_N
X_N  = fft(sgnls, [], 2);

% Averaged DFT
figure(1);
X_Avg = sum(X_N, 1)/ size(X_N, 1);
stem(abs(X_Avg)); % Magnitude
% Separate the highest four peaks
hold on;
sorted_magX = sort(abs(X_Avg), 'descend');  % Sort magnitudes in descending order
```

```
threshold = sorted_magX(9)+1;           % The 4th highest peak
yline(threshold, 'r--','Harmonics are Above');  % Plot a red dashed line

title(['Averaged DFT with K = ', num2str(K)]);
xlabel('Frequency Index');
ylabel('Magnitude');

% To find the minimum number of subsets required
figure(2);
for i = 1:L
    X_N_i  = X_N(1:i,:);
    X_Avg = sum(X_N_i, 1)/ size(X_N_i, 1);
    subplot(3,5,i);
    stem(abs(X_Avg)); % Magnitude
    title(['Avg of ', num2str(i), ' sets']);
    xlabel('Frequency Index');
    xlim([0 floor((K+1)/2)]);
    ylabel('Magnitude');
end
 hold off;
```

3.1.3 Code for instruction 3.1.6

- This is an extra code block to visualize the cycle mismatch from a harmonic set without noise

```
Fs = 128;      % Sampling frequency in Hz
duration = 14; % Duration in seconds
N = Fs * duration; % Total number of samples

t = (0:N-1) / Fs;  % Time vector from 0 to duration with N points

% Frequencies for the harmonics (in Hz)
f1 = 5;    % Fundamental frequency
f2 = 10;   % Second harmonic
f3 = 15;   % Third harmonic
f4 = 20;   % Fourth harmonic

% Amplitudes for each harmonic
A1 = 1;    % Amplitude of the first harmonic
A2 = 0.8;  % Amplitude of the second harmonic
A3 = 0.6;  % Amplitude of the third harmonic
A4 = 0.4;  % Amplitude of the fourth harmonic

% Generate the signal with 4 harmonics
signal = A1*sin(2*pi*f1*t) + A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t) +
A4*sin(2*pi*f4*t);
```

```
% Plot the generated signal
figure;

%plot(t, signal);
%title('Generated Signal with 4 Harmonics');
%xlabel('Time (s)');
%ylabel('Amplitude');

K = 135;%sample size
L = floor(length(s_N)/ K);%number of sets of size K
sgnls = transpose( reshape( signal(1:K*L), [K L]));% Samples divided into 'L' number
of subsets of size 'K'
X_N  = fft(sgnls, [], 2);


stem(abs(X_N(1,:))); % Magnitude
title(['DFT for K = ',num2str(K)]);
xlabel('Frequency Index');
ylabel('Magnitude');
```

3.2. Interpolation

- This is the main code. Find the interpolation function ("*intpol*") below.

```
clear;
clc;
% Load the first 20,000 samples of a music clip "Hallelujah" by Handel
load handel
N =20000;
x = y(1:N);

% Downsample x by taking one sample from every 2, 3, and 4 samples
x2 = x(1:2:N);
x3 = x(1:3:N);
x4 = x(1:4:N);

% Call interpolating function passing the number of samples required per
% every existing sample
itpl_x2 = intpol(x2, 1);
itpl_x3 = intpol(x3, 2);
itpl_x4 = intpol(x4, 3);
```

```
% Calculate the L2 distance between the true signal and the interpolated
n1 = norm(x-itpl_x2);
n2 = norm(x-itpl_x3(1:N) );
n3 = norm(x-itpl_x4);

disp(['L2 distance between x and the interpolated x2 = ', string(n1)]);
disp(['L2 distance between x and the interpolated x3 = ', string(n2)]);
disp(['L2 distance between x and the interpolated x4 = ', string(n3)]);


% Plot the interpolated signals along with the true signal
i = 0;
for y = [itpl_x2(1:50), itpl_x3(1:50), itpl_x4(1:50)]
  i=i+1;
  figure(i);
  stem(x(1:50));
  hold on;
  stem(y);
  title("'x' and Interpolated 'x"+string(i+1)+"'")
  legend("'x'", "Interpolated 'x'");
  hold off;
end
```

- "*intpol*" function.

```
function itpl_x = intpol(smpl_x, K)
% This function interpolates downsampled function smpl_x and produce itpl_x
% with K+1 times more samples.
% This is done by zero padding at the middle of the DFT
  X = fft(smpl_x);
  N = length(X);
  N1 = (N+1)/2;% For odd N
  N2 = N/2;% For even N

  if rem(N, 2) == 0 % If even
    zpd_X = [X(1 : N2); X(N2 +1)/2; zeros(K*N -1,1); X(N2 +1)/2; X((N2 +2) : N)];
  else % If odd
    zpd_X = [X(1 : N1);zeros(K*N,1);X( (N1+1): N)];
  end
  % Take the inverse and scale
  ifft_X = ifft(zpd_X);
```

```
    itpl_x = (K+1)*ifft_X;

end
```