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

Detecting Harmonics in Noisy Data and Signal Interpolation using DFT

Abeysinghe D.U. 210011X

Submitted in partial fulfillment of the requirements for the module
EN 3551 Digital Signal Processing

16th September 2024

3 Harmonic Detection

3.1 Loading the signal with noise

The noise corrupted signal was downloaded and `singal11.m` was selected as my index number is 210011X.

The following code was used to load the signal file.

```
    load('signal11.mat','xn_test');
```

3.2 Construction of subsets from $\{x[n]\}$

Several samples were created by taking the 128, 256, 512, 1024, and 1792 samples from the signal $\{x[n]\}$.

```
%dviding xn_test into subsamples
S1 = xn_test(1:129);
S2 = xn_test(1:257);
S3 = xn_test(1:513);
S4 = xn_test(1:1025);
S5 = xn_test(1:1793);
```

3.3 Applying DFT to each subset

The DFT of each sample can be calculated using the `fft()` function of Matlab.

```
% Apply DFT to each subsample
dft_S1 = fft(S1);
dft_S2 = fft(S2);
dft_S3 = fft(S3);
dft_S4 = fft(S4);
dft_S5 = fft(S5);
```

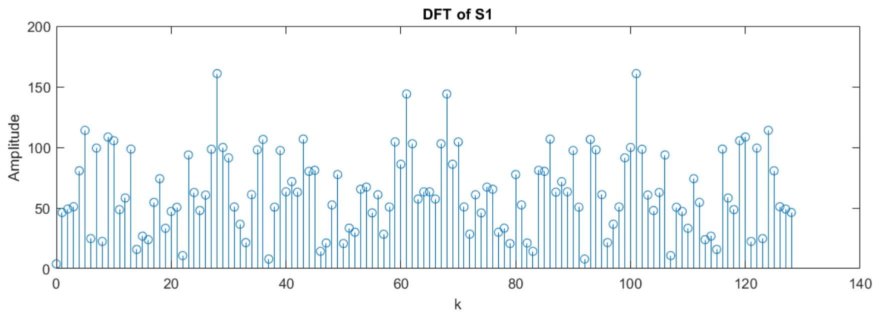


Figure 1: DFT of S1

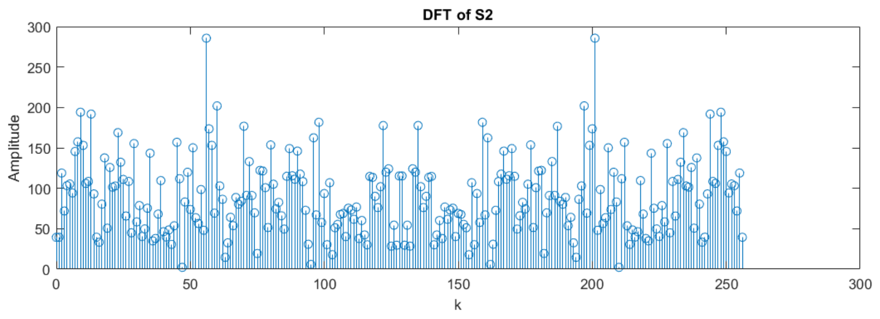


Figure 2: DFT of S2

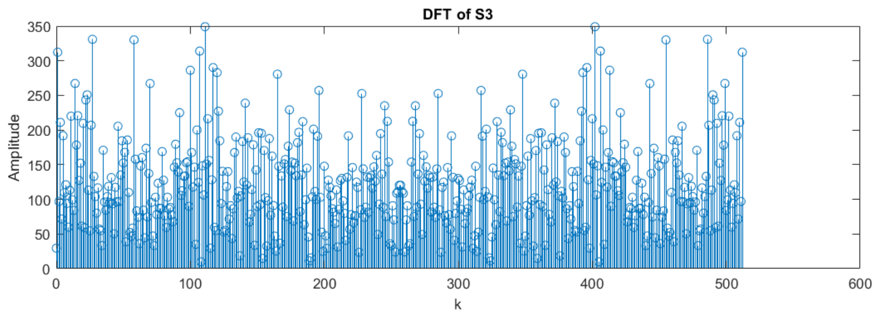


Figure 3: DFT of S3

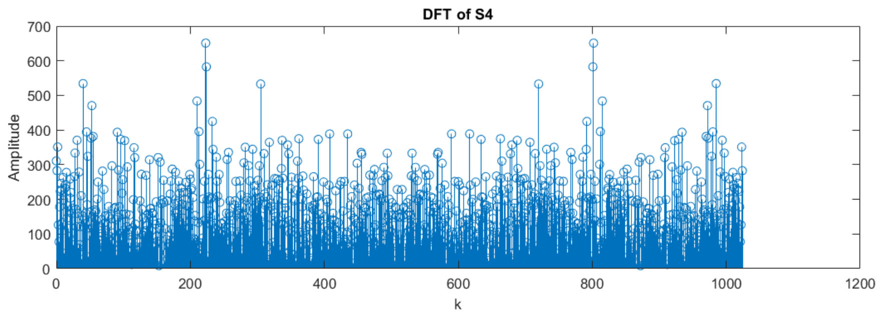


Figure 4: DFT of S4

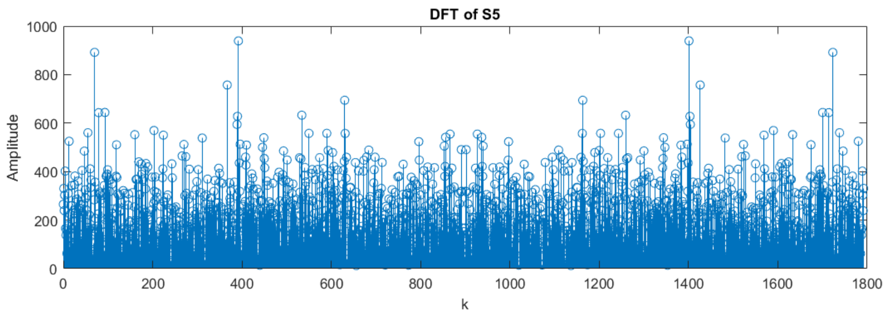


Figure 5: DFT of S5

For the ease of identifying harmonics, I converted the k values into frequencies and plotted the magnitudes of the DFT against frequency values.

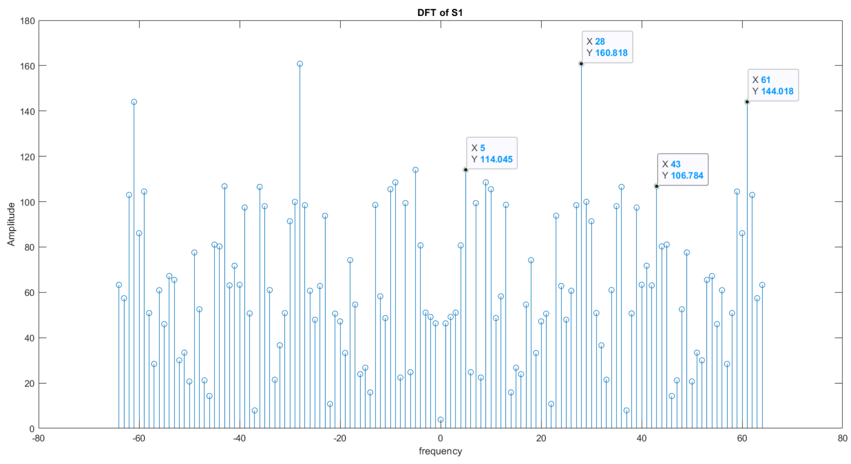


Figure 6: DFT of S1

In this subset multiple peaks can be observed as possible values of harmonics with much prominent peaks at 28Hz and 61Hz.

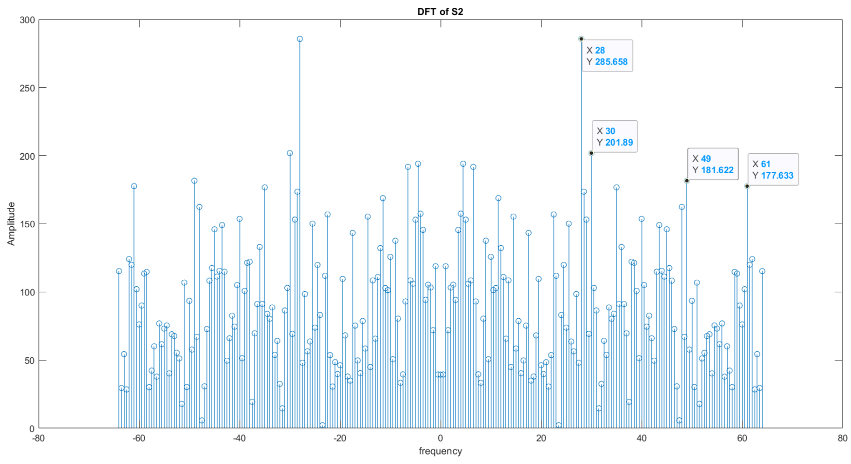


Figure 7: DFT of S2

In the subset S2, a peak can be clearly observed at 28Hz. This can be an affirmation that there can be a harmonic at 28Hz but we cannot be sure at this stage. The same can be said about the peak at 61Hz.

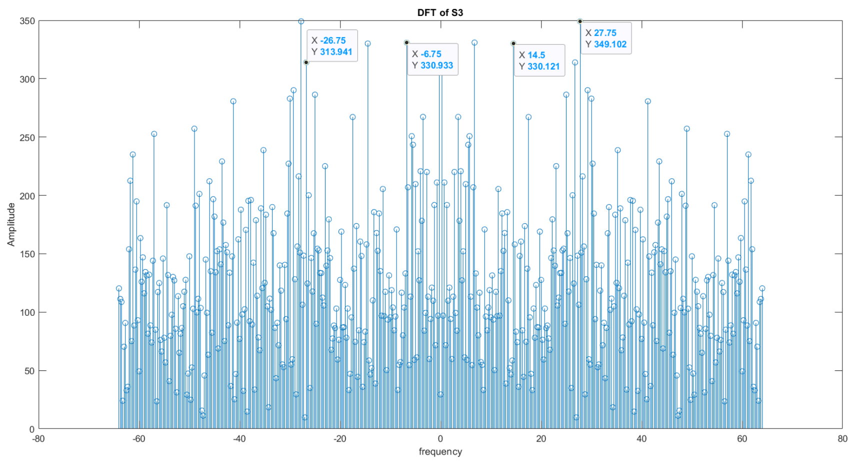


Figure 8: DFT of S3

In S3, apart from the peak around 26-28Hz, we can see some activity near 14Hz and 6-7Hz. These peaks could be due to the new time samples, due to the additive noise, or a combination of both. We cannot be sure there is a harmonic at these peaks.

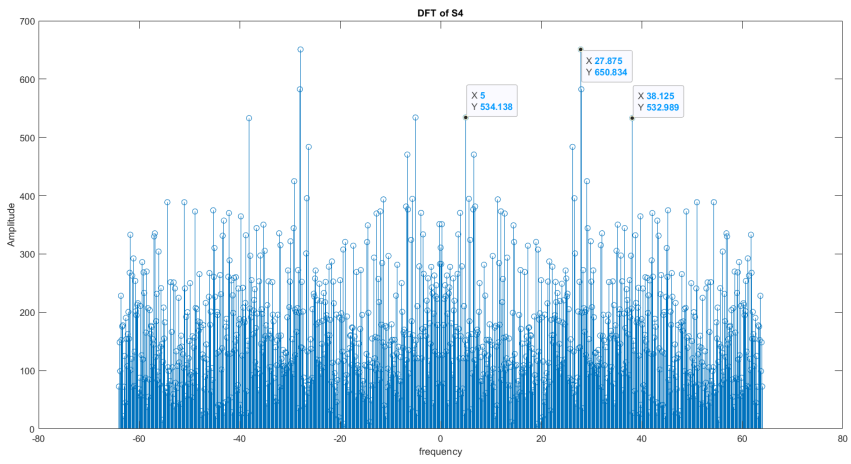


Figure 9: DFT of S4

In subset S4, more peaks are formed at 38-29Hz and 5Hz.

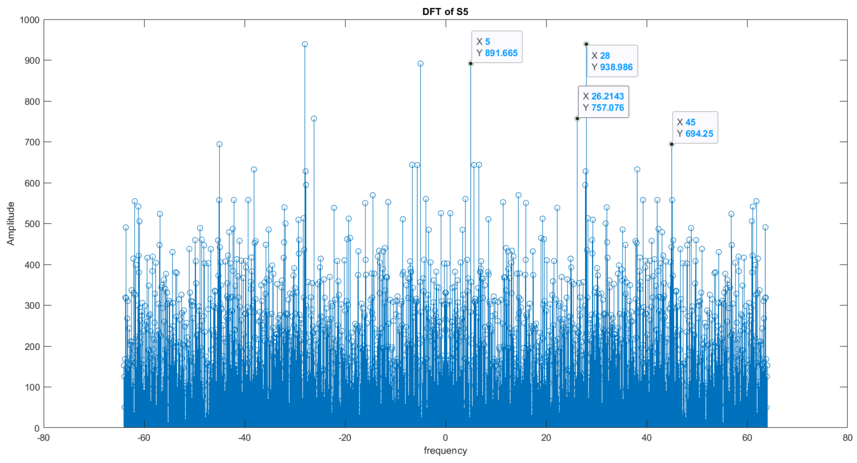


Figure 10: DFT of S5

After looking at the complete subset, S5, we can observe that the peak at 28Hz has been consistent throughout the subsets, increasing the possibility of it being a harmonic. Also, the peak at 5Hz can also be considered a possible value for a harmonic.

Even though the magnitude has decreased in the subsets, we can think of 61Hz as a harmonic as well since it has maintained a relative peak from its surrounding frequencies throughout.

3.4 DFT Averaging

The DFT of the signal was averaged and plotted using the signal subsets of equal lengths.

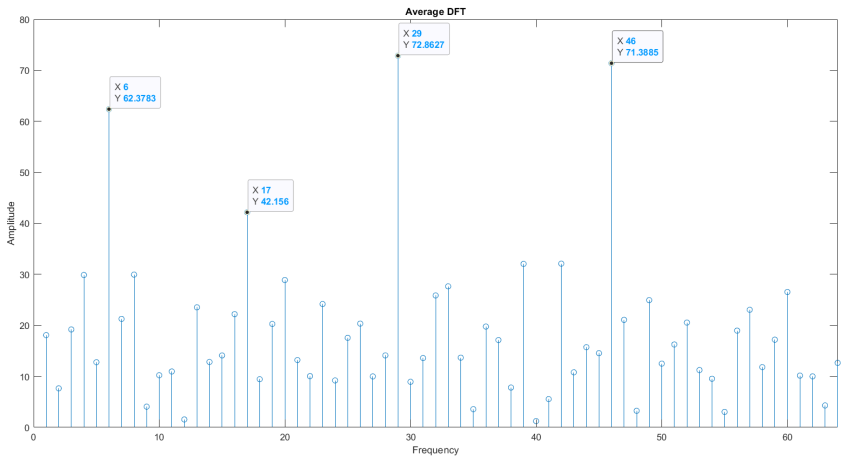


Figure 11: Averaged DFT

The averaged plot gives a better view of the harmonics. A much more prominent peak can be found at 6Hz. Another interesting observation is the somewhat larger peak at 29Hz. Both these two peaks(or the region of the peaks) have been mostly consistent throughout the non-averaged plots of the subsets as well. Apart from these frequencies 17Hz and 46Hz have large magnitudes compared to the other frequency values of the spectrum.

Therefore the harmonics can be determined as 6Hz, 17Hz, 29Hz, and 46Hz.

3.5 Smallest Value for L

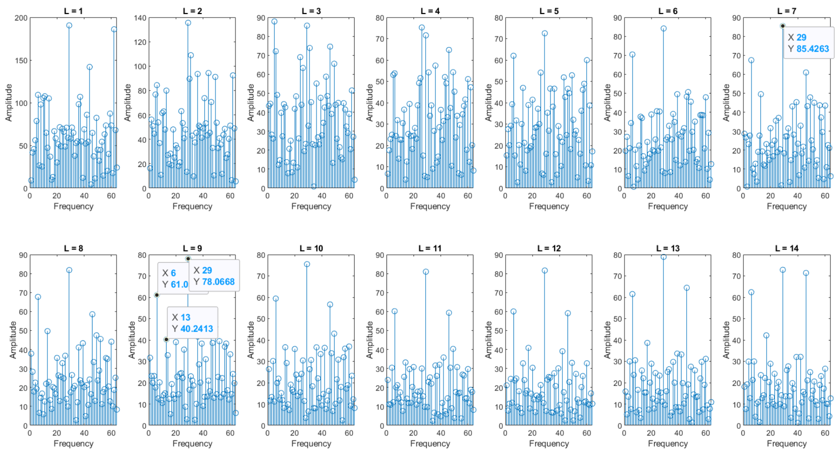


Figure 12: Averaged DFT with different L

The harmonics start to clearly become only after L increments past 7. Therefore, we can conclude that L = 7 is the lowest value L can take.

3.6 Using other values for K

3.6.1 k=100

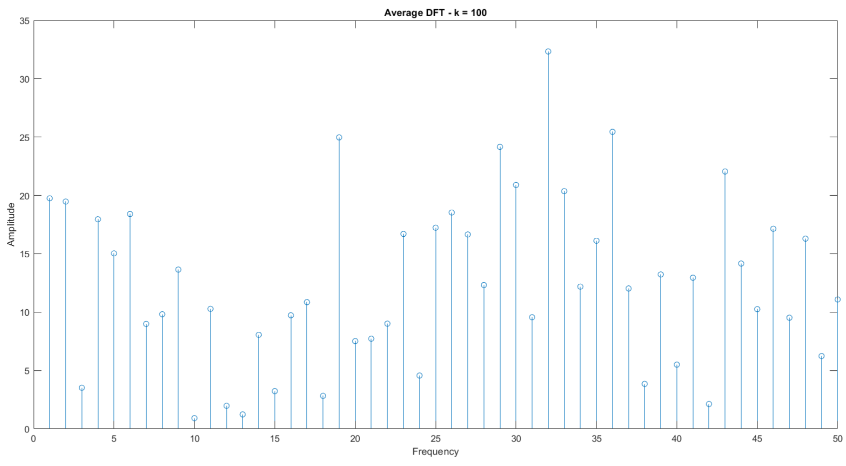


Figure 13: Averaged DFT with k=100

3.6.2 k=135

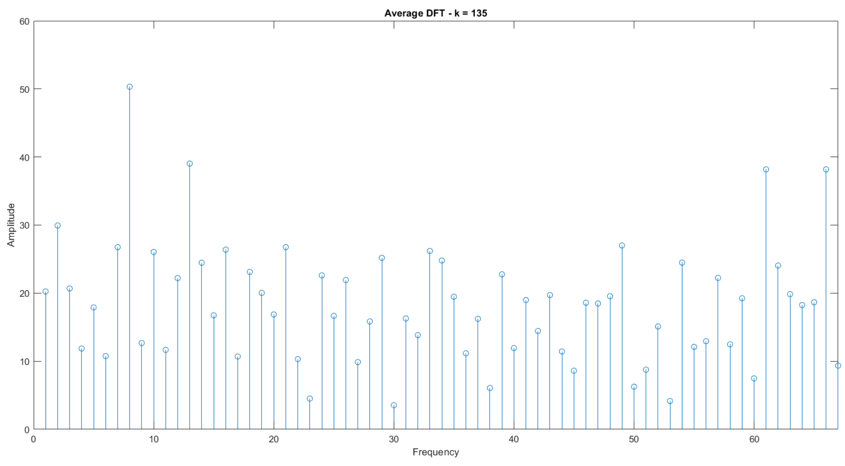


Figure 14: Averaged DFT with k=135

If the value of "K" is set to 100 or 135, the harmonics become less clear. The subset length, represented by K, is consistent for all subsets and defines the number of samples within each group. When the sampling frequency is 128 Hz and K is set to 128, each subset contains a full cycle of the signal, ensuring that the samples are identical across all subsets. However, using other K values, like 100 or 135, can cause issues. When K is not equal to 128, the subsets may not capture complete signal cycles, leading to sample misalignment. This misalignment interferes with cross-subset harmonic contributions, complicating the identification of harmonics. For example, when using techniques like DFT with $K = 100$, the first sample in one subset may not align with the same point in the signal cycle as the first sample in another subset, making averaging difficult.

4 Interpolation

4.1 Loading the signal "handel"

4.2 Signals to be used

The following code is used to define the signals to be used.

```
N = 2000;
signal_subset = y(1:N);
x = y(1:N);
x2 = x(1:2:N);
x3 = x(1:3:N);
x4 = x(1:4:N);
```

4.3 Using DFT-Based method to interpolate the signals

4.3.1 x2 with K = 1

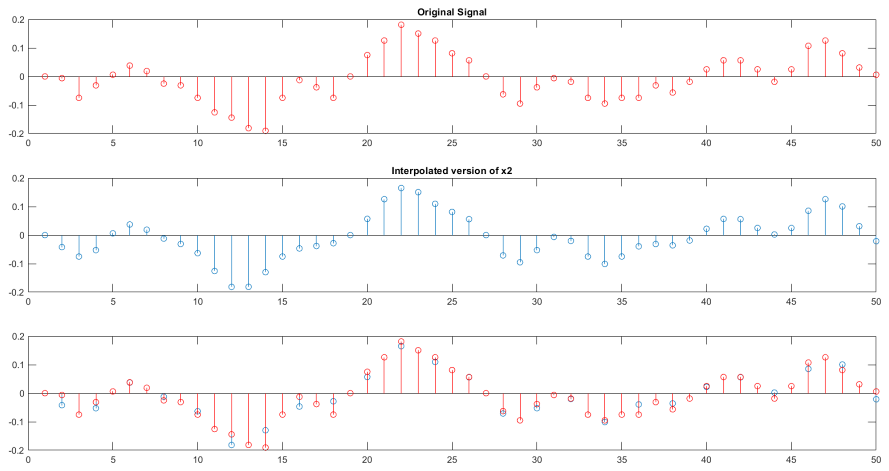


Figure 15: Reconstructed signal using x2 and K=1

4.3.2 x3 with K = 2

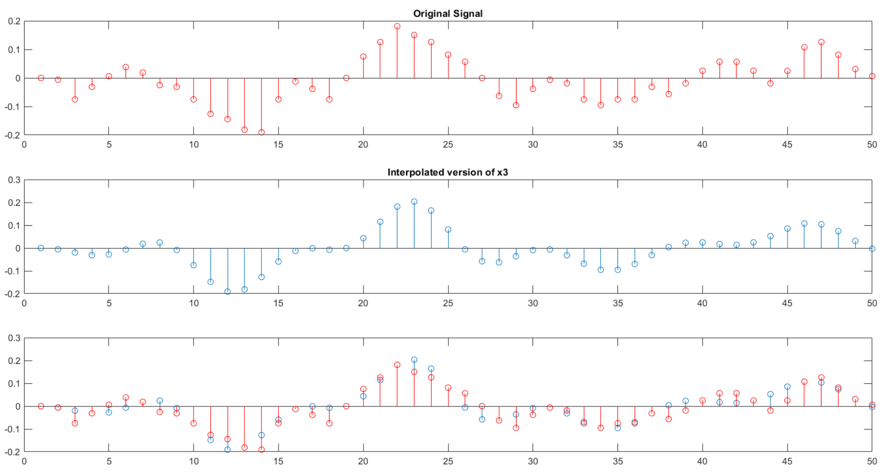


Figure 16: Reconstructed signal using x3 and K=2

4.3.3 x4 with K = 3

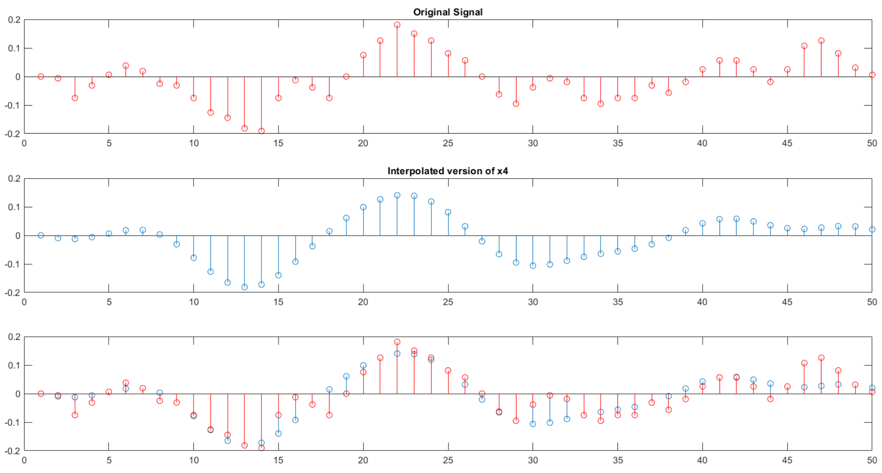
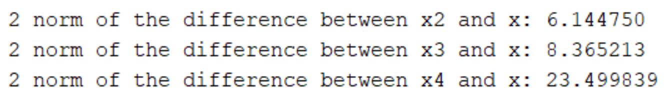


Figure 17: Reconstructed signal using x4 and K=3

After analyzing the plots, it is clear that increasing the K value produces a much smoother interpolated signal. However, as the K value grows, the 2-norm of the difference between the original signal and its interpolated version also increases. Adding more zeros introduces additional frequency components into the signal's spectrum, which raises the energy of the frequency spectrum. This, in turn, alters the DFT's frequency resolution, resulting in a less smooth frequency representation. Consequently, accurately representing the original signal's frequency components becomes harder, leading to a significantly larger difference between the interpolated and original signal frequencies.



```
2 norm of the difference between x2 and x: 6.144750
2 norm of the difference between x3 and x: 8.365213
2 norm of the difference between x4 and x: 23.499839
```

Figure 18: Calculated norms

From the data above, we observe that the x2 signal has a lower norm value. This indicates that the difference between the original signal and the x2 signal is smaller compared to the other interpolated signals. This further suggests that the x2 signal is the closest match to the original signal.

Appendix A

Matlab code for Harmonic Detection

```matlab
    % Load the signal data from a MAT file
load('signal11.mat', 'xn_test');

% Define the subsets of the signal with varying lengths
S1 = xn_test(1:129);
S2 = xn_test(1:257);
S3 = xn_test(1:513);
S4 = xn_test(1:1025);
S5 = xn_test(1:1793);

% Apply Discrete Fourier Transform (DFT) to each subset
dft_S1 = fftshift(fft(S1));
dft_S2 = fftshift(fft(S2));
dft_S3 = fftshift(fft(S3));
dft_S4 = fftshift(fft(S4));
dft_S5 = fftshift(fft(S5));

% Sampling frequency
fs = 128;

% Length of each subset
N1 = length(S1);
N2 = length(S2);
N3 = length(S3);
N4 = length(S4);
N5 = length(S5);

% Compute frequency axes for plotting
freq1 = linspace(-fs/2, fs/2, N1);
freq2 = linspace(-fs/2, fs/2, N2);
freq3 = linspace(-fs/2, fs/2, N3);
freq4 = linspace(-fs/2, fs/2, N4);
freq5 = linspace(-fs/2, fs/2, N5);

% Plot DFT of S1
figure('Position', [100, 100, 1000, 300]);
stem(freq1, abs(dft_S1));
title('DFT␣of␣S1');
xlabel('Frequency␣(Hz)');
ylabel('Amplitude');

% Plot DFT of S2
figure('Position', [100, 100, 1000, 300]);
stem(freq2, abs(dft_S2));
title('DFT␣of␣S2');
xlabel('Frequency␣(Hz)');
ylabel('Amplitude');

% Plot DFT of S3
```

```
figure('Position', [100, 100, 1000, 300]);
stem(freq3, abs(dft_S3));
title('DFT␣of␣S3');
xlabel('Frequency␣(Hz)');
ylabel('Amplitude');

% Plot DFT of S4
figure('Position', [100, 100, 1000, 300]);
stem(freq4, abs(dft_S4));
title('DFT␣of␣S4');
xlabel('Frequency␣(Hz)');
ylabel('Amplitude');

% Plot DFT of S5
figure('Position', [100, 100, 1000, 300]);
stem(freq5, abs(dft_S5));
title('DFT␣of␣S5');
xlabel('Frequency␣(Hz)');
ylabel('Amplitude');

% Parameters for averaging DFT
k = 128;  % Length of each subset
L = 14;   % Number of subsets
N = length(xn_test);

% Compute average DFT with k = 128
sum_dft = zeros(1, k);
for i = 1:L
    sub_x = xn_test((i-1)*k+1:i*k);  % Extract subset
    dft_sub_x = fft(sub_x);          % Compute DFT of subset
    sum_dft = sum_dft + dft_sub_x;   % Accumulate DFTs
end
avg_dft = sum_dft / L;  % Compute average DFT

% Plot average DFT with k = 128
figure('Position', [100, 100, 1000, 300]);
stem(abs(avg_dft));
title('Average␣DFT␣-␣k␣=␣128');
xlabel('Frequency␣(Hz)');
ylabel('Amplitude');
xlim([0, 64]);

% Compute and plot average DFT for varying L (1 to 14)
figure('Position', [100, 100, 1000, 300]);
for L = 1:14
    sum_dft = zeros(1, k);
    for i = 1:L
        sub_x = xn_test((i-1)*k+1:i*k);  % Extract subset
        dft_sub_x = fft(sub_x);          % Compute DFT of subset
        sum_dft = sum_dft + dft_sub_x;   % Accumulate DFTs
    end
    avg_dft = sum_dft / L;  % Compute average DFT

    % Plot average DFT for each L
    subplot(2, 7, L);
    stem(abs(avg_dft));
    title(['L␣=␣' num2str(L)]);
    xlim([0, 64]);
    xlabel('Frequency␣(Hz)');
    ylabel('Amplitude');
end

% Parameters for averaging DFT with k = 100
k = 100;
L = 14;
N = length(xn_test);

% Compute average DFT with k = 100
avg_dft = zeros(1, k);
for i = 1:L
    sub_x = xn_test((i-1)*k+1:i*k);  % Extract subset
    dft_sub_x = fft(sub_x);          % Compute DFT of subset
    sum_dft = sum_dft + dft_sub_x;   % Accumulate DFTs
end
avg_dft = sum_dft / L;  % Compute average DFT

% Plot average DFT with k = 100
figure('Position', [100, 100, 1000, 300]);
stem(abs(avg_dft));
title('Average␣DFT␣-␣k␣=␣100');
xlabel('Frequency␣(Hz)');
ylabel('Amplitude');
xlim([0, 50]);

% Parameters for averaging DFT with k = 135
k = 135;
N = length(xn_test);
L = floor(N / k);  % Number of subsets with k = 135

% Compute average DFT with k = 135
avg_dft = zeros(1, k);
for i = 1:L
    sub_x = xn_test((i-1)*k+1:i*k);  % Extract subset
    dft_sub_x = fft(sub_x);          % Compute DFT of subset
    sum_dft = sum_dft + dft_sub_x;   % Accumulate DFTs
end
avg_dft = sum_dft / L;  % Compute average DFT

% Plot average DFT with k = 135
figure('Position', [100, 100, 1000, 300]);
stem(abs(avg_dft));
title('Average␣DFT␣-␣k␣=␣135');
xlabel('Frequency␣(Hz)');
ylabel('Amplitude');
xlim([0, floor(k / 2)]);
```

Appendix B

Interpolation Function

```matlab
   function Xz = interpolation(X, K)
    % Interpolation function to add zeros between samples in a signal X.


    % Get the length of the input signal
    N = length(X);

    N1 = (N + 1) / 2;  % Midpoint for odd-length signal
    N2 = N / 2;        % Midpoint for even-length signal

    if mod(N, 2) == 0  % Check if the length of X is even
        Xz = [X(1:N2); X(N2 + 1) / 2; zeros((K * N) - 1, 1); X(N2 + 1) / 2; X((N2 + 2):N)];

    else  % Case for odd-length signal
        Xz = [X(1:N1); zeros(K * N, 1); X((N1 + 1):N)];

    end
end
```

Main Code to plot the signals

```matlab

% Loading the signal data from 'handel.mat'
load('handel')

% Creating sub-samples by down-sampling the original signal
N = 20000;                     % Number of samples to consider from the signal
x = y(1:N);                    % Original signal subset
x2 = x(1:2:N);
x3 = x(1:3:N);
x4 = x(1:4:N);

% Discrete Fourier Transform (DFT) of the sub-samples
dft_x2 = fft(x2);
dft_x3 = fft(x3);
dft_x4 = fft(x4);

% Interpolating the Signals in Frequency Domain
interpolated_x2 = interpolation(dft_x2, 1);
interpolated_x3 = interpolation(dft_x3, 2);
interpolated_x4 = interpolation(dft_x4, 3);

% Apply IDFT to get time-domain signals
ifft_x2 = ifft(interpolated_x2);
ifft_x3 = ifft(interpolated_x3);
```

```
ifft_x4 = ifft(interpolated_x4);

% Adjust the length of interpolated signals to match the
    original signal length
new_x2 = ifft_x2 * 2;           % Scale up the signal by 2
new_x3 = ifft_x3 * 3;           % Scale up the signal by 3
new_x4 = ifft_x4 * 4;           % Scale up the signal by 4

% Trim or extend the length of the signals to match the required
    length
new_x2 = new_x2(1:((2)*(length(x2)-1))+2);
new_x3 = new_x3(1:((3)*(length(x3)-1))+2);
new_x4 = new_x4(1:((4)*(length(x4)-1))+4);


figure('Position', [100, 100, 1200, 400]); % Create a figure
    window
subplot(3,1,1)
stem(y(1:50), 'red');            % Plot the original signal
title('Original␣Signal');
subplot(3,1,2)
stem(new_x2(1:50));             % Plot the interpolated version
    of x2
title('Interpolated␣version␣of␣x2');
subplot(3,1,3)
stem(new_x2(1:50));             % Plot the interpolated version
    of x2
hold on;
stem(y(1:50), 'red');           % Overlay the original signal
hold off;

figure('Position', [100, 100, 1200, 400]);
subplot(3,1,1)
stem(y(1:50), 'red');            % Plot the original signal
title('Original␣Signal');
subplot(3,1,2)
stem(new_x3(1:50));             % Plot the interpolated version
    of x3
title('Interpolated␣version␣of␣x3');
subplot(3,1,3)
stem(new_x3(1:50));             % Plot the interpolated version
    of x3
hold on;
stem(y(1:50), 'red');           % Overlay the original signal
hold off;

figure('Position', [100, 100, 1200, 400]);
subplot(3,1,1)
stem(y(1:50), 'red');            % Plot the original signal
title('Original␣Signal');
subplot(3,1,2)
stem(new_x4(1:50));             % Plot the interpolated version
    of x4
title('Interpolated␣version␣of␣x4');
```

```
subplot(3,1,3)
stem(new_x4(1:50));                % Plot the interpolated version of x4
hold on;
stem(y(1:50), 'red');              % Overlay the original signal
hold off;

% Calculate the 2-norm between the original signal and the interpolated signals
two_norm_x2 = norm(new_x2 - x);
fprintf('2-norm␣of␣the␣difference␣between␣x2␣and␣x:␣%f\n', two_norm_x2);
two_norm_x3 = norm(new_x3 - x);
fprintf('2-norm␣of␣the␣difference␣between␣x3␣and␣x:␣%f\n', two_norm_x3);
two_norm_x4 = norm(new_x4 - x);
fprintf('2-norm␣of␣the␣difference␣between␣x4␣and␣x:␣%f\n', two_norm_x4);
```