EN3551 - Digital Signal Processing

Assignment 01

Detecting Harmonics in Noisy Data and Signal Interpolation using DFT

Name: Samaramanna M. A.

Index Number: 220551K

Date: 12 September 2025

1 Harmonic Detection

1.1 Loading the signal from the signal set

Loading the signal using `load` function and plotting the actual signal. The sampling is done at a frequency of 128Hz and signal contains frequencies up to 64Hz.

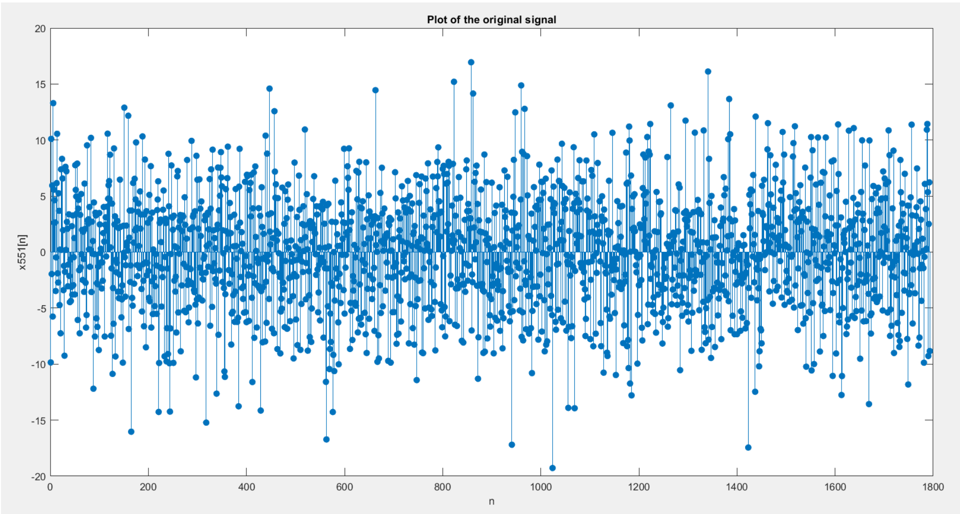
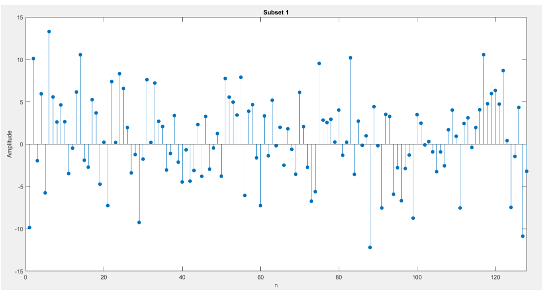
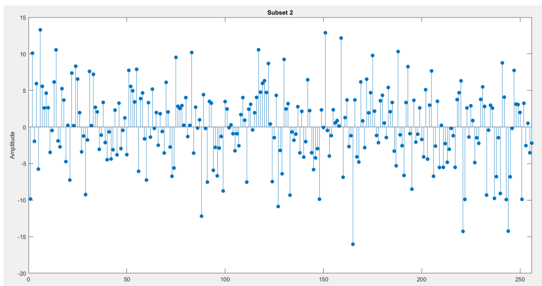


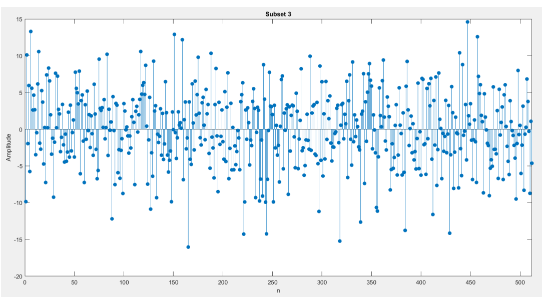
Figure 1: Original signal

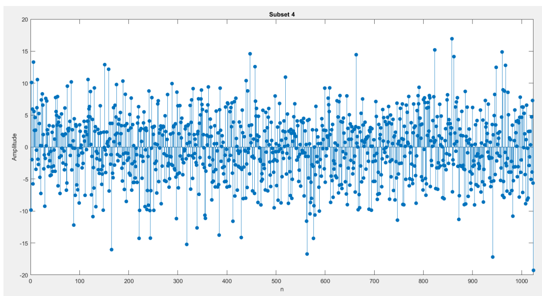
1.2 Constructing Subsets

Constructing several subsets by taking the first 128,256,512,1024 and 1792 samples from the sequence x[n] and denote them by S1, S2, S3, S4 and S5. Plots are stated below.









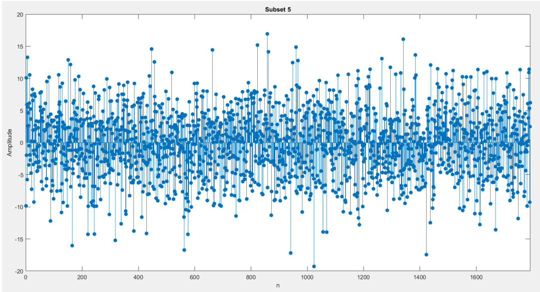


Figure 2: Plots of each subset S1, S2, S3, S4, S5

1.3 Apply DFT for each subsets and plot

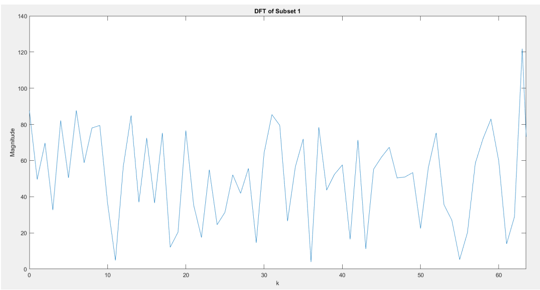
The N-point Discrete Fourier Transform (DFT) of a signal $x[n]$ is

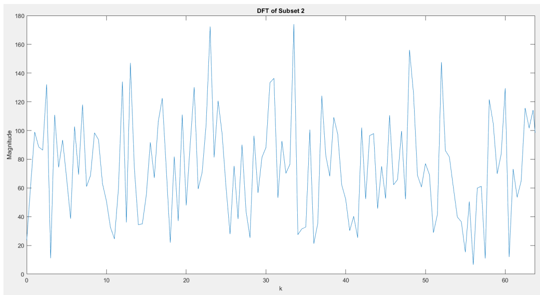
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N}, \qquad k = 0, 1, \ldots, N-1.$$

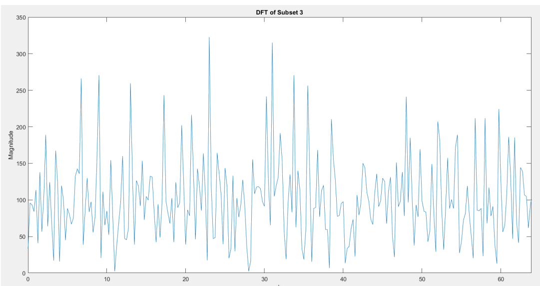
In this work, the DFT was computed using MATLAB's `fft` function, which implements the Fast Fourier Transform (FFT) algorithm. The FFT provides the same result as the DFT but with a much lower computational cost $\mathcal{O}(N \log N)$ compared to the direct $\mathcal{O}(N^2)$ summation. The frequency axis for plotting was generated using

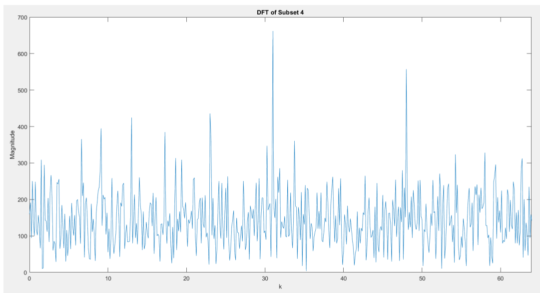
$$f_k = k \cdot \frac{f_s}{N}, \qquad 0 \leq k < N,$$

where f_s is the sampling frequency.









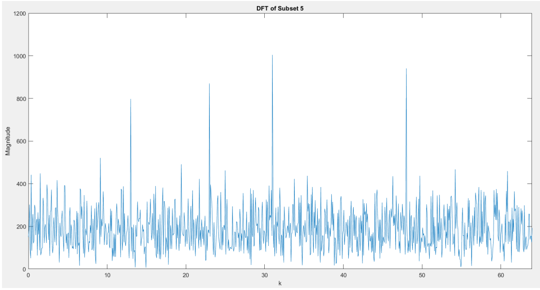


Figure 3: DFT magnitude plots belonging to each subset S1, S2, S3, S4, S5

Observation: As the DFT length K increases, the frequency resolution improves because the spacing between frequency bins becomes smaller. This results in sharper and more distinct spectral peaks, making it easier to identify sinusoidal components in the signal. However, because the signal is corrupted by noise, small peaks may still be masked by the noise floor even at higher resolutions.

1.4 DFT Averaging with $K = 128$ and $L = 14$

To overcome this limitation, DFT averaging is applied. The signal is divided into L equal-length segments (Here 14 segments of length 128), the DFT of each segment is computed, and the results are averaged in the complex domain. This averaging process reduces the random noise variance roughly in proportion to 1/L while preserving the coherent sinusoidal components, thereby making the hidden peaks more visible.

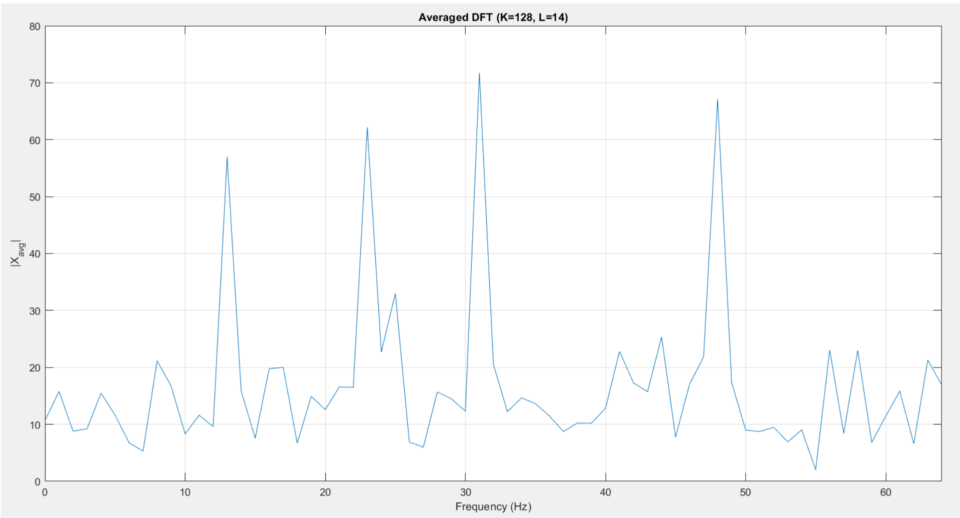


Figure 4: Frequency response after applying DFT averaging

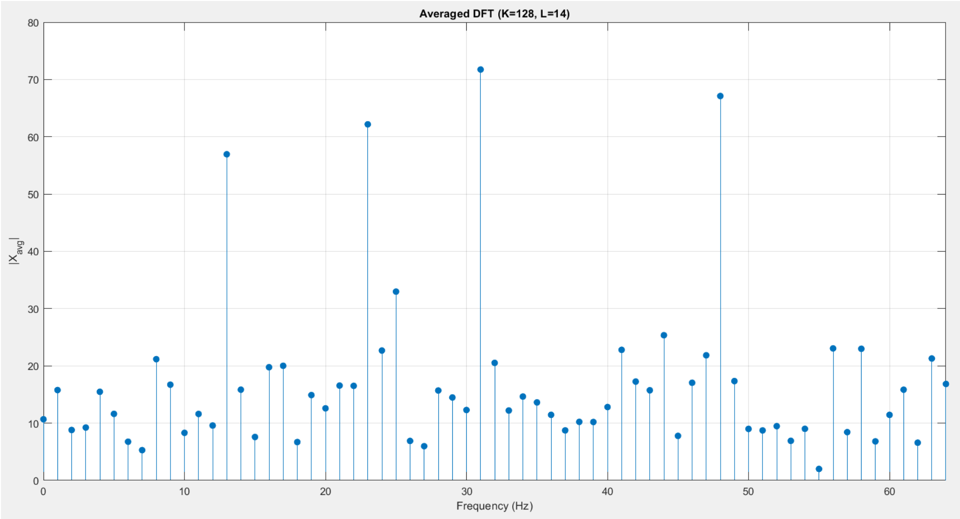


Figure 5: Frequency response after applying DFT averaging(as discrete samples

1.5 Minimum value for L

In the previous section, the signal was partitioned into L non-overlapping blocks of length K=128 samples, and the DFT of each block was computed. The complex DFTs were then averaged across all L blocks to form the averaged spectrum. To determine the minimum number of segments required for reliable harmonic detection, this procedure was repeated while incrementally increasing L from 1 to 14. As L increased, the variance of the noise floor decreased approximately by a factor of 1/L, causing the spectral peaks corresponding to the sinusoidal harmonics to become progressively more prominent. The smallest L was chosen as the one at which four distinct peaks were clearly visible below 64 Hz, ensuring that all harmonics were reliably detected with minimal computational effort.

The plots of frequency responses of L values, where we can clearly identify the harmonics below 64Hz are stated below.

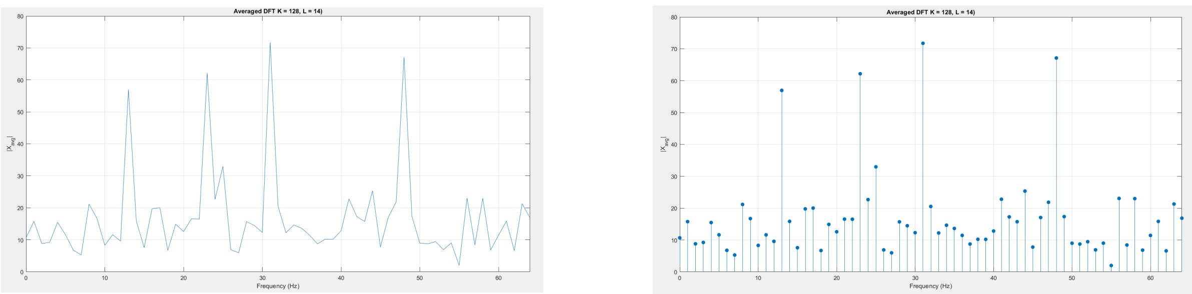
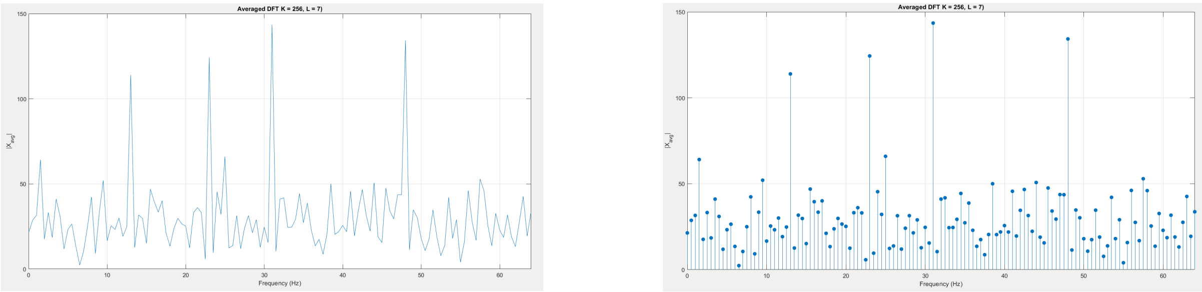
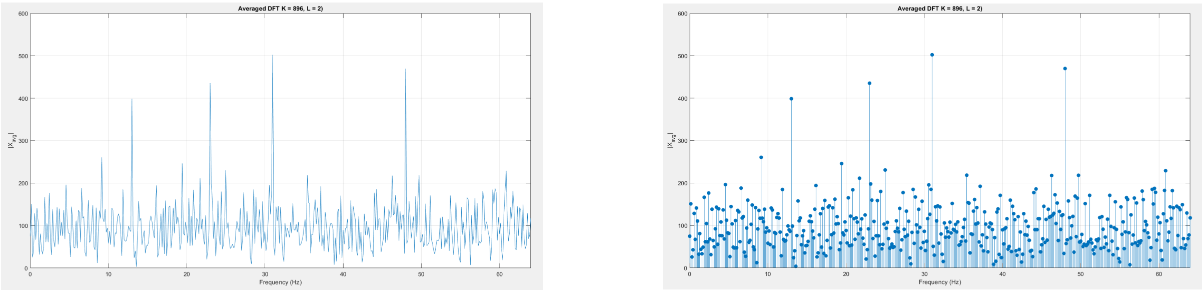


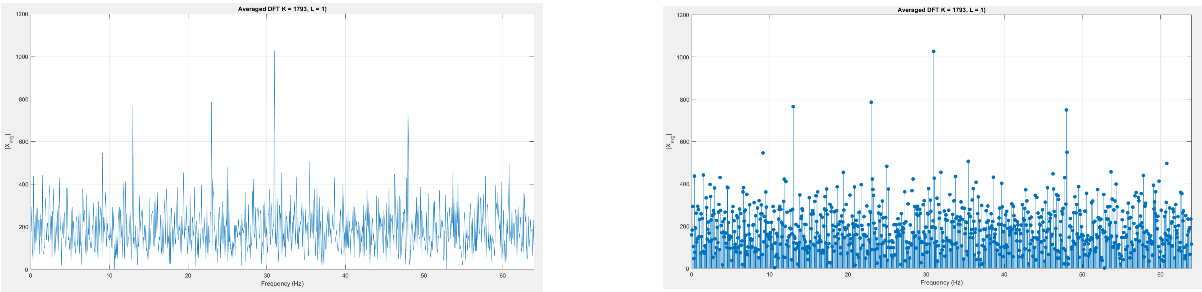
Figure 6: Frequency response after applying DFT averaging (K=14)



Frequency response after applying DFT averaging (K=7)



Frequency response after applying DFT averaging (K=2)



Frequency response after applying DFT averaging (K=1)

By observing the plots we can identify the four harmonics below 64Hz are **13Hz, 23Hz, 31Hz and 48Hz**, and the minimum value for L is 1.

1.6 Use other values for K (say K = 100 or K = 135)

When choosing the segment length K and the number of segments L for DFT averaging, it is important to ensure that these values fit well with the total signal length. If K is not a factor of the signal length, the last block may have fewer samples, leading to either discarded data or zero-padding, which can reduce the accuracy of the averaged spectrum. For example, in a signal of 1792 samples, choosing K=128 divides the signal evenly into 14 blocks, making it ideal for DFT averaging. In contrast, choosing K=100 or K=135 would leave incomplete blocks, causing a significant drop in effective sample usage and making them unsuitable for reliable harmonic detection. Therefore, K and L should be selected to maximize sample usage without compromising the spectral estimation.

2 Interpolation

2.1 Test Signal

We use the first 20,000 samples of Handel's *Hallelujah* available in MATLAB via the `load handel` command.

2.2 Generating the signals

Let the signal be $y[n]$ and

$$x = y[1 : 20000].$$

Downsampled versions are generated as ($N = 20000$)

$$x_2 = x(1 : 2 : N)$$

$$x_3 = x(1 : 3 : N)$$

$$x_4 = x(1 : 4 : N)$$

2.3 DFT-based Interpolation

The downsampled signals were interpolated using zero-padding in the frequency domain.
Results:

1. For x_2 interpolated with $K = 2$, the 2-norm error was small, and the interpolated waveform closely matched the original. The first 50 samples of both signals show strong agreement.

2. For x_3 interpolated with $K = 3$, the error increased, and some distortion appeared, although the general waveform shape was preserved.

3. For x_4 interpolated with $K = 4$, the error was the largest. Distortions were more pronounced since higher downsampling removed more information.

```
2-norm error for x2 interpolation = 6.1448
2-norm error for x3 interpolation = 8.3652
2-norm error for x4 interpolation = 23.4998
```

2.4 Plots

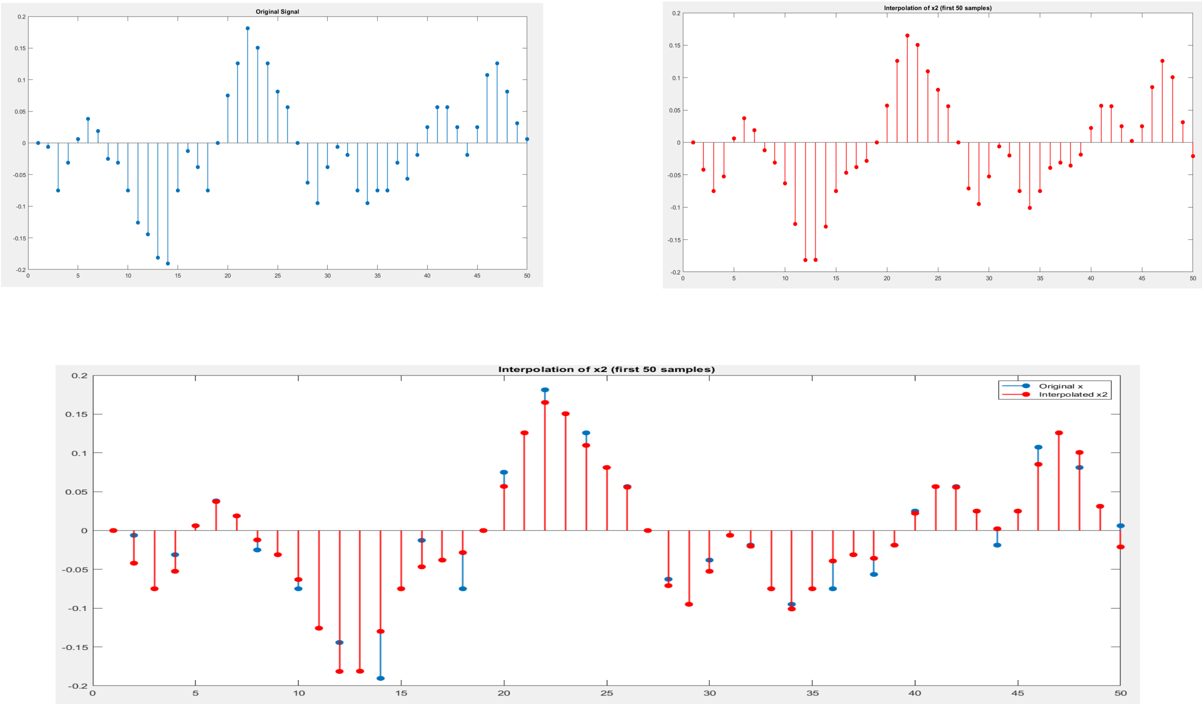


Figure 7: Original x signal (blue) and the signal interpolated using x2 (red)

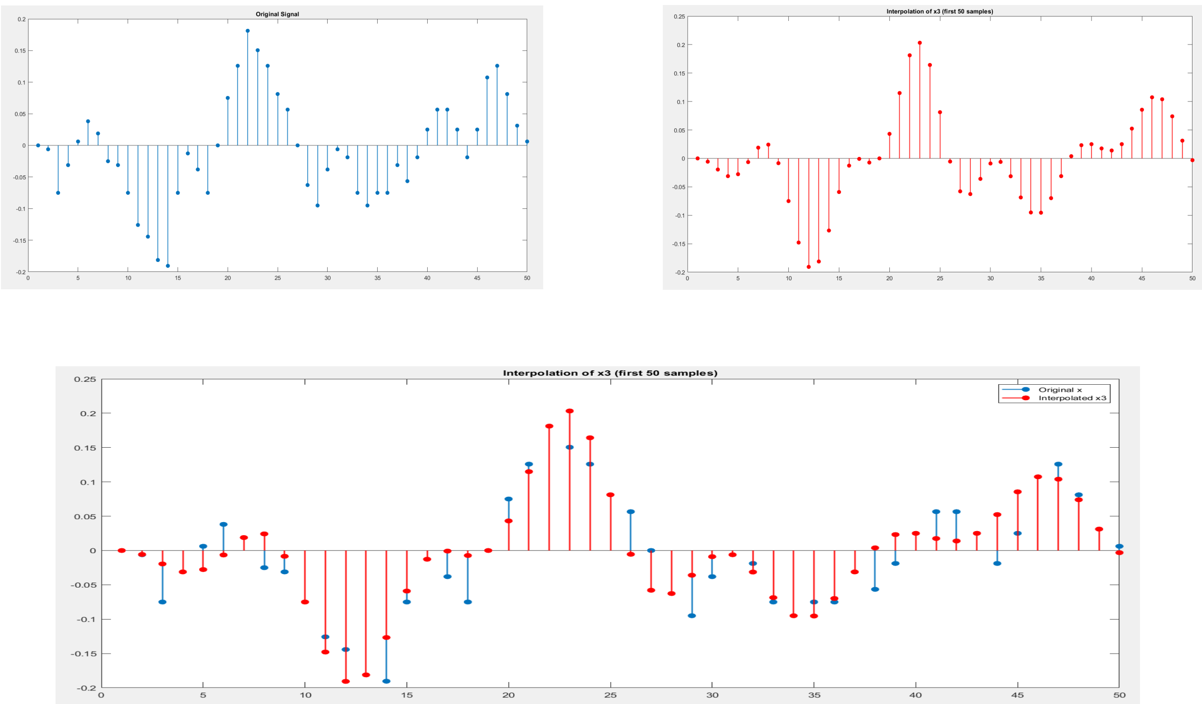
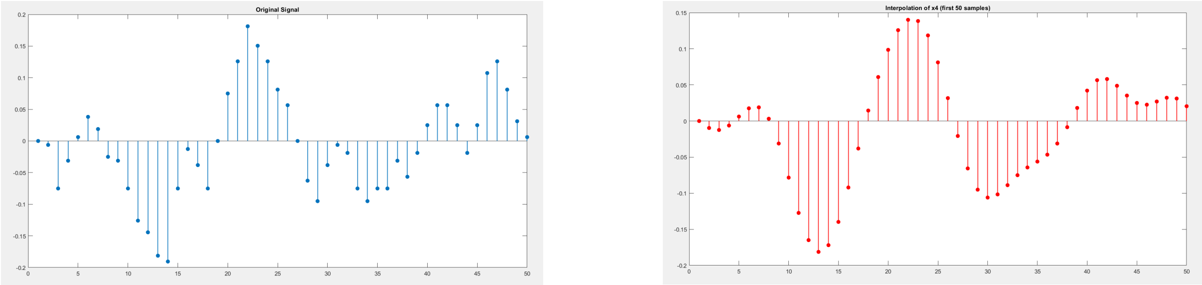


Figure 8: Original x signal (blue) and the signal interpolated using x3 (red)



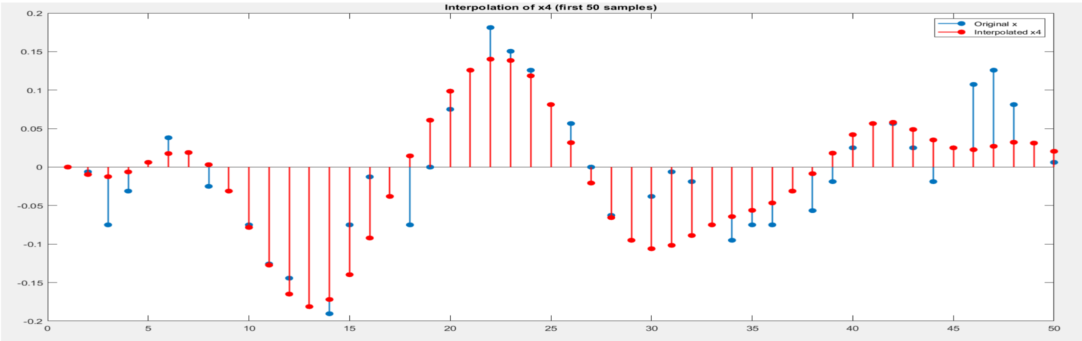


Figure 9: Original x signal (blue) and the signal interpolated using x4 (red)

A MATLAB Codes

A.1 Harmonic Detection

```
%load the signal and plot the original signal
load('signal551.mat','xn_test');
my_signal = xn_test(:);
signal_length = length(my_signal);
stem(my_signal, 'filled');
xlabel('n');
ylabel('x551[n]');
title('Plot of the original signal');

fs = 128;  %sampling rate is 128Hz
time = 14;    %samples were collected over a 14s time
no_of_samples = time*fs;
fprintf('Number of samples of the signal: %d\n', no_of_samples);

%making the subsets of the given signal
S1 = my_signal(1:128);
S2 = my_signal(1:256);
S3 = my_signal(1:512);
S4 = my_signal(1:1024);
```

```
S5 = my_signal(1:1792);

subsets = {S1, S2, S3, S4, S5};

plot each subset
for s=1:length(subsets)
    figure;
    stem(subsets{s}, 'filled');
    xlabel('n');
    xlim([0 length(subsets{s})])
    ylabel('Amplitude');
    title(sprintf('Subset %d', s));
end

for s=1:length(subsets)
    dft_signal = fft(subsets{s});
    mag_dft = abs(dft_signal);
    figure;
    stem(mag_dft, 'filled');
    xlabel('k');
    xlim([0 length(subsets{s})]);
    ylabel('Magnitude');
    title(sprintf('DFT of Subset %d', s));
end

for s=1:length(subsets)
    K = length(subsets{s});
    dft_signal = fft(subsets{s});
    mag_dft = abs(dft_signal);
    freq = (0:K-1)*(fs/K);
    figure;
    plot(freq(1:floor(K/2)+1), mag_dft(1:floor(K/2)+1));
    xlabel('k');
    xlim([0 max(freq)/2]);
    ylabel('Magnitude');
    title(sprintf('DFT of Subset %d', s));
end


DFT averaging
K = 128;
L = 14;

```

```
start_i = 1;
end_i = K;

X_sum = zeros(1,K);

% Apply DFT to each subset
for n=1:L
    subset = xn_test(start_i: end_i);
    subset_dft = fft(subset, K);
    % Update X_sum
    X_sum = X_sum + subset_dft;
    % Update indices
    start_i = end_i + 1;
    end_i = end_i + K;
end

X_a = abs(X_sum) / L;
disp(X_a)

f = (0:K-1)*(fs/K);

figure; plot(f(1:K/2+1), X_a(1:K/2+1));
xlabel('Frequency (Hz)'); ylabel('|X_{avg}|');
xlim([0 64]);
title('Averaged DFT (K=128, L=14)'); grid on;

figure; stem(f(1:K/2+1), X_a(1:K/2+1), 'filled');
xlabel('Frequency (Hz)'); ylabel('|X_{avg}|');
xlim([0 64])
title('Averaged DFT (K=128, L=14)'); grid on;


% Find minimum L
L_vals = 1:14;

for L=L_vals
    fprintf('L =  %d\n', L);
    K = floor(length(xn_test)/L);
    fprintf('K =  %d\n', K);
    start_index = 1;
    end_index = K;
    samples_count = K*L;
    fprintf('No of samples: %d\n', samples_count)
```

```


    X_sum = zeros(1,K);
    % DFT averaging for each iteration
    for n=1:L
        subset = xn_test(start_index:end_index);
        dft = fft(subset, K);
        %Update X_sum
        X_sum = X_sum + dft;
        %Update indices
        start_index = end_index + 1;
        end_index = end_index + K;
    end

    X_a = abs(X_sum) / L;

    f = (0:K-1)*(fs/K);

    figure;
    plot(f(1:floor(K/2)+1), X_a(1:floor(K/2)+1));
    xlabel('Frequency (Hz)'); ylabel('|X_{avg}|');
    xlim([0 64]);
    title(sprintf('Averaged DFT K = %d, L = %d)', K, L)); grid on
        ;

    figure;
    stem(f(1:floor(K/2)+1), X_a(1:floor(K/2)+1), 'filled');
    xlabel('Frequency (Hz)'); ylabel('|X_{avg}|');
    xlim([0 64])
    title(sprintf('Averaged DFT K = %d, L = %d)', K, L)); grid on
        ;

end
```

A.2 Interpolation

```
% Interpolation using DFT-based method
clc; clear; close all;

% Load the Handel signal
load handel;
N = 20000;
```

```

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

% Interpolation of x2 with K=1
x2_interp = dft_interpolate(x2, 2, N);  % x2 is downsampled by 2 interpolate by 2
err2 = norm(x - x2_interp, 2);            % 2-norm error
fprintf(2-norm error for x2 interpolation = %.4f\n , err2);

figure;
stem(1:50, x(1:50), 'filled', 'LineWidth', 1.2); hold on;
stem(1:50, x2_interp(1:50), 'filled', 'red', 'LineWidth', 1.2);
legend('Original x', 'Interpolated x2');
title('Interpolation of x2 (first 50 samples)');

% Interpolation of x3 with K=2
x3_interp = dft_interpolate(x3, 3, N);  % interpolate by 3
err3 = norm(x - x3_interp, 2);
fprintf(2-norm error for x3 interpolation = %.4f\n , err3);

figure;
stem(1:50, x(1:50), 'filled', 'LineWidth', 1.2); hold on;
stem(1:50, x3_interp(1:50), 'filled', 'red', 'LineWidth', 1.2);
legend('Original x', 'Interpolated x3');
title('Interpolation of x3 (first 50 samples)');

% Interpolation of x4 with K=3
x4_interp = dft_interpolate(x4, 4, N);  % interpolate by 4
err4 = norm(x - x4_interp, 2);
fprintf(2-norm error for x4 interpolation = %.4f\n , err4);

figure;
stem(1:50, x(1:50), 'filled', 'LineWidth', 1.2); hold on;
stem(1:50, x4_interp(1:50), 'filled', 'red', 'LineWidth', 1.2);
legend('Original x', 'Interpolated x4');
title('Interpolation of x4 (first 50 samples)');

% Function: DFT-based interpolation
% x_down : downsampled signal
% K      : interpolation factor
```

```
% N_full : original signal length
function x_interp = dft_interpolate(x_down, K, N_full)
    M = length(x_down);           % length of downsampled signal
    % Take DFT of downsampled signal
    X = fft(x_down);

    % Zero-padding in frequency domain
    if mod(M,2) == 0
        % Even-length
        Xzp = [X(1:M/2); zeros((K-1)*M,1); X(M/2+1:end)];
    else
        % Odd-length
        Xzp = [X(1:(M+1)/2); zeros((K-1)*M,1); X((M+1)/2+1:end)];
    end

    % Inverse DFT to get interpolated signal
    x_interp = real(ifft(Xzp))*K;

    % Trim or pad to match original length
    if length(x_interp) > N_full
        x_interp = x_interp(1:N_full);
    else
        x_interp = [x_interp; zeros(N_full-length(x_interp),1)];
    end
end
```