SIO 207A Final Project Report

Ruipu Ji

I. Data Set

This report provides an implementation of the complex basebanding and Fast Fourier Transform (FFT) architecture as shown in Figure 1. A discrete-time signal $x(n)$ with 4096 data points (n = 0, 1, 2…4095) is selected for analysis. The discrete-time signal $x(n)$ has the analog form $x(t)$ as defined in the following equation:

$$x(t) = \sum_{l=1}^{3} A_l \cos(2\pi f_l t + \phi_l)$$

where the parameters are defined in Table 1. The sampling frequency of the signal is f_s = 1000 Hz.

Table 1. Parameters used in the analog form $x(t)$ for the discrete-time signal $x(n)$.

l	A_l	f_l	Φ_l
1	100	160	0
2	10	237	0
3	1	240	0

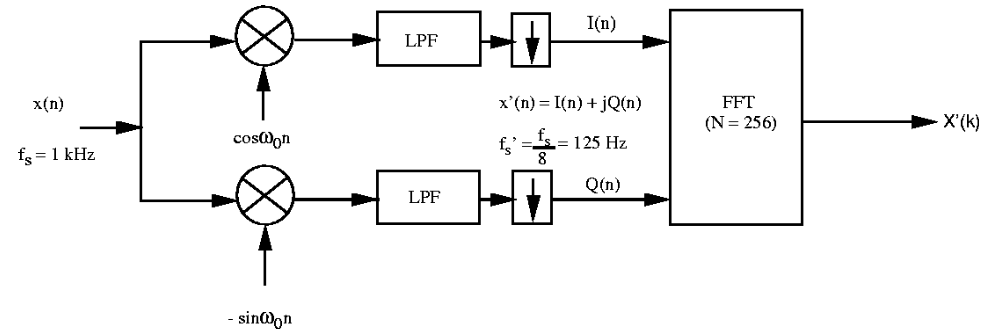

Figure 1. Complex basebanding and Fast Fourier Transform (FFT) architecture.

The first 256 data points of the discrete-time signal $x(n)$ are plotted in Figure 2. The logarithmic magnitude of its Fourier Transform using a Kaiser-Bessel window (α=2.5 or β=7.85) with NFFT = 256 is also presented in Figure 2. The analog frequency bin width of the FFT is 3.906 Hz, which is calculated based on the procedures as presented in Appendix 1. The bin indices for each analog frequency component are summarized in Table 2. These analog frequency components reside in 4 different frequency bins, which correspond to the 4 different peaks in the FFT magnitude plot. Because of the large analog frequency bin width (3.906 Hz), frequency components f_2 and f_3 both reside in bin #61. The two frequency components cannot be identified separately, so only one peak is observed for frequency components f_2 and f_3 (similarly $-f_2$ and $-f_3$ are also observed as one peak). Given the sampling frequency of 1000 Hz and NFFT = 256, the frequency bin indices for f_2 and f_3 are always 61, which is not affected by the window applied to the signal. Therefore, rectangular windowed FFT will not indicate distinct spectral peaks for f_2 and f_3.

Table 2. Frequency components of the original signal.

f_l [Hz]	Bin Index	$\lvert X(f_l)\rvert$ [dB]	f_l [Hz]	Bin Index	$\lvert X(f_l)\rvert$ [dB]
160	41	0	-160	-41	0
237	61	-20	-237	-61	-20
240	61	-20	-240	-61	-20

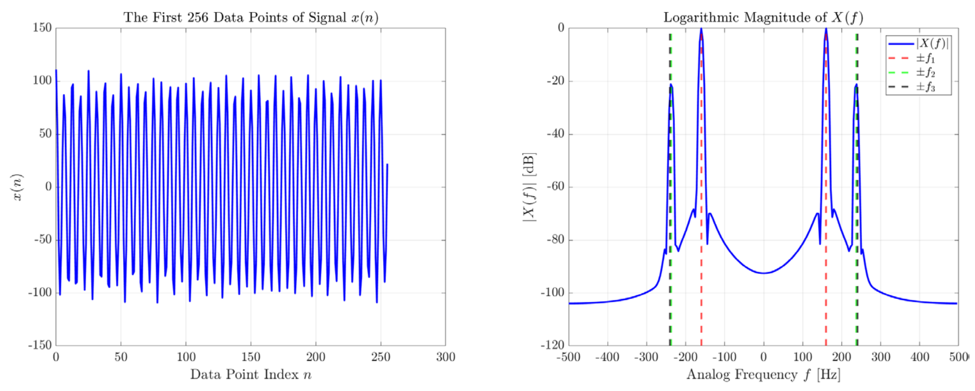

Figure 2. The first 256 data points of signal $x(n)$ and the logarithmic magnitude of its Fourier Transform $X(f)$.

II. Decimation Filter Design

A decimation filter is designed based on the equiripple FIR filter design algorithm. The filter is a 64-coefficient linear phase FIR low-pass filter with passband cutoff frequency of 40 Hz and stopband cutoff frequency of 85 Hz. The passband/stopband weight ratio of the filter is 50. Figure 3 shows the impulse response $h(n)$ and frequency response $H(f)$ of the decimation filter. The frequency response $H(f)$ is calculated by applying the Fourier Transform on the impulse response $h(n)$ using a rectangular window with NFFT = 1024. An expanded view of $H(f)$ with analog frequency -40 Hz $\leq f \leq$ 40 Hz (within passband frequency range) is also presented in Figure 3. Smooth magnitude is observed for the passband magnitude. The magnitude of the passband ripple is less than 4 dB.

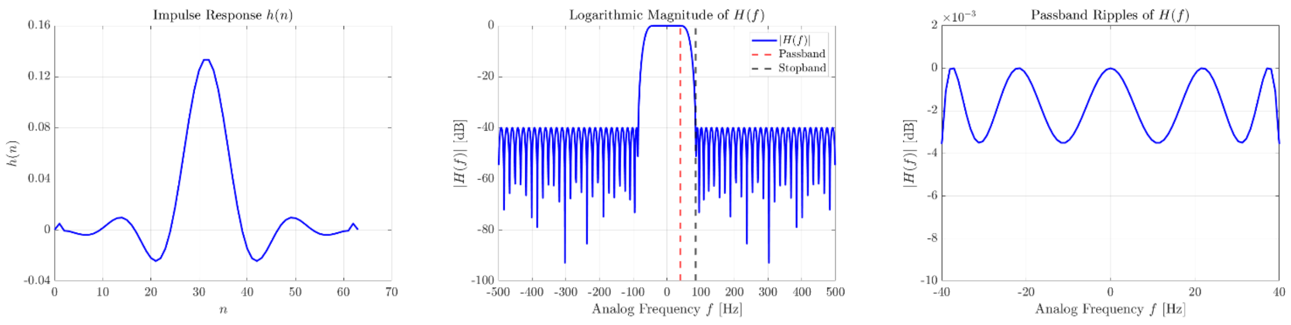

Figure 3. Impulse response and frequency response of the FIR low-pass filter.

III. Complex Basebanding and Desampling

The complex multiplication $e^{-j\omega n}x(n)$ is implemented where $\omega = 2\pi(f_0/f_s)$ and the center frequency f_0 = 250 Hz. The Fourier Transform of the first 256 data points in new complex sequence is calculated using a Kaiser Bessel window (α=2.5 or β=7.85) with NFFT = 256. The logarithmic magnitude of the Fourier Transform is shown in Figure 4 (left). The frequency components for the signal after complex basebanding are presented in Table 3. It is observed that the frequency component after complex basebanding f_l' can be calculated from the frequency component of the original signal f_l and the center frequency f_0 based on the following equation (detailed calculations are presented in Appendix 1):

$$f_l' = f_l - f_0$$

The mathematical derivation of this frequency shift is presented as follows.

Signal after complex basebanding:

$$\tilde{x}(n) = x(n) \cdot e^{-j2\pi(f_0/f_s)n}$$

Fourier Transform of the original signal with frequency component f_l:

$$X(f_l) = \sum_{n=-\infty}^{\infty} x(n) \cdot e^{-j2\pi(f_l/f_s)n}$$

Fourier Transform of the signal after complex basebanding:

$$\tilde{X}(f) = \sum_{n=-\infty}^{\infty} \tilde{x}(n) \cdot e^{-j2\pi(f_l/f_s)n} = \sum_{n=-\infty}^{\infty} x(n) \cdot e^{-j2\pi(f_0/f_s)n} \cdot e^{-j2\pi(f_l/f_s)n} = \sum_{n=-\infty}^{\infty} x(n) \cdot e^{-j2\pi(f_0+f_l/f_s)n}$$

$$\tilde{X}(f) = X(f_l + f_0)$$

Therefore, the spectrum of the signal translates to the left with the magnitude of f_0 after complex basebanding.

Table 3. Frequency components of the signal after complex basebanding.

f_l **[Hz]**	f_l' **[Hz]**	$\|X(f_l')\|$ **[dB]**	f_l **[Hz]**	f_l' **[Hz]**	$\|X(f_l')\|$ **[dB]**
160	-90	0	-160	-410	0
237	-13	-20	-237	-487	-20
240	-10	-20	-240	-490	-20

The sequence after complex basebanding is filtered by the FIR low-pass filter designed in Section II. The filtered signal is obtained as $y(n)$. The Fourier Transform of the windowed filtered signal $y(n)$ (n = 256, 257…511) is calculated using a Kaiser Bessel window (α=2.5 or β=7.85) with NFFT = 256. The logarithmic magnitude of the Fourier Transform is shown in Figure 4 (right). The frequency components of the signal $y(n)$ are presented in Table 4. The frequency values are the same compared to the signal before filtering because filtering does not shift the frequency of the signal. However, the magnitude of the peak values outside the passband frequency range (-40 Hz $\leq f \leq$ 40 Hz) are attenuated compared to the signal before filtering. Detailed calculations for the frequency components of the signal $y(n)$ are listed in Appendix 1.

Table 4. Frequency components of the signal after complex basebanding and low-pass filtering.

f_l **[Hz]**	f_l' **[Hz]**	$\|Y(f_l')\|$ **[dB]**	f_l **[Hz]**	f_l' **[Hz]**	$\|Y(f_l')\|$ **[dB]**
160	-90	-20	-160	-410	-20
237	-13	0	-237	-487	-40
240	-10	0	-240	-490	-40

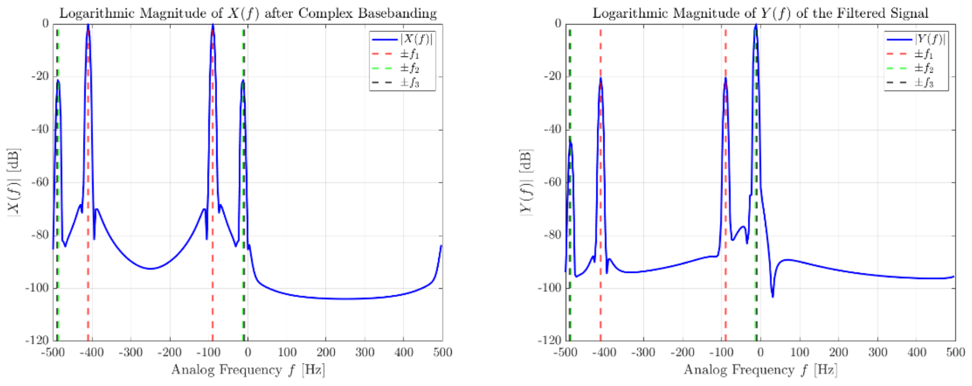

Figure 4. Logarithmic magnitude of the Fourier Transform for the complex baseband signal before (left) and after (right) passing the FIR low-pass filter.

IV. High Resolution Spectral Analysis

The complex filtered sequence is desampled by a factor of 8. The desampled sequence $x'(n)$ has a new sampling frequency $f_s' = 125$ Hz. The Fourier Transform of the windowed filtered signal $x'(n)$ ($n = 256, 257\ldots511$) is calculated using a Kaiser Bessel window (α=2.5 or β=7.85) with NFFT = 256. The logarithmic magnitude of the Fourier Transform is shown in Figure 5. The analog frequency bin width of the FFT is 0.488 Hz, which is 1/8 of the frequency bin width for the FFT of the original signal. The frequency values corresponding to the peaks of $|X'(f)|$ are presented in Table 5. Detailed calculations are presented in Appendix 1. Since the absolute values of some frequency components are greater than the Nyquist frequency ($f_s'/2 = 62.5$ Hz), aliasing is observed in the Fourier transform magnitude plot.

Table 5. Frequency components of the signal after complex basebanding, low-pass filtering and desampling.

f_i [Hz]	f_i' [Hz]	$\|X'(f_i')\|$ [dB]	f_i [Hz]	f_i' [Hz]	$\|X'(f_i')\|$ [dB]
160	35	-20	-160	-35	-20
237	-13	0	-237	13	-40
240	-10	-20	-240	10	-60

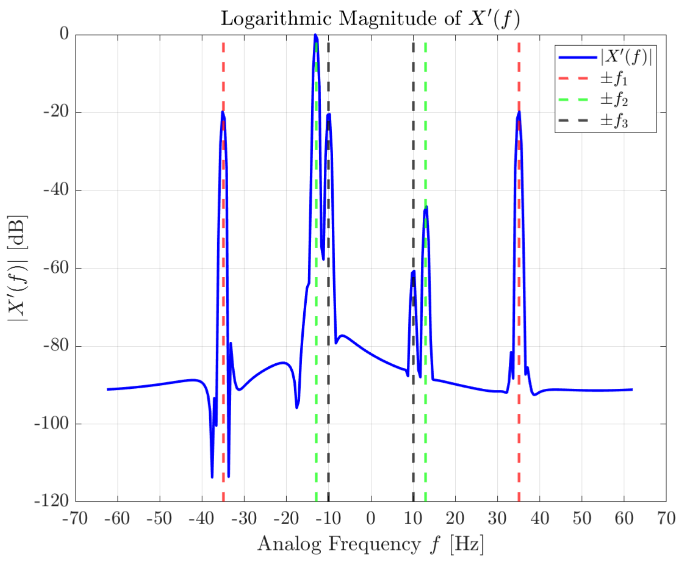

Figure 5. Logarithmic magnitude of the desampled signal $x'(n)$.

A second iteration in the design of the FIR low-pass filter is conducted. All requirements are to remain as originally given except for the passband/stopband weight ratio. Figure 6 and Figure 7 present the analysis result for the filter with passband/stopband weight ratio of 10 and 100 (the original weight ratio is 50). It is observed that a lower weight ratio has larger passband ripples and lower Fourier Transform magnitude in the stopband (larger stopband attenuation). In contrast, a higher weight ratio has smaller passband ripples and larger Fourier Transform magnitude in the stopband (smaller stopband attenuation). In practice, the selection of the passband/stopband weight ratio depends on the required behavior of the filter. A lower weight ratio is suggested if we want to remove the signal within the stopband (e.g. noise within the stopband has a significant effect on the results) regardless of the potential error introduced to the signal within the passband because of the large passband ripples. A higher weight ratio is suggested if we want to keep an accurate and smooth signal within the passband regardless of the potential effect from the signal within the stopband because of the smaller stopband attenuation.

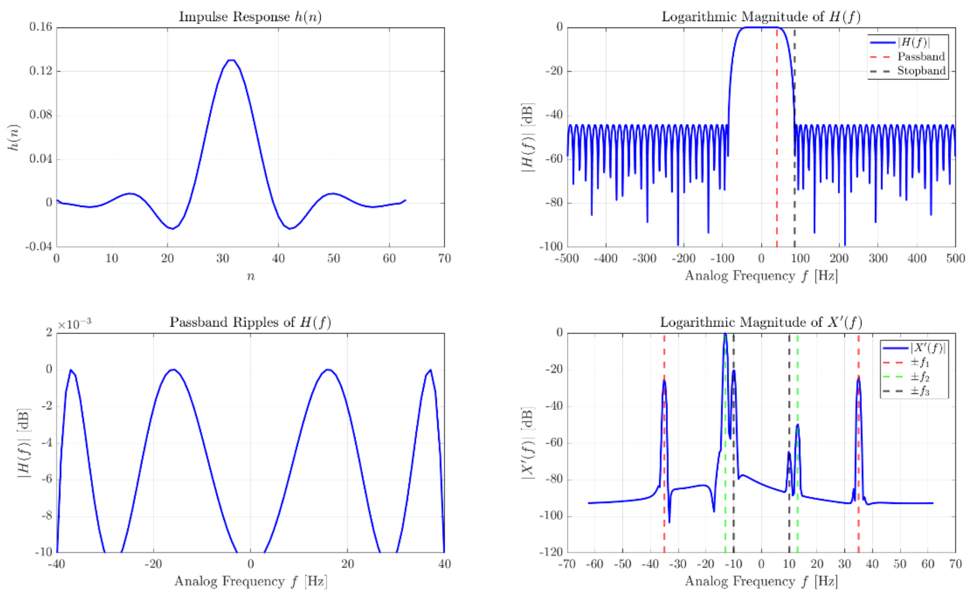

Figure 6. Analysis result for the case of passband/stopband weight ratio = 10.

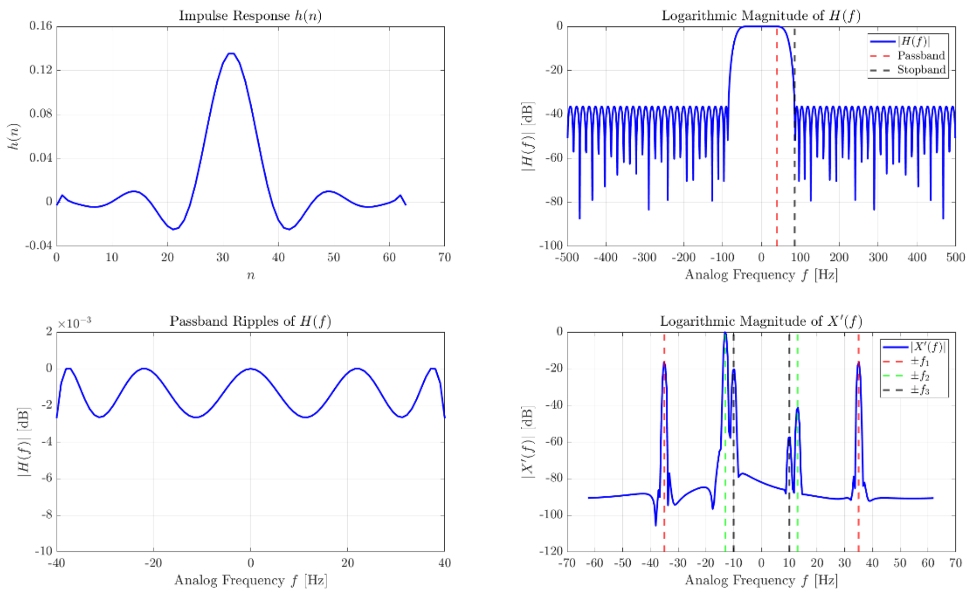

Figure 7. Analysis result for the case of passband/stopband weight ratio = 100.

Appendix 1. Hand Calculation

I. Data Set

$f_1 := 160\ Hz$ $f_2 := 237\ Hz$ $f_3 := 240\ Hz$ Analog frequency components of the signal

$A_1 := 100$ $A_2 := 10$ $A_3 := 1$ Linear amplitude for each analog frequency component

$f_s := 1000\ Hz$ Sampling frequency

$NFFT := 256$

$BinWidth := \frac{f_s}{NFFT} = 3.906\ Hz$ Analog frequency bin width of the FFT

$Bin\_f_1 := \text{round}\left(\frac{f_1}{BinWidth}\right) = 41$ Frequency bin index of analog frequency component f_1

$Bin\_f_2 := \text{round}\left(\frac{f_2}{BinWidth}\right) = 61$ Frequency bin index of analog frequency component f_2

$Bin\_f_3 := \text{round}\left(\frac{f_3}{BinWidth}\right) = 61$ Frequency bin index of analog frequency component f_3

Given the sampling frequency of 1000 Hz and NFFT = 256, the frequency bin indices for f_2 and f_3 are always 61, which is not affected by the window applied to the signal. Therefore, rectangular windowed FFT will not indicate distinct spectral peaks for f_2 and f_3.

$20 \cdot \log\left(A_1\right) = 40$ Magnitude of the peak corresponding to +/-f_1

$20 \cdot \log\left(A_2\right) + 20 \cdot \log\left(A_3\right) = 20$ Magnitude of the peak corresponding to +/-f_2 and +/-f_3 (same frequency bin)

Therefore, after the normalization, the magnitude of the peak corresponding to +/-f_1 is 0 dB and the magnitude of the peak corresponding to +/-f_2 and +/-f_3 (same frequency bin) is -20 dB.

III. Complex Basebanding and Desampling

1. Complex basebanding:

$f_0 := 250\ Hz$ Center frequency

Frequency components after complex basebanding:

$f_1' := f_1 - f_0 = -90\ Hz$ $f_2' := f_2 - f_0 = -13\ Hz$ $f_3' := f_3 - f_0 = -10\ Hz$

$f_1'' := -f_1 - f_0 = -410\ Hz$ $f_2'' := -f_2 - f_0 = -487\ Hz$ $f_3'' := -f_3 - f_0 = -490\ Hz$

Complex basebanding does not change the magnitude of the signal. Therefore, the magnitude of peaks corresponding to each frequency component remain the same as previous.

2. Low-pass filtering:
Low-pass filtering does not shift the frequency of the signal. Therefore, frequency values of the signal remain the same as previous.

$20 \cdot \log\left(A_1\right) - 40 = 0$ Magnitude of the peak corresponding to f_1

$20 \cdot \log\left(A_2\right) + 20 \cdot \log\left(A_3\right) = 20$ Magnitude of the peak corresponding to f_2 and f_3 (same frequency bin)

$20 \cdot \log\left(A_1\right) - 40 = 0$ Magnitude of the peak corresponding to $-f_1$

$20 \cdot \log\left(A_2\right) + 20 \cdot \log\left(A_3\right) - 40 = -20$ Magnitude of the peak corresponding to $-f_2$ and $-f_3$ (same frequency bin)

Therefore, after the normalization, the magnitude of the peak corresponding to f_1 is -20 dB; the magnitude of the peak corresponding to f_2 and f_3 (same frequency bin) is 0 dB; the magnitude of the peak corresponding to $-f_1$ is -20 dB; the magnitude of the peak corresponding to $-f_2$ and $-f_3$ (same frequency bin) is -40 dB;

IV. High Resolution Spectral Analysis

$f_s' := \frac{f_s}{8} = 125\ Hz$ Sampling frequency after decimation

$BinWidth := \frac{f_s'}{NFFT} = 0.488\ Hz$ Analog frequency bin width of the FFT after desampling

Frequency components after desampling:

$f_1' := f_1 - f_0 + f_s' = 35\ Hz$ $f_2' := f_2 - f_0 = -13\ Hz$ $f_3' := f_3 - f_0 = -10\ Hz$

$f_1'' := -f_1 - f_0 + 3\ f_s' = -35\ Hz$ $f_2'' := -f_2 - f_0 + 4\ f_s' = 13\ Hz$ $f_3'' := -f_3 - f_0 + 4\ f_s' = 10\ Hz$

$20 \cdot \log\left(A_1\right) - 40 = 0$ Magnitude of the peak corresponding to f_1 (-20 dB after normalization)

$20 \cdot \log\left(A_2\right) = 20$ Magnitude of the peak corresponding to f_2 (0 dB after normalization)

$20 \cdot \log\left(A_3\right) = 0$ Magnitude of the peak corresponding to f_3 (-20 dB after normalization)

$20 \cdot \log\left(A_1\right) - 40 = 0$ Magnitude of the peak corresponding to $-f_1$ (-20 dB after normalization)

$20 \cdot \log\left(A_2\right) - 40 = -20$ Magnitude of the peak corresponding to $-f_2$ (-40 dB after normalization)

$20 \cdot \log\left(A_3\right) - 40 = -40$ Magnitude of the peak corresponding to $-f_3$ (-60 dB after normalization)

Appendix 2. MATLAB Code

```matlab
% SIO 207A Final Project
% Ruipu Ji

% Initialization and default plot settings.
clear; clc; close all;

set(0, 'DefaultAxesFontSize', 15);
set(0, 'DefaultTextFontSize', 15);

set(0, 'DefaultTextInterpreter', 'latex');
set(0, 'DefaultLegendInterpreter', 'latex');
set(0, 'DefaultAxesTickLabelInterpreter', 'latex');

%% Data Set.
% Generate a 4096-point discrete-time signal x(n). ------------------------
A = [100 10 1]; % Magnitude.
f = [160 237 240]; % Analog frequency [Hz].
Phi = [0 0 0]; % Phase angle [rad].

fs = 1000; % Sampling frequency [Hz].
n = 4096; % Number of data points.
t = (0:1:n-1)'*1/fs; % Time vector.

x = zeros(n,1); % Initialize the signal sequence as an empty array.

% Generate the discrete-time signal sequence x(n).
for idx = 1:size(A,2)
    x = x + A(idx)*cos(2*pi*f(idx)*t+Phi(idx));
end

% Plot the first 256 data points of the signal x(n). ----------------------
NFFT = 256;

figure('Position', [0, 0, 1800, 600]);

subplot(1,2,1);
hold on;
plot(0:1:NFFT-1, x(1:NFFT), 'b', 'LineWidth', 2);
grid on;
box on;
xlim([0 300]);
xticks(0:50:300);
ylim([-150 150]);
yticks(-150:50:150);
xlabel('Data Point Index $n$');
ylabel('$x(n)$');
title('The First 256 Data Points of Signal $x(n)$');

% 256-point FFT of the signal using the Kaiser-Bessel window. -------------
% Kaiser-Bessel window (alpha = 2.5 or beta = alpha*pi = 7.85).
KaiserBesselWindow = kaiser(NFFT, 7.85);

% Window the first 256 data points of the signal x(n).
x_256_KB = x(1:NFFT) .* KaiserBesselWindow;

% Perform 256-point FFT on the windowed signal and calculate the logarithmic↙
```

```
magnitude.
X_256_KB_magnitude = 20*log10(abs(fftshift(fft(x_256_KB, NFFT))));
X_256_KB_magnitude = X_256_KB_magnitude - max(X_256_KB_magnitude); % Normalize the results.

% Calculate bin width of the FFT.
BinWidth = fs/NFFT;

% Calculate the frequency bin index for analog frequencies.
f_idx = zeros(1, size(f,2));

for idx = 1:size(f_idx,2)
    f_idx(1,idx) = round(f(idx)/BinWidth);
end

% Plot the result.
f_fft = (-fs/2:BinWidth:fs/2-BinWidth)';

subplot(1,2,2);
hold on;
plot(f_fft, X_256_KB_magnitude, 'b', 'LineWidth', 2);
xline(f(1), 'r--', 'LineWidth', 2);
xline(f(2), 'g--', 'LineWidth', 2);
xline(f(3), 'k--', 'LineWidth', 2);
xline(-f(1), 'r--', 'LineWidth', 2);
xline(-f(2), 'g--', 'LineWidth', 2);
xline(-f(3), 'k--', 'LineWidth', 2);
grid on;
box on;
xlim([-500 500]);
xticks(-500:100:500);
ylim([-120 0]);
yticks(-120:20:0);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|X(f)|$ [dB]');
legend('$|X(f)|$', '$\pm f_1$', '$\pm f_2$', '$\pm f_3$', 'Location', 'northeast');
title('Logrithmic Magnitude of $X(f)$');

exportgraphics(gcf, 'Figure1-PartI.png', 'ContentType', 'image');

%% Decimation Filter Design.
N_coefficients = 64; % Number of coefficients.
fc_passband = 40; % Passband cutoff frequency [Hz].
fc_stopband = 85; % Stopband cutoff frequency [Hz].
Weight = [50 1]; % Define weight ratio vector [passband stopband].

% Filter design using an equiripple FIR filter design algorithm.
f_parameter = [0 fc_passband/(fs/2) fc_stopband/(fs/2) 1];
a_parameter = [1 1 0 0];
h = firpm(N_coefficients-1, f_parameter, a_parameter, Weight)'; % Filter design.

% Calculate the frequency response of the filter.
NFFT_filter = 1024; % NFFT for the filter calculation.
f_filter_fft = (-0.5:1/NFFT_filter:0.5-1/NFFT_filter)' * fs; % Frequency vector for the plot.
RectangularWindow = rectwin(NFFT_filter); % Rectangular window.
```

```
h_padded = padarray(h, [NFFT_filter-size(h,1) 0], 'post'); % Pad the filter to the↙
same length as NFFT.

% Perform 256-point FFT on the windowed signal and calculate the logarithmic↙
magnitude.
H_magnitude = 20*log10(abs(fftshift(fft(h_padded.*RectangularWindow, NFFT_filter))));
H_magnitude = H_magnitude - max(H_magnitude); % Normalize the result.

% Plot the result.
figure('Position', [0, 0, 2500, 500]);

subplot(1,3,1);
hold on;
plot(0:1:N_coefficients-1, h, 'b', 'LineWidth', 2);
grid on;
box on;
xlim([0 70]);
xticks(0:10:70);
ylim([-0.04 0.16]);
yticks(-0.04:0.04:0.16);
xlabel('$n$');
ylabel('$h(n)$');
title('Impulse Response $h(n)$');

subplot(1,3,2);
hold on;
plot(f_filter_fft, H_magnitude, 'b', 'LineWidth', 2);
xline(fc_passband, 'r--', 'LineWidth', 2);
xline(fc_stopband, 'k--', 'LineWidth', 2);
grid on;
box on;
xlim([-500 500]);
xticks(-500:100:500);
ylim([-100 0]);
yticks(-100:20:0);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|H(f)|$ [dB]');
legend('$|H(f)|$', 'Passband', 'Stopband', 'Location', 'northeast');
title('Logrithmic Magnitude of $H(f)$');

subplot(1,3,3);
hold on;
plot(f_filter_fft, H_magnitude, 'b', 'LineWidth', 2);
grid on;
box on;
xlim([0 40]);
xticks(0:10:40);
ylim([-0.01 0.002]);
yticks(-0.01:0.002:0.002);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|H(f)|$ [dB]');
title('Passband Ripples of $H(f)$');

exportgraphics(gcf, 'Figure2-PartII.png', 'ContentType', 'image');

%% Complex Basebanding and Desampling.
```

```matlab
% Complex multiplication on the discrete-time signal x(n). ----------------
f0 = 250; % Center frequency [Hz].
w0 = 2*pi*f0/fs; % Center frequency in rad.
x_complex = zeros(size(x,1), 1); % Initialization for the complex sequence.

for idx = 1:size(x,1)
    x_complex(idx) = exp(-1i*w0*idx)*x(idx);
end

% 256-point FFT of the complex sequence. ----------------------------------
% Window the first 256 data points of the complex sequence.
x_complex_256_KB = x_complex(1:NFFT) .* KaiserBesselWindow;

% Perform 256-point FFT on the windowed signal and calculate the logarithmic magnitude.
X_complex_256_KB_magnitude = 20*log10(abs(fftshift(fft(x_complex_256_KB, NFFT))));
X_complex_256_KB_magnitude = X_complex_256_KB_magnitude - max(X_complex_256_KB_magnitude); % Normalize the results.

% Plot the results.
figure('Position', [0, 0, 1800, 600]);

subplot(1,2,1);
hold on;
plot(f_fft, X_complex_256_KB_magnitude, 'b', 'LineWidth', 2);
xline(f(1)-f0, 'r--', 'LineWidth', 2);
xline(f(2)-f0, 'g--', 'LineWidth', 2);
xline(f(3)-f0, 'k--', 'LineWidth', 2);
xline(-f(1)-f0, 'r--', 'LineWidth', 2);
xline(-f(2)-f0, 'g--', 'LineWidth', 2);
xline(-f(3)-f0, 'k--', 'LineWidth', 2);
grid on;
box on;
xlim([-500 500]);
xticks(-500:100:500);
ylim([-120 0]);
yticks(-120:20:0);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|X(f)|$ [dB]');
legend('$|X(f)|$', '$\pm f_1$', '$\pm f_2$', '$\pm f_3$', 'Location', 'northeast');
title('Logrithmic Magnitude of $X(f)$ after Complex Basebanding');

% Low pass filter the complex sequence in the time domain. ----------------
y_complex = filter(h, 1, x_complex);

% Window the data points (n = 256-511) of the signal y(n).
y_complex_256_KB = y_complex(NFFT+1:2*NFFT) .* KaiserBesselWindow;

% Perform 256-point FFT on the windowed signal and calculate the logarithmic magnitude.
Y_complex_256_KB_magnitude = 20*log10(abs(fftshift(fft(y_complex_256_KB, NFFT))));
Y_complex_256_KB_magnitude = Y_complex_256_KB_magnitude - max(Y_complex_256_KB_magnitude); % Normalize the results.

% Plot the results.
subplot(1,2,2);
```

```matlab
hold on;
plot(f_fft, Y_complex_256_KB_magnitude, 'b', 'LineWidth', 2);
xline(f(1)-f0, 'r--', 'LineWidth', 2);
xline(f(2)-f0, 'g--', 'LineWidth', 2);
xline(f(3)-f0, 'k--', 'LineWidth', 2);
xline(-f(1)-f0, 'r--', 'LineWidth', 2);
xline(-f(2)-f0, 'g--', 'LineWidth', 2);
xline(-f(3)-f0, 'k--', 'LineWidth', 2);
grid on;
box on;
xlim([-500 500]);
xticks(-500:100:500);
ylim([-120 0]);
yticks(-120:20:0);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|Y(f)|$ [dB]');
legend('$|Y(f)|$', '$\pm f_1$', '$\pm f_2$', '$\pm f_3$', 'Location', 'northeast');
title('Logrithmic Magnitude of $Y(f)$ of the Filtered Signal');

exportgraphics(gcf, 'Figure3-PartIII.png', 'ContentType', 'image');

% Desample the complex filtered sequence y(n) by a factor of 8. -----------
factor_downsample = 8;
x_prime = y_complex(1:factor_downsample:end);

%% High Resolution Spectral Analysis.
% Window the data points (n = 256-511) of the signal x'(n).
x_prime_256_KB = x_prime(NFFT+1:2*NFFT) .* KaiserBesselWindow;

% Perform 256-point FFT on the windowed signal and calculate the logarithmic magnitude.
X_prime_256_KB_magnitude = 20*log10(abs(fftshift(fft(x_prime_256_KB, NFFT))));
X_prime_256_KB_magnitude = X_prime_256_KB_magnitude - max(X_prime_256_KB_magnitude); % Normalize the results.

% Calculate the upadated bin width of the FFT.
fs_prime = fs/factor_downsample;
BinWidth_prime = fs_prime/NFFT;
f_prime_fft = (-fs_prime/2:BinWidth_prime:fs_prime/2-BinWidth_prime)';

% Plot the results.
figure('Position', [0, 0, 800, 600]);
hold on;
plot(f_prime_fft, X_prime_256_KB_magnitude, 'b', 'LineWidth', 2);
xline(f(1)-f0+fs_prime, 'r--', 'LineWidth', 2);
xline(f(2)-f0, 'g--', 'LineWidth', 2);
xline(f(3)-f0, 'k--', 'LineWidth', 2);
xline(-f(1)-f0+3*fs_prime, 'r--', 'LineWidth', 2);
xline(-f(2)-f0+4*fs_prime, 'g--', 'LineWidth', 2);
xline(-f(3)-f0+4*fs_prime, 'k--', 'LineWidth', 2);
grid on;
box on;
xlim([-70 70]);
xticks(-70:10:70);
ylim([-120 0]);
yticks(-120:20:0);
```

```matlab
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|X^\prime(f)|$ [dB]');
legend('$|X^\prime(f)|$', '$\pm f_1$', '$\pm f_2$', '$\pm f_3$', 'Location', 'northeast');
title('Logrithmic Magnitude of $X^\prime(f)$');

exportgraphics(gcf, 'Figure4-PartIV.png', 'ContentType', 'image');

%% Second Iteration on LPF Design.
% Trial #1: Use passband/stopband weight ratio = 10. ----------------------
Weight2 = [10 1]; % Define weight ratio vector [passband stopband].
h2 = firpm(N_coefficients-1, f_parameter, a_parameter, Weight2)'; % Filter design.

% Calculate the frequency response of the filter.
h2_padded = padarray(h2, [NFFT_filter-size(h2,1) 0], 'post'); % Pad the filter to the same length as NFFT.

% Perform 256-point FFT on the windowed signal and calculate the logarithmic magnitude.
H2_magnitude = 20*log10(abs(fftshift(fft(h2_padded.*RectangularWindow, NFFT_filter))));
H2_magnitude = H2_magnitude - max(H2_magnitude); % Normalize the result.

% Plot the result.
figure('Position', [0, 0, 1800, 1000]);

subplot(2,2,1);
hold on;
plot(0:1:N_coefficients-1, h2, 'b', 'LineWidth', 2);
grid on;
box on;
xlim([0 70]);
xticks(0:10:70);
ylim([-0.04 0.16]);
yticks(-0.04:0.04:0.16);
xlabel('$n$');
ylabel('$h(n)$');
title('Impulse Response $h(n)$');

subplot(2,2,2);
hold on;
plot(f_filter_fft, H2_magnitude, 'b', 'LineWidth', 2);
xline(fc_passband, 'r--', 'LineWidth', 2);
xline(fc_stopband, 'k--', 'LineWidth', 2);
grid on;
box on;
xlim([-500 500]);
xticks(-500:100:500);
ylim([-100 0]);
yticks(-100:20:0);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|H(f)|$ [dB]');
legend('$|H(f)|$', 'Passband', 'Stopband', 'Location', 'northeast');
title('Logrithmic Magnitude of $H(f)$');

subplot(2,2,3);
```

```
hold on;
plot(f_filter_fft, H2_magnitude, 'b', 'LineWidth', 2);
grid on;
box on;
xlim([0 40]);
xticks(0:10:40);
ylim([-0.01 0.002]);
yticks(-0.01:0.002:0.002);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|H(f)|$ [dB]');
title('Passband Ripples of $H(f)$');

% Low pass filter the complex sequence in the time domain.
y2_complex = filter(h2, 1, x_complex);

% Desample the complex filtered sequence y(n) by a factor of 8.
x2_prime = y2_complex(1:factor_downsample:end);

% Window the data points (n = 256-511) of the signal x'(n).
x2_prime_256_KB = x2_prime(NFFT+1:2*NFFT) .* KaiserBesselWindow;

% Perform 256-point FFT on the windowed signal and calculate the logarithmic↙
magnitude.
X2_prime_256_KB_magnitude = 20*log10(abs(fftshift(fft(x2_prime_256_KB, NFFT))));
X2_prime_256_KB_magnitude = X2_prime_256_KB_magnitude - max↙
(X2_prime_256_KB_magnitude); % Normalize the results.

% Plot the results.
subplot(2,2,4);
hold on;
plot(f_prime_fft, X2_prime_256_KB_magnitude, 'b', 'LineWidth', 2);
xline(f(1)-f0+fs_prime, 'r--', 'LineWidth', 2);
xline(f(2)-f0, 'g--', 'LineWidth', 2);
xline(f(3)-f0, 'k--', 'LineWidth', 2);
xline(-f(1)-f0+3*fs_prime, 'r--', 'LineWidth', 2);
xline(-f(2)-f0+4*fs_prime, 'g--', 'LineWidth', 2);
xline(-f(3)-f0+4*fs_prime, 'k--', 'LineWidth', 2);
grid on;
box on;
xlim([-70 70]);
xticks(-70:10:70);
ylim([-120 0]);
yticks(-120:20:0);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|X^\prime(f)|$ [dB]');
legend('$|X^\prime(f)|$', '$\pm f_1$', '$\pm f_2$', '$\pm f_3$', 'Location',↙
'northeast');
title('Logrithmic Magnitude of $X^\prime(f)$');

exportgraphics(gcf, 'Figure5-PartIV-Trial#1.png', 'ContentType', 'image');

% Trial #2: Use passband/stopband weight ratio = 100. ----------------------
Weight3 = [100 1]; % Define weight ratio vector [passband stopband].
h3 = firpm(N_coefficients-1, f_parameter, a_parameter, Weight3)'; % Filter design.

% Calculate the frequency response of the filter.
```

```
h3_padded = padarray(h3, [NFFT_filter-size(h3,1) 0], 'post'); % Pad the filter to the same length as NFFT.

% Perform 256-point FFT on the windowed signal and calculate the logarithmic magnitude.
H3_magnitude = 20*log10(abs(fftshift(fft(h3_padded.*RectangularWindow, NFFT_filter))));
H3_magnitude = H3_magnitude - max(H3_magnitude); % Normalize the result.

% Plot the result.
figure('Position', [0, 0, 1800, 1000]);

subplot(2,2,1);
hold on;
plot(0:1:N_coefficients-1, h3, 'b', 'LineWidth', 2);
grid on;
box on;
xlim([0 70]);
xticks(0:10:70);
ylim([-0.04 0.16]);
yticks(-0.04:0.04:0.16);
xlabel('$n$');
ylabel('$h(n)$');
title('Impulse Response $h(n)$');

subplot(2,2,2);
hold on;
plot(f_filter_fft, H3_magnitude, 'b', 'LineWidth', 2);
xline(fc_passband, 'r--', 'LineWidth', 2);
xline(fc_stopband, 'k--', 'LineWidth', 2);
grid on;
box on;
xlim([-500 500]);
xticks(-500:100:500);
ylim([-100 0]);
yticks(-100:20:0);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|H(f)|$ [dB]');
legend('$|H(f)|$', 'Passband', 'Stopband', 'Location', 'northeast');
title('Logrithmic Magnitude of $H(f)$');

subplot(2,2,3);
hold on;
plot(f_filter_fft, H3_magnitude, 'b', 'LineWidth', 2);
grid on;
box on;
xlim([0 40]);
xticks(0:10:40);
ylim([-0.01 0.002]);
yticks(-0.01:0.002:0.002);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|H(f)|$ [dB]');
title('Passband Ripples of $H(f)$');

% Low pass filter the complex sequence in the time domain.
y3_complex = filter(h3, 1, x_complex);
```

```matlab

% Desample the complex filtered sequence y(n) by a factor of 8.
x3_prime = y3_complex(1:factor_downsample:end);

% Window the data points (n = 256-511) of the signal x'(n).
x3_prime_256_KB = x3_prime(NFFT+1:2*NFFT) .* KaiserBesselWindow;

% Perform 256-point FFT on the windowed signal and calculate the logarithmic magnitude.
X3_prime_256_KB_magnitude = 20*log10(abs(fftshift(fft(x3_prime_256_KB, NFFT))));
X3_prime_256_KB_magnitude = X3_prime_256_KB_magnitude - max(X3_prime_256_KB_magnitude); % Normalize the results.

% Plot the results.
subplot(2,2,4);
hold on;
plot(f_prime_fft, X3_prime_256_KB_magnitude, 'b', 'LineWidth', 2);
xline(f(1)-f0+fs_prime, 'r--', 'LineWidth', 2);
xline(f(2)-f0, 'g--', 'LineWidth', 2);
xline(f(3)-f0, 'k--', 'LineWidth', 2);
xline(-f(1)-f0+3*fs_prime, 'r--', 'LineWidth', 2);
xline(-f(2)-f0+4*fs_prime, 'g--', 'LineWidth', 2);
xline(-f(3)-f0+4*fs_prime, 'k--', 'LineWidth', 2);
grid on;
box on;
xlim([-70 70]);
xticks(-70:10:70);
ylim([-120 0]);
yticks(-120:20:0);
xlabel('Analog Frequency $f$ [Hz]');
ylabel('$|X^\prime(f)|$ [dB]');
legend('$|X^\prime(f)|$', '$\pm f_1$', '$\pm f_2$', '$\pm f_3$', 'Location', 'northeast');
title('Logrithmic Magnitude of $X^\prime(f)$');

exportgraphics(gcf, 'Figure6-PartIV-Trial#2.png', 'ContentType', 'image');

```