Appendix 2. MATLAB Code

```
% 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.
```

```
% 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);
```

```
% 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');
```