Assignment 02

submitted for

EN3551 - Digital Signal Processing

Department of Electronic and Telecommunication Engineering
University of Moratuwa

Udugamasooriya P. H. J.
220658U
Progress on GitHub

12 September 2025

1 Applying 2D DCT for Image Compression

Question 03 Five subsets of the provided signal were formed as described. The magnitude plots of the DFTs of each of these subsets are indicated in Figure 1.

Figure 1: DFT-magnitude plots of signals S_1 through S_5

To understand the differences between the DFTs above, we start by noting that an N-point signal $x[n]$ in the time domain will have an N-point DFT $X[k]$ in the frequency domain, related together through the inverse DFT relation as follows;

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[n] \cdot e^{j\frac{2\pi k}{N}n}, \text{ for } n = 0, \ldots, N-1.$$

We interpret the above expression as a decomposition of $x[n]$ into a linear combination of N complex exponentials

$$e^{j\omega_k n} = e^{j\frac{2\pi k}{N}n} \; (k = 1, \ldots, N-1),$$

where

$$\omega_k = \frac{2\pi k}{N}.$$

Notice that with bigger N, the frequencies of the complex exponentials that $x[n]$ is decomposed into become more closely and finely spaced. This means that a higher-order DFT can detect frequency content that simply goes undetected with a lower-order DFT.

Further, there are effects of spectral leakage; when we have to work with a lower resolution frequency axis, frequency components that cannot be detected exactly might “leak” into adjacent frequency values, causing a spreading out out frequencies, further concealing the actual harmonics present.

The variation of the profundity of these effects are clearly visible in the plots above. It is most obvious with S_1, where no clear distinct peaks are visible, but as the number of samples is increased, certain peaks start to stand out more prominently.

For the problem of what harmonics are actually present, we look for the four most prominent spikes in the DFT of S_5; the signal whose DFT has the most clearly distinguishable spikes. These spikes are indicated in Figure 2.

Figure 2: Spikes in the DFT-magnitude plot of S_5

We conclude therefore that the harmonics present are **14 Hz**, **23 Hz**, **38 Hz** and **48 Hz**.

Question 04 DFT averaging is implemented in Listing 5 in Appendix A.

Figure 3 shows the magnitude plot resulting from averaging the DFTs of $L = 14$ consecutive subsets taken from the given signal, with the most prominent spikes selected.

Figure 3: Magnitude plot of the averaged DFT over several subsets

We observe that the above plot is a much clearer plot, with the same frequencies **14 Hz**, **23 Hz**, **38 Hz** and **48 Hz** as detected before standing out prominently.

To analyze this procedure, let us start by denoting the clean signal by $s[n]$ and the Gaussian noise (assumed AWGN) by $w[n]$, so that the observed signal, which we will denote $x[n]$ is given by

$$x[n] = s[n] + w[n], \text{ for } n = 1, \ldots, N-1,$$

where in our case, $N = 1792$. Note that $w[n]$ is a random variable for each n; we assume that all the $w[n]$'s are independent, normally distributed (iid) random variables, with zero mean and some variance σ^2. It follows then that $x[n]$ is also a random variable for each n.

Denote by $X\left(e^{j\omega}\right)$ the following:

$$X\left(e^{j\omega}\right) = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\omega n},$$

and let $S\left(e^{j\omega}\right)$ and $W\left(e^{j\omega}\right)$ be defined similarly. Note that $S\left(e^{j\omega}\right)$ is precisely the DTFT of $s[n]$, whereas $X\left(e^{j\omega}\right)$ and $W\left(e^{j\omega}\right)$ are linear combinations of iid Gaussian random variables.

Now, observe that

$$X\left(e^{j\omega}\right) = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\omega n} = \sum_{n=0}^{K-1} x[n] \cdot e^{-j\omega n} + \sum_{n=K}^{2K-1} x[n] \cdot e^{-j\omega n} + \cdots + \sum_{n=\frac{N}{K}-1}^{N-1} x[n] \cdot e^{-j\omega n},$$

where we break the sum defining $X\left(e^{j\omega}\right)$ into smaller sums taken over smaller subsets of $x[n]$, each having K samples. Let us denote a general term in the above sum by

$$X_a\left(e^{j\omega}\right) = \sum_{n=aK}^{(a+1)K-1} x[n] \cdot e^{-j\omega n},$$

so that

$$X\left(e^{j\omega}\right) = \sum_{a=0}^{\frac{N}{K}-1} X_a\left(e^{j\omega}\right).$$

Note that

$$\begin{aligned} X_a\left(e^{j\omega}\right) &= \sum_{n=aK}^{(a+1)K-1} x[n] \cdot e^{-j\omega n} \\ &= \sum_{n=0}^{K-1} x[n+aK] \cdot e^{-j\omega(n+aK)} \\ &= e^{-j\omega aK} \sum_{n=0}^{K-1} x[n+aK] \cdot e^{-j\omega n}. \end{aligned}$$

By letting

$$X_{aK}\left(e^{j\omega}\right) = \sum_{n=0}^{K-1} x[n+aK]\cdot e^{-j\omega n},$$

and putting all this together, using linearity properties, we find that

$$\begin{aligned}
S\left(e^{j\omega}\right) &= X\left(e^{j\omega}\right) - W\left(e^{j\omega}\right)\\
\sum_{a=0}^{\frac{N}{K}-1} S_a\left(e^{j\omega}\right) &= \sum_{a=0}^{\frac{N}{K}-1} X_a\left(e^{j\omega}\right) - \sum_{a=0}^{\frac{N}{K}-1} W_a\left(e^{j\omega}\right)\\
\sum_{a=0}^{\frac{N}{K}-1} e^{-j\omega aK}\cdot S_{aK}\left(e^{j\omega}\right) &= \sum_{a=0}^{\frac{N}{K}-1} e^{-j\omega aK}\cdot X_{aK}\left(e^{j\omega}\right) - \sum_{a=0}^{\frac{N}{K}-1} e^{-j\omega aK}\cdot W_{aK}\left(e^{j\omega}\right).
\end{aligned}$$

Setting $\omega = \dfrac{2\pi k}{K}$ $(k = 1,\dots,K-1)$, we obtain

$$S\left(e^{j\frac{2\pi k}{K}}\right) = \sum_{a=0}^{\frac{N}{K}-1} \cdot S_{aK}\left(e^{j\omega}\right) = \sum_{a=0}^{\frac{N}{K}-1} X_{aK}\left(e^{j\frac{2\pi k}{K}}\right) - \sum_{a=0}^{\frac{N}{K}-1} W_{aK}\left(e^{j\frac{2\pi k}{K}}\right),$$

because $e^{-j\frac{2\pi k}{K}aK} = e^{-j2\pi ak} = 1$, for any integer a and k.

Define $X[k]$ for $k = 1,\dots,K-1$ by

$$X_{aK}[k] = \sum_{n=0}^{K-1} x[n+aK]\cdot e^{-j\frac{2\pi}{K}kn},$$

and let $S[k]$ and $W[k]$ also be defined similarly. Noting that $X_{aK}[k] = X_{aK}\left(e^{j\frac{2\pi}{K}k}\right)$, we can then also write

$$S\left(e^{j\frac{2\pi k}{K}}\right) = \sum_{a=0}^{\frac{N}{K}-1} \cdot S_{aK}[k] = \sum_{a=0}^{\frac{N}{K}-1} X_{aK}[k] - \sum_{a=0}^{\frac{N}{K}-1} W_{aK}[k].$$

Finally, dividing through by $\dfrac{N}{K}$ yields

$$\frac{K}{N} S\left(e^{j\frac{2\pi k}{K}}\right) = \frac{K}{N}\sum_{a=0}^{\frac{N}{K}-1} \cdot S_{aK}[k] = \frac{K}{N}\sum_{a=0}^{\frac{N}{K}-1} X_{aK}[k] - \frac{K}{N}\sum_{a=0}^{\frac{N}{K}-1} W_{aK}[k].$$

But, note that

$$W_{aK}[k] = \sum_{n=0}^{K-1} w[n+aK]\cdot e^{-j\frac{2\pi k}{K}n},$$

where the $w[n+aK]$'s are iid Gaussian random variables. Hence, it follows that $W_{aK}[k]$ is itself a Gaussian random variable. It is easy to see that due to the iid assumption, the $W_{aK}[k]$'s are also iid, for each a and k.

We note specifically that each $W_{aK}[k]$ is Gaussian with mean

$$\mathbb{E}\left(W_{aK}[k]\right) = \sum_{n=0}^{K-1} \mathbb{E}\left(w[n+aK]\right)\cdot e^{-j\frac{2\pi k}{K}n} = 0,$$

and we know that the sample mean over any finite set of observed $W_{aK}[k]$'s is an unbiased estimator of this mean, whose value approaches the true mean as the number of observations increases.

The term

$$\frac{K}{N}\sum_{a=0}^{\frac{N}{K}-1} W_{aK}[k]$$

in the expression derived above is exactly the sample mean of some observed values of $W_{aK}[k]$'s once the signal $x[k]$ is given, and

$$\frac{K}{N}\sum_{a=0}^{\frac{N}{K}-1} X_{aK}[k]$$

is the average of $\dfrac{N}{K}$ K-point DFTs taken over $\dfrac{N}{K}$ consecutive subsets of $x[n]$.

Hence, we have shown that the averaging process lets us approximate $S\left(e^{j\frac{2\pi k}{K}}\right)$ from an AWGN-corrupted signal by acting to reduce the impact of AWGN.

Question 05 Reducing L while holding K constant has the effect of dropping out samples from the signal. Plots obtained for different values of L starting from 13 downwards are shown in Figure 4.

Figure 4: DFT averaging with various values of L for $K = 128$

The four main peaks corresponding to the harmonics present remain prominent until $L = 7$. At $L = 6$, five prominent peaks are visible; one more close to zero. Hence, the smallest value suitable for L with $K = 128$ is $L = 7$.

Question 06 Values of K such that there exists an integer L for which $KL = 1792$ may be used, if it is desired to use all samples of the signal for analysis, i.e., factors of 1792. Other values of K may be used, but with different integer L such that $KL < 1792$. Doing this will result in only fewer of the samples being used for the computation, leading to reduced accuracy in the results.

2 Interpolation

Question 01 A plot of the first 50 samples of the loaded original signal is given in Figure 5.

Figure 5: Original signal

Question 03 Steps (a) through (c) were carried out using the code in Listing 6 in Appendix A. Outputs obtained from the code and relevant plots are as shown in Listings 1, 2, 3 and Figures 6, 7, and 8 respectively.

In each figure, we plot the interpolated version of the signal first, and then superimpose it on the original signal to allow for better comparison.

An analysis of the results obtained follows.

(a)

Figure 6: x_2 Interpolated

```
Norm of difference: 6.1447
```

Listing 1: Code output, $\|x_2 - x\|$

(b)

Figure 7: x_3 Interpolated

```
Norm of difference: 8.3652
```

Listing 2: Code output, $\|x_3 - x\|$

(c)

Figure 8: x_4 Interpolated

```
Norm of difference: 23.4998
```

Listing 3: Code output, $\|x_4 - x\|$

(d) Figure 9 shows a plot of the original signal, each interpolated signal separately, and finally, a superimposition of all of the above.

Figure 9: Comparison of original signal against all interpolated versions

It is clear that the interpolated signal has become less and less of an accurate representation of the actual signal, the more and more zeros we insert to the frequency axis.

This is evident from the increase in the norms of the differences between the original and interpolated versions.

As can be seen from the plots above, the interpolated version reconstructed from x_4 shows a lot of smooth variation; with too few samples to interpolate from, rapid variations in the time domain signal cannot be reconstructed.

However, the reconstruction from x_2 follows the signal very closely. This uses more samples from the signal and therefore can be used to reconstruct the signal more accurately.

A Code Snippets

A.1 Harmonic Detection

```
clear;
clc;
close all;

% 3.1.1 - Load the relevant signal
load('signal658.mat');

% Define useful constants
fs = 128; % Sampling frequency (given)
Ns = [128 256 512 1024 1792]; % Sizes of subsets to be formed (given)

% For plotting
figure(1);
tiledlayout(3, 2, "TileSpacing", "compact", "Padding", "compact");

% Try to detect harmonics from the DFTs of increasingly bigger subsets of
% the signal
```

```
for i = 1:5
       N_i = Ns(i); % Number of samples in the ith subset
       f_i = [0:(N_i-1)] * fs / N_i; % Frequency axis corresponding to the ith subset

       % 3.1.2 - Form the ith subset s_i
       s_i = xn_test(1:N_i);

       % 3.1.3 - Obtain the DFT S_i of the ith subset, and then its magnitude
       S_i = fft(s_i);
       S_i_mag = abs(S_i);

       % We will plot the first four figures in a 2x2 configuration, with the
       % last one in a row of its own
       if i <= 4
       nexttile(i);
       else
       nexttile(i, [1, 2]);
       end

       % 3.1.3 - Plot the magnitudes of the DFT of the ith signal
       stem(f_i, S_i_mag, "Marker", "o", "MarkerSize", 3, "MarkerFaceColor", "auto");

       title("DFT-Magnitude of $S_" + i + "$", "Interpreter", "latex");
       xlabel("Frequency (Hz)", "Interpreter", "latex");
       xlim([0 f_i(N_i)]);
       grid on;
       grid minor;
end

% Try to detect harmonics by averaging

L = 14; % Number of subsets
K = 128; % Number of samples in a subset

% 3.1.4 - Find the average DFT from L = 14 consecutive subsets of K = 128
% samples each, and then obtain its magnitude
X_avg = dft_average(xn_test, L, K);
X_avg_mag = abs(X_avg);

f = [0:(K-1)] * fs / K; % Define the frequency axis

% Plot the magnitude of the average DFT
figure(2);

stem(f, X_avg_mag, "Marker", "o", "MarkerSize", 3, "MarkerFaceColor", "auto");

title("Magnitude of Averaged DFTs", "Interpreter", "latex");
xlabel("Frequency (Hz)", "Interpreter", "latex");
xlim([0 f(K)]);
grid on;
grid minor;

% 3.1.5 - What is the smallest value of L such that the peaks remain
% visible?

figure(3);
tiledlayout(3, 2);

% We will try smaller values L_s for L, starting from 1 less than the last
% used value
for L_s = [13 10 8 7 6 5]
        % Obtain the average DFT for the chosen pair L_s, K
        X_avg_ = dft_average(xn_test, L_s, K);
        X_avg_mag_ = abs(X_avg_);

        nexttile;

        % Plot
        stem(f, X_avg_mag_, "Marker", "o", "MarkerSize", 3, "MarkerFaceColor", "auto");

        title("Average DFT Magnitude, $L=" + L_s + "$, $K=" + K + "$", "Interpreter", "
            latex");
        xlabel("Frequency (Hz)", "Interpreter", "latex");
        xlim([0 f(K)]);
        grid on;
        grid minor;
end
```

Listing 4: Main Code

```
function [result] = dft_average(x, L, K)
% DFT_AVERAGE Average DFT of L consecutive subsets of K samples each from x
% x = time-domain signal
% L = number of subsets
% K = number of samples from each subset
%
% Computes the DFT of L consecutive K-point signal formed by partitioning
% x, and returns the average; i.e., sum of the DFTs divided by L

% We compute the DFTs of K-point signals; they will have K samples
% themselves; construct a dummy K-dimensional array to store the result
result = complex(zeros(1, K));

for j = 0:(L-1)
        x_j = x(j*K+1:(j+1)*K); % jth subset
        X_j = fft(x_j); % DFT of the jth subset

        % Keep accumulating the DFT over subsets
        result = result + X_j;
end

% Average the DFT
result = result / L;

end
```

Listing 5: DFT averaging

A.2 Interpolation

```
clear;
clc;
close all;

% 3.2.1 - Load the signal
load handel;

% We will only be looking at the first 20,000 samples of the signal
N = 20000;
x_1 = y(1:N);

% Visualize the original signal
figure;

stem(x_1(1:50), "Marker", "o", "MarkerSize", 3, "MarkerFaceColor", "auto");

title("Original Signal", "Interpreter", "latex");
grid on;
grid minor;

% Empty array to store interpolations for later comparison
interpolations = {};

for i = 2:4
       % 3.2.2 - Form the signal x_i as described
       x_i = y(1:i:N);

       % 3.2.3 -
       % Obtain an interpolated version of x_i from the zero-padded DFT of x_i
       % with K = i - 1
       x_i_interpolated = interpolate(zero_padded_dft(x_i, i-1), i-1);

       % Compute the norm of the difference between the original signal and
       % the interpolated version
       disp("Norm of difference: " + difference_norm(x_i_interpolated, x_1));

       % Plot the signals for comparison
       plot_interpolation_v_actual(x_i_interpolated, x_1);

       % Save the current interpolated version for later comparison
       interpolations{end+1} = x_i_interpolated;
end

% Plot all interpolated signals together, and on top of each other with the
% original signal for comparison
figure;

% Plot the original signal
subplot(5, 1, 1);

stem(x_1(1:50), "Marker", "o", "MarkerSize", 3, "MarkerFaceColor", "auto");
```

```
title("Original Signal", "Interpreter", "latex");
grid on;
grid minor;

% Plot each interpolated signal in succession
for i = 1:3
       subplot(5, 1, i + 1);

       interpolation_i = interpolations{i};
       stem(interpolation_i(1:50), "Marker", "o", "MarkerSize", 3, "MarkerFaceColor", "
           auto");

       title("$x_" + (i+1) + "$ Interpolated", "Interpreter", "latex");
       grid on;
       grid minor;
end

% Plot all interpolations and the original signal on top of each other
subplot(5, 1, 5);

stem(x_1(1:50), "Marker", "o", "MarkerSize", 3, "MarkerFaceColor", "auto");
hold on;

for j = 1:3
       interpolation_j = interpolations{j};
       stem(interpolation_j(1:50), "Marker", "o", "MarkerSize", 2+j);
       hold on;
end

title("Comparison", "Interpreter", "latex");
legend("Original Signal", "$x_2$ Interpolated", "$x_3$ Interpolated", "$x_4$
    Interpolated", "Interpreter", "latex");
grid on;
grid minor;

% Helper Functions

function [] = plot_interpolation_v_actual(interpolation, actual)
       figure;

       % Plot the interpolated signal
       subplot(2, 1, 1);

       stem(interpolation(1:50), "Marker", "o", "MarkerSize", 3, "MarkerFaceColor", "
           auto");

       title("Interpolated Signal", "Interpreter", "latex");
       grid on;
       grid minor;

       % Plot the interpolated signal on top of the actual signal for
       % comparison
       subplot(2, 1, 2);
```

```
        stem(interpolation(1:50), "Marker", "o", "MarkerSize", 3, "MarkerFaceColor", "
            auto");
        hold on;
        stem(actual(1:50), "Marker", "o", "MarkerSize", 5);

        title("Interpolated Signal vs. Actual Signal", "Interpreter", "latex");
        legend("Interpolated Signal", "Actual Signal");
        grid on;
        grid minor;
end

function [result] = difference_norm(x1, x2)
        % Find the length of the shorter signal
        N = min(length(x1), length(x2));

        % Truncate both signals to the same length
        x1_trunc = x1(1:N);
        x2_trunc = x2(1:N);

        % Compute the norm of the difference between the truncated signals
        result = norm(x1_trunc - x2_trunc);
end
```

Listing 6: Main Code

```
function [result] = zero_padded_dft(x, K)
% ZERO_PADDED_DFT Obtain an (K+1)N-point DFT of x with zeros inserted in
% the middle
%
% x = time-domain signal, of length N
% K = sets the length of the resulting DFT
%
% Compute the DFT of x, split it in the middle, insert zeros between the
% two halves, and return the result. Splitting and other manipulations are
% done depending on the parity of N.

% Start by obtaining the DFT of x
X = fft(x);
N = length(X);

if mod(N, 2) % If X has an odd number of samples
        n = (N+1)/2; % Index of the point to split from

        % Insert KN zeros as described
        result = [
                X(1:n);
                complex(zeros(K*N, 1));
                X((n+1):N)
        ];
else % If X has an even number of samples
        n = N/2; % Index of the point to split from

        % Insert KN - 1 zeros as described
        result = [
                X(1:n);
                X(n+1)/2;
                complex(zeros((K*N)-1, 1));
                X(n+1)/2;
                X((n+2):N)
        ];
end

end
```

Listing 7: Zero-padding DFT

```
function [result] = interpolate(X, K)
%INTERPOLATE

N = length(X)/(K+1);

x = (K+1) * ifft(X);

result = x(1:((K+1)*(N-1)+1));

end
```

Listing 8: Interpolation