Faranak Dayyani - student number: 1002373674

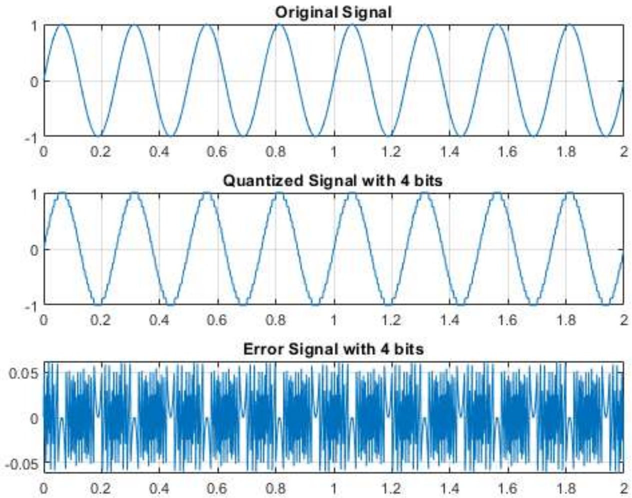
Contents

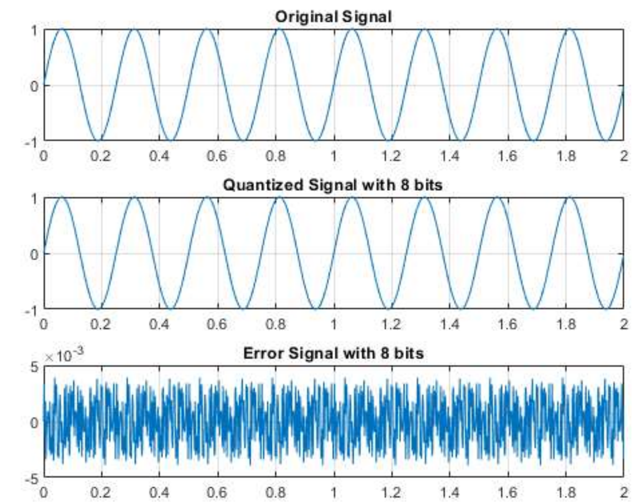
Question 1:

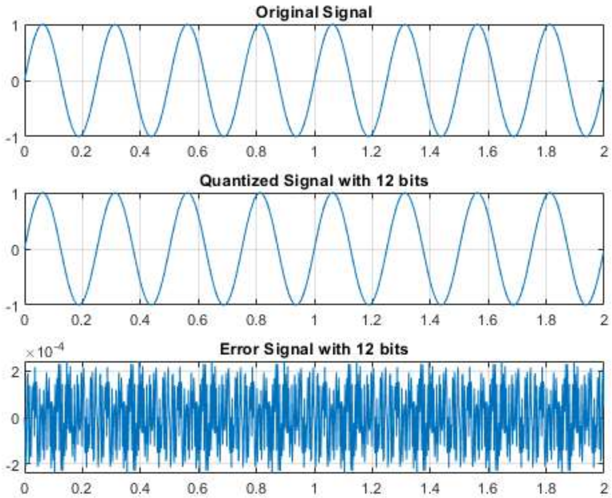
initializing variables

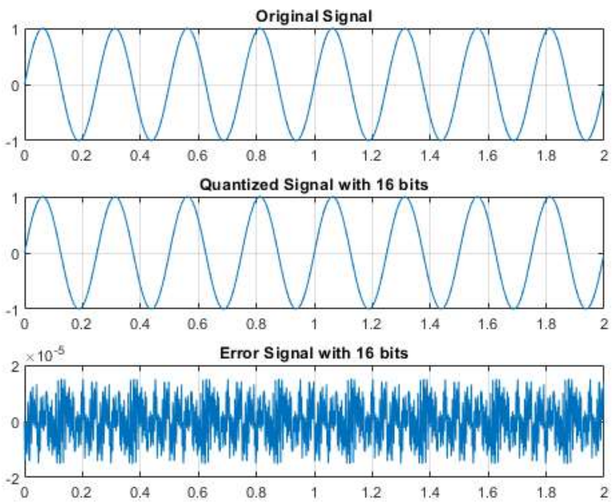
```
Ts = 0.002;
N = 1000;
f = 4;
b = 0; %bits
t = (0:N-1)*Ts;

original_signal = sin(2*pi*f*t);

for b = 4:4:16
    % defining the quantized signal
    quantized_signal = quantization(original_signal, b);
    error_signal = original_signal - quantized_signal;

    figure;
    subplot(3,1,1);
    plot(t, original_signal);
    title('Original Signal');
    grid on
    subplot(3,1,2);
    plot(t, quantized_signal);
    title(sprintf('Quantized Signal with %d bits',b));

    grid on
    subplot(3,1,3);
    plot(t, error_signal);
    title(sprintf('Error Signal with %d bits',b));
    grid on

    max_err = max(error_signal);
    disp(sprintf('The amplitude for %d bits error signal is: ',b)); disp(max_err);

end
```

```
The amplitude for 4 bits error signal is:
    0.0621

The amplitude for 8 bits error signal is:
    0.0039

The amplitude for 12 bits error signal is:
   2.4326e-04

The amplitude for 16 bits error signal is:
   1.5146e-05
```


Original Signal
Quantized Signal with 4 bits
Error Signal with 4 bits

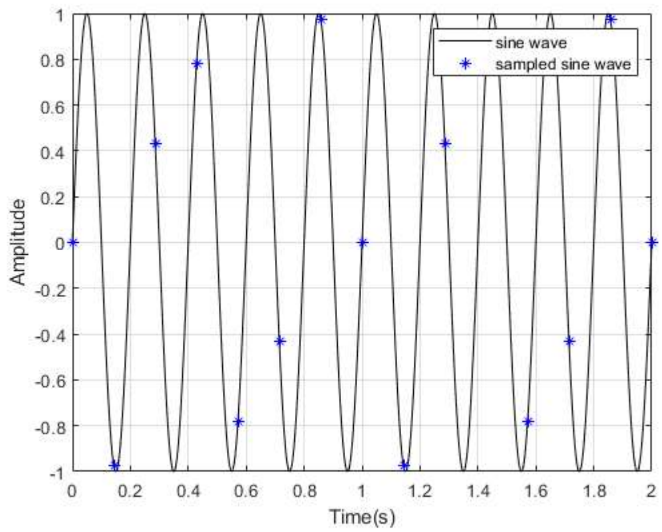

Original Signal
Quantized Signal with 8 bits
Error Signal with 8 bits

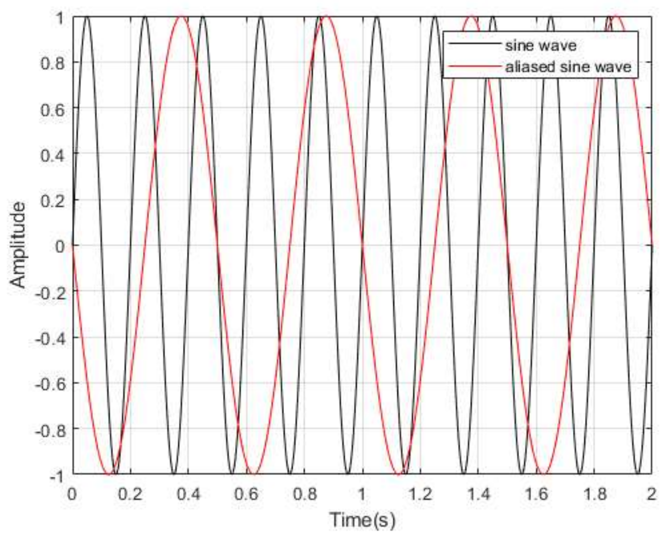




Question 2:

```
N = 1000;
f = 5;
T = 1/f;
fs = 7;
Ts = 1/fs;

t = linspace(0,2,1000);

signal = sin(2*pi*f*t);

t_sampled = 0:Ts:2;
signal_sampled = sin(2*pi*f*t_sampled);

signal_aliased = -1*sin(2*pi*2*t);

figure;
plot(t, signal,'k'); hold on; plot(t_sampled, signal_sampled,'b*'); hold off;
xlabel('Time(s)'); ylabel('Amplitude'); grid on; legend('sine wave','sampled sine wave');

figure;
plot(t, signal,'k'); hold on; plot(t, signal_aliased,'r'); hold off;
xlabel('Time(s)'); ylabel('Amplitude'); grid on; legend('sine wave','aliased sine wave');
```



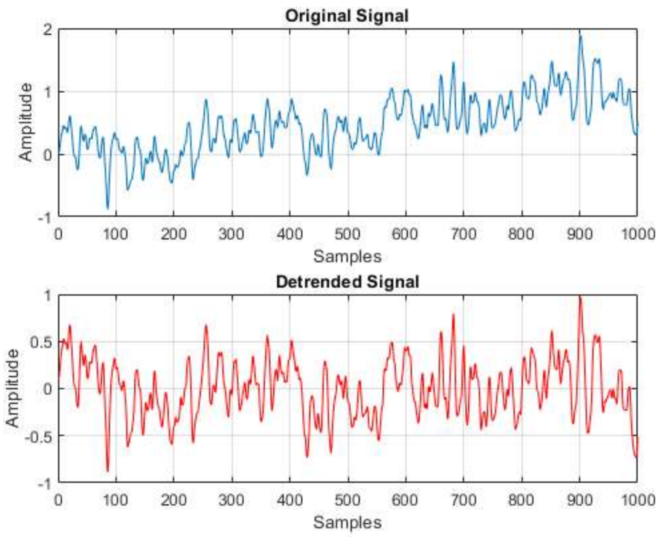


Question 3:

```
%Loading the signal data
s = load('data_c1.mat');
dataset = s.x;

% segmenting the data
segmented_data = cell(1,length(dataset)/100);

k = 1;
for l=1:100:length(dataset)

    segmented_data{k} = dataset(l:l+99);
    k = k+1;
end

% evaluating the mean in each segment
means = cell(1,length(dataset)/100);

for i=1:length(segmented_data)
    mean_segment = segmented_data(i);
    means{i} = mean(mean_segment{1,1});
end

% evaluating the variance in each segment
variance = cell(1,length(dataset)/100);

for j=1:length(segmented_data)
    variance_segment = segmented_data(j);
    variance{j} = var(variance_segment{1,1});
end
disp('Means for 10 segments:'); disp(means);
disp('Variance for 10 segments:'); disp(variance);
disp('The variance and mean are different for each segment, therefore, the signal is nonstationary.');
disp('-------------------------------------------------------------------------------------');
% The variance and mean are different for each segment, therefore, the signal is nonstationary.
```

```
% applying MATLAB's detrend operator
d_dataset = detrend(dataset);

% segmenting the detrended signal
d_segmented_data = cell(1,length(d_dataset)/100);

d_k = 1;
for d_l=1:100:length(d_dataset)

    d_segmented_data{d_k} = d_dataset(d_l:d_l+99);
    d_k = d_k+1;
end

% calculating the mean for the detrended signal
d_means = cell(1,length(d_dataset)/100);

for d_i=1:length(d_segmented_data)
    d_mean_segment = d_segmented_data(d_i);
    d_means{d_i} = mean(d_mean_segment{1,1});
end

% calculating the variance for the detrended signal
d_variance = cell(1,length(d_dataset)/100);

for d_j=1:length(d_segmented_data)
    d_variance_segment = d_segmented_data(d_j);
    d_variance{d_j} = var(d_variance_segment{1,1});
end

disp('Means for 10 segments of detrended signal:'); disp(d_means);
disp('Variance for 10 segments of detrended signal:'); disp(d_variance);
disp('the variance for the detrended signal compared to the original signal are very (extremely) close in values but the mean values are more different between the
% the variance for the detrended signal compared to the original signal are very (extremely) close in values
% but the mean values are more different between the original signal and the detrended signal.

% plotting the two signals: original and detrended
figure;
subplot(2,1,1);
plot(dataset);
xlabel('Samples');
ylabel('Amplitude');
title('Original Signal');
grid on;
subplot(2,1,2);
plot(d_dataset,'r');
xlabel('Samples');
ylabel('Amplitude');
title('Detrended Signal');
grid on;
```

```
Means for 10 segments:
  Columns 1 through 5

    {[0.1461]}    {[-0.1058]}    {[0.1875]}    {[0.3915]}    {[0.3092]}

  Columns 6 through 10

    {[0.4516]}    {[0.6808]}    {[0.6537]}    {[1.0329]}    {[0.9620]}

Variance for 10 segments:
  Columns 1 through 5

    {[0.0878]}    {[0.0518]}    {[0.0897]}    {[0.0448]}    {[0.0939]}

  Columns 6 through 10

    {[0.0995]}    {[0.0828]}    {[0.0424]}    {[0.0582]}    {[0.1378]}

The variance and mean are different for each segment, therefore, the signal is nonstationary.
------------------------------------------------------------------------------------
Means for 10 segments of detrended signal:
  Columns 1 through 5

    {[0.1807]}    {[-0.1835]}    {[-0.0026]}    {[0.0891]}    {[-0.1055]}

  Columns 6 through 10

    {[-0.0755]}    {[0.0413]}    {[-0.0981]}    {[0.1687]}    {[-0.0146]}

Variance for 10 segments of detrended signal:
  Columns 1 through 5

    {[0.0963]}    {[0.0561]}    {[0.0827]}    {[0.0406]}    {[0.0957]}

  Columns 6 through 10

    {[0.0858]}    {[0.0807]}    {[0.0440]}    {[0.0578]}    {[0.1518]}

the variance for the detrended signal compared to the original signal are very (extremely) close in values but the mean values are more different between the origin
```

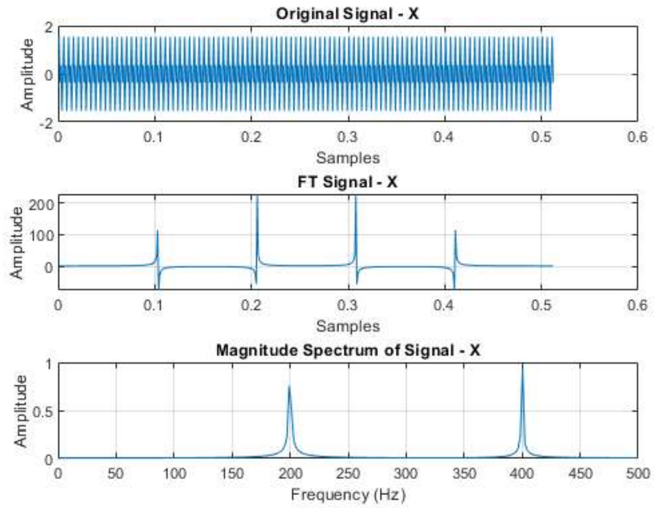


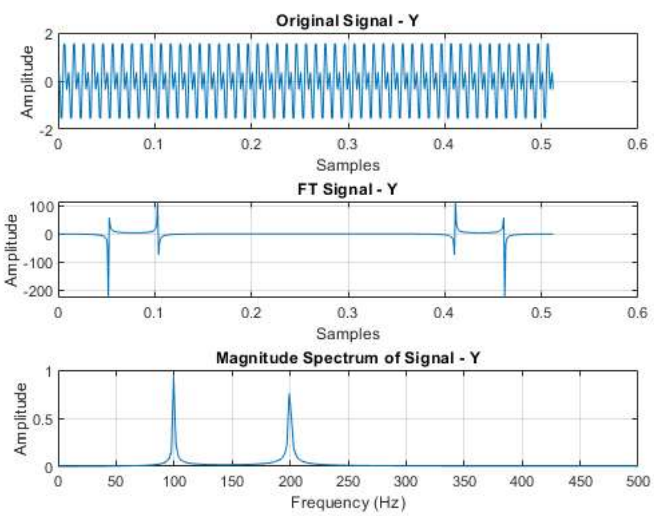
Question 4:

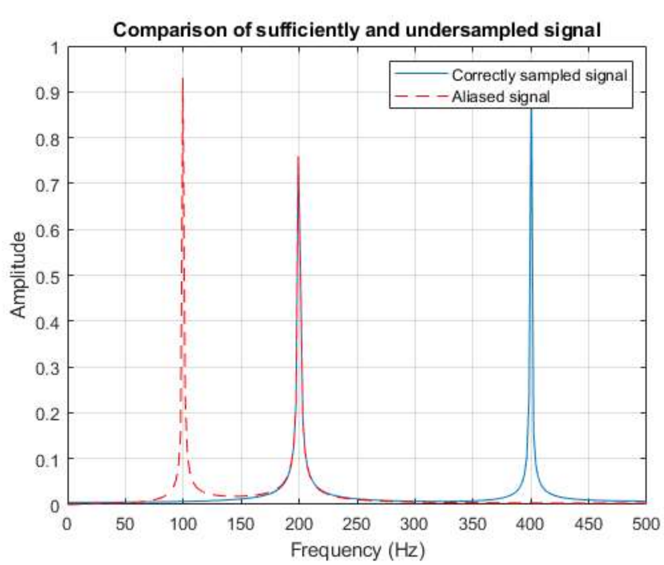
```
% data given:
N = 512;
fs = 1000;
T = 1/fs;
f1 = 200;
f2_1 = 400;
f2_2 = 900;

% frequency and time vector calculated:
t1 = (0:N-1)*T;
t = (1:N)/fs;

% defining the sinusoidal waves:
x = sin(2*pi*f1*t) + sin(2*pi*f2_1*t);
y = sin(2*pi*f1*t) + sin(2*pi*f2_2*t);

% applying FT to the sinusoidal waves:
x_ft = fft(x);
y_ft = fft(y);

% magnitude of the spectrum:
x_abs = abs(x_ft/N);
x_spec = x_abs(1:N/2+1);
x_spec(2:end-1) = 2*x_spec(2:end-1);

y_abs = abs(y_ft/N);
y_spec = y_abs(1:N/2+1);
y_spec(2:end-1) = 2*y_spec(2:end-1);

f_axis = fs*(0:(N/2))/N;

figure;
subplot(3,1,1);
plot(t, x); title('Original Signal - X'); xlabel('Samples'); ylabel('Amplitude'); grid on;

subplot(3,1,2);
plot(t, x_ft); title('FT Signal - X'); xlabel('Samples'); ylabel('Amplitude'); grid on;

subplot(3,1,3);
plot(f_axis, x_spec); title('Magnitude Spectrum of Signal - X');
xlabel('Frequency (Hz)'); ylabel('Amplitude'); grid on;

figure;
subplot(3,1,1);
plot(t, y); title('Original Signal - Y'); xlabel('Samples'); ylabel('Amplitude'); grid on;

subplot(3,1,2);
plot(t, y_ft); title('FT Signal - Y'); xlabel('Samples'); ylabel('Amplitude'); grid on;

subplot(3,1,3);
plot(f_axis, y_spec); title('Magnitude Spectrum of Signal - Y');
xlabel('Frequency (Hz)'); ylabel('Amplitude'); grid on;

% final plot
figure;
plot(f_axis, x_spec, '-');
hold on
plot(f_axis, y_spec, '--r');

legend('Correctly sampled signal', 'Aliased signal');
```

```
title('Comparison of sufficiently and undersampled signal');
xlabel('Frequency (Hz)'); ylabel('Amplitude'); grid on;
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```

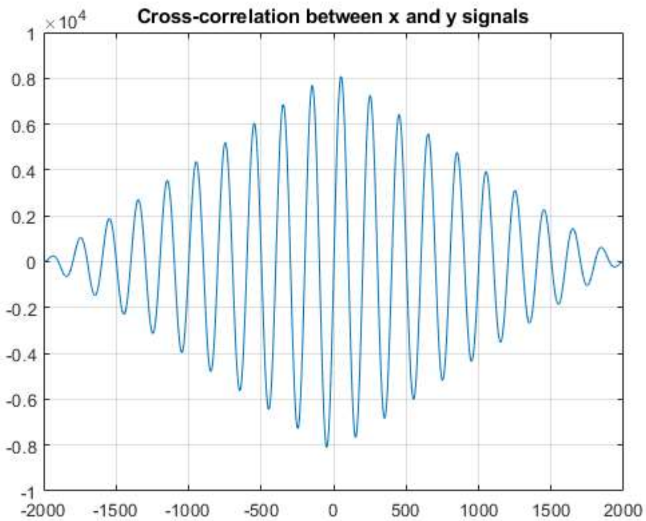






Question 5:

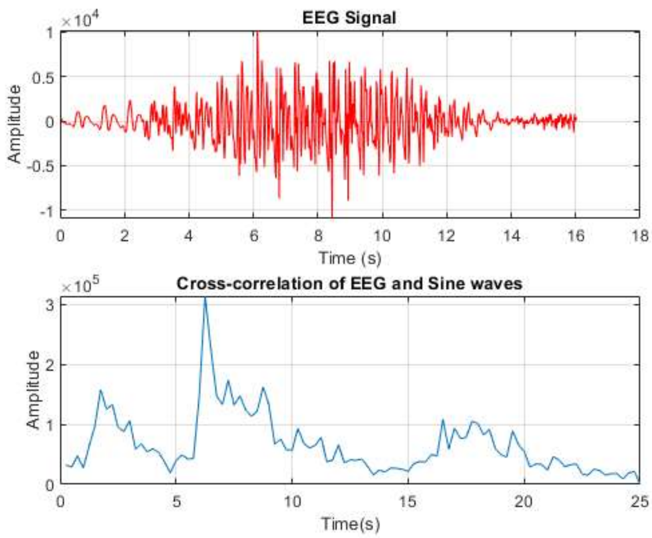
```
%Loading the signal data
s = load('sines1.mat');
d_x = s.x;
d_y = s.y;

fs = 2000;
%f = 10;

% frequency and time vector calculated:
N = length(d_x);
f = (1:N)*fs/N;
t = (1:N)/fs;

% cross-correlation between two the two signals
[c2,lags2] = xcorr(d_x,d_y);

figure;
plot(lags2, c2); grid on;
title('Cross-correlation between x and y signals');
```

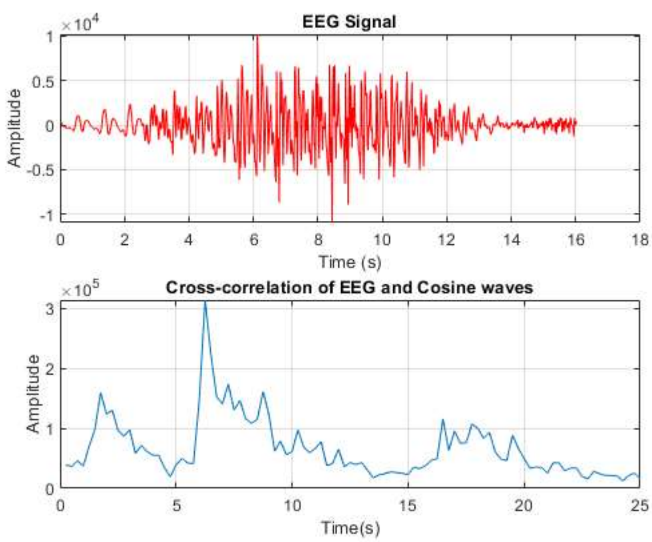


Question 6:

```
s = load('eeg_data.mat');
eeg = s.eeg;

fs = 50;
Ts = 1/fs;

% frequency and time vector calculated:
N = length(eeg);
f = (1:N)*fs/N;
t = (1:N)/fs;

% Correlation with Sinusoidal waves
corr_sin = [];
min_corr_sin = [];
freq_sin = [];

for i = 1:25/0.25

    sine_signal = sin(2*pi*(0.25*i)*t);
    [corr, lag] = xcorr(eeg, sine_signal);

    corr_sin(i) = max(corr);
    min_corr_sin(i) = min(corr);
    freq_sin(i) = i*0.25;

end

figure;
subplot(2,1,1);
plot(t, eeg,'r'); title('EEG Signal'); xlabel('Time (s)'); ylabel('Amplitude'); grid on;

subplot(2,1,2);
plot(freq_sin, corr_sin); title('Cross-correlation of EEG and Sine waves');
xlabel('Time(s)'); ylabel('Amplitude'); grid on;

% getting the max values of correlation
[corr_sin_max, corr_sin_max_ind] = max(corr_sin);
```

```
% Correlation with Cosine waves
corr_cos = [];
min_corr_cos = [];
freq_cos = [];

for j = 1:25/0.25

    cosine_signal = cos(2*pi*(0.25*j)*t);
    [corr2, lag2] = xcorr(eeg, cosine_signal);

    corr_cos(j) = max(corr2);
    min_corr_cos(j) = min(corr2);
    freq_cos(j) = j*0.25;

end

figure;
subplot(2,1,1);
plot(t, eeg,'r'); title('EEG Signal'); xlabel('Time (s)'); ylabel('Amplitude'); grid on;

subplot(2,1,2);
plot(freq_cos, corr_cos); title('Cross-correlation of EEG and Cosine waves');
xlabel('Time(s)'); ylabel('Amplitude'); grid on;

% getting the max values of correlation
[corr_cos_max, corr_cos_max_ind] = max(corr_cos);
```



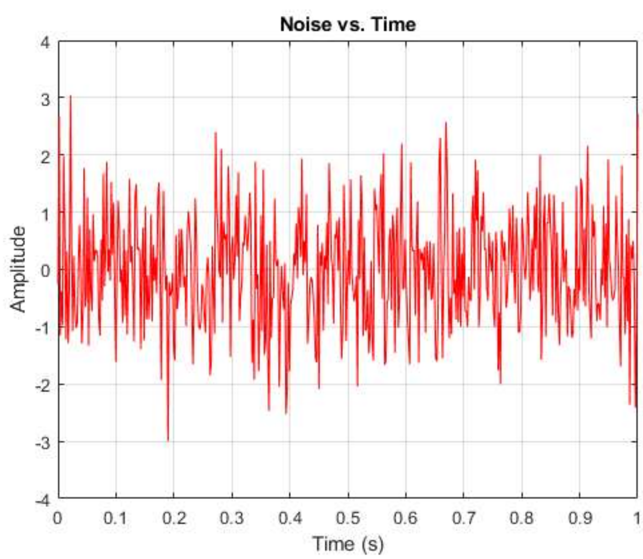


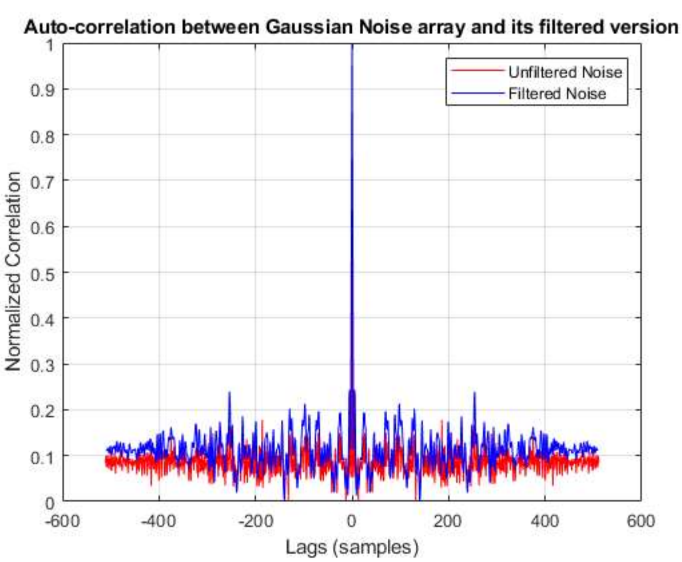
Question 7:

constructing the noise array

```
H = [1/3 1/3 1/3];
noise = randn(1, 512);
```

```
% adding the filter
y = [];
for i = 3:length(noise)

    y(i-2) = (1/3)*(noise(i) + noise(i-1)+ noise(i-2));

end

t = (0:length(noise)-1)*(1/(length(noise)-1));

figure;
plot(t, noise,'r');
title('Noise vs. Time'); xlabel('Time (s)'); ylabel('Amplitude'); grid on;

[noise_corr, noise_lag] = xcorr(noise, noise);
norm_corr = (noise_corr - min(noise_corr))./(max(noise_corr) - min(noise_corr));

[filter_corr, filter_lag] = xcorr(y, y);
norm_corr_2 = (filter_corr - min(filter_corr))./(max(filter_corr) - min(filter_corr));


figure;
plot(noise_lag, norm_corr,'r'); hold on;
plot(filter_lag, norm_corr_2,'b'); hold off;
title('Auto-correlation between Gaussian Noise array and its filtered version');
xlabel('Lags (samples)'); ylabel('Normalized Correlation');
legend('Unfiltered Noise','Filtered Noise'); grid on;

% By normalizing the auto-correlation, it can be seen that white noise if
% uncorrelated sample to sample but the moving average filter has caused
% some correlations between the sample points. This is because the filter
% output depends on the preceeding sample values.
```

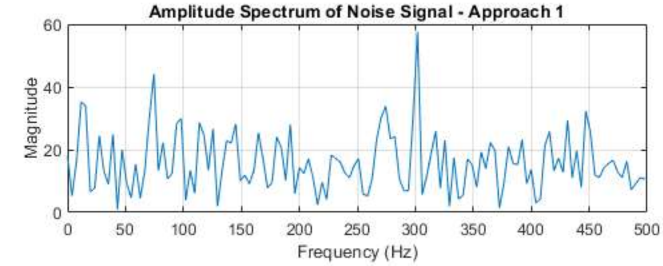


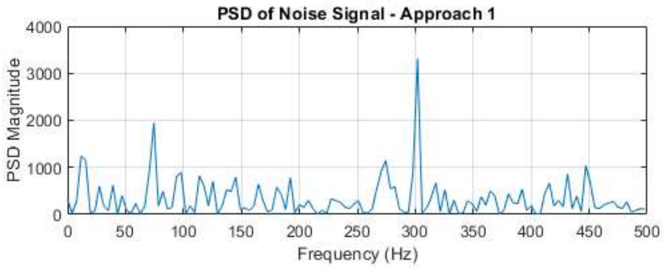


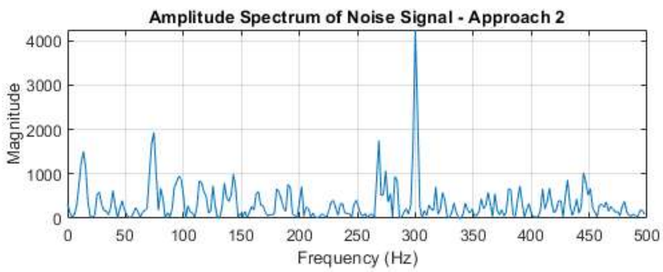
Question 8:

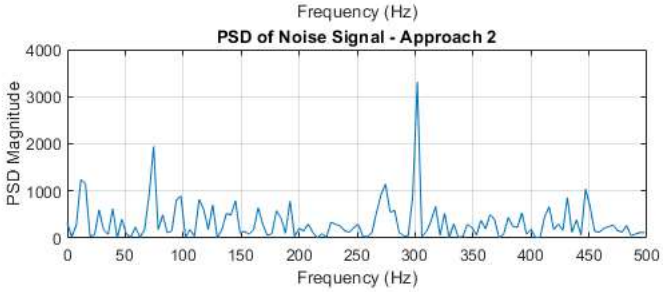
```
[waveform_noise, time, waveform, snr_out] = sig_noise(300, -12, 256);

% Approach one
L = length(waveform_noise);
fs = 1000;
f = fs/(L-1)*(0:L-1);
f1 = f(1:end/2);

% taking the FT of the noise signal
noise_ft = fft(waveform_noise);
noise_ft_abs = abs(noise_ft);
noise_spec = noise_ft_abs(1:L/2);

figure;
subplot(2,1,1); plot(f1, noise_spec);
title('Amplitude Spectrum of Noise Signal - Approach 1'); xlabel('Frequency (Hz)'); ylabel('Magnitude'); grid on;
subplot(2,1,2); plot(f1, noise_spec.^2);
title('PSD of Noise Signal - Approach 1'); xlabel('Frequency (Hz)'); ylabel('PSD Magnitude'); grid on;

% Approach two
[w_corr, w_lags] = xcorr(waveform_noise, waveform_noise);

L2 = length(w_corr);
f_1 = fs/(L2-1)*(0:L2-1);
f2 = f_1(1:end/2);

w_corr_ft = fft(w_corr);
w_corr_ft_abs = abs(w_corr_ft);
w_corr_spec = w_corr_ft_abs(1:L2/2);

figure;
subplot(2,1,1); plot(f2, w_corr_spec);
title('Amplitude Spectrum of Noise Signal - Approach 2'); xlabel('Frequency (Hz)'); ylabel('Magnitude'); grid on;
subplot(2,1,2); plot(f1, noise_spec.^2);
title('PSD of Noise Signal - Approach 2'); xlabel('Frequency (Hz)'); ylabel('PSD Magnitude'); grid on;
```

```
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
```



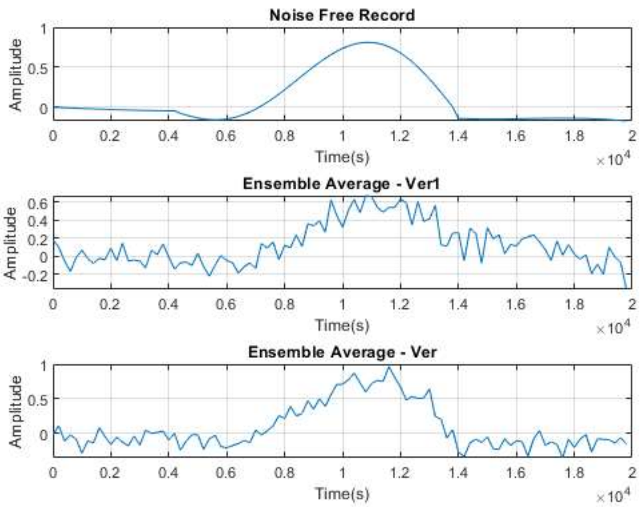






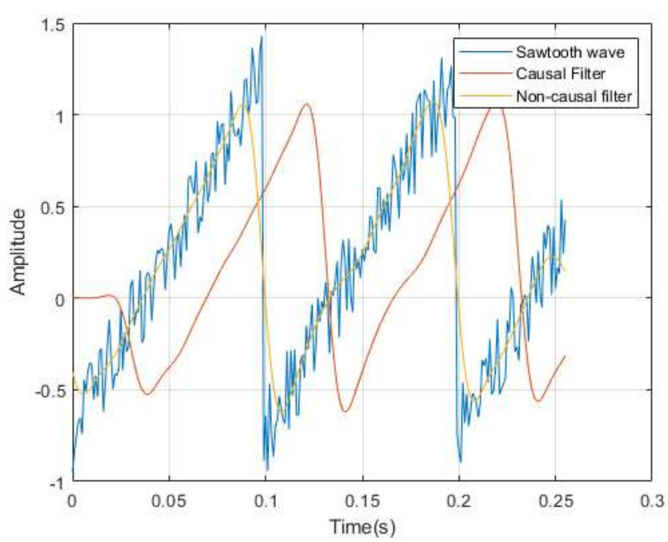
Question 9:

```
s = load('ver_problem2.mat');

actual_ver = s.actual_ver; % the noise-free VER
ver = s.ver; % 100 noise signals recorded with a fixed reference signal
ver1 = s.ver1; % 100 records recorded with a reference signal that varies randomly by +-150 ms

ts = 0.005;
fs = 1/ts;

% ensemble averages
ver_avg = mean(ver);
ver1_avg = mean(ver1);

N = length(actual_ver);
N1 = length(ver);
N2 = length(ver1);
t = (0:N-1)*fs;
t1 = (0:N1-1)*fs;
t2 = (0:N2-1)*fs;

% plotting the ensemble averages
figure;
subplot(3,1,1);
plot(t, actual_ver); title('Noise Free Record');
xlabel('Time(s)'); ylabel('Amplitude'); grid on;
subplot(3,1,2);
plot(t2, ver1_avg); title('Ensemble Average - Ver1');
xlabel('Time(s)'); ylabel('Amplitude'); grid on;
subplot(3,1,3);
plot(t1, ver_avg); title('Ensemble Average - Ver');
xlabel('Time(s)'); ylabel('Amplitude'); grid on;
```



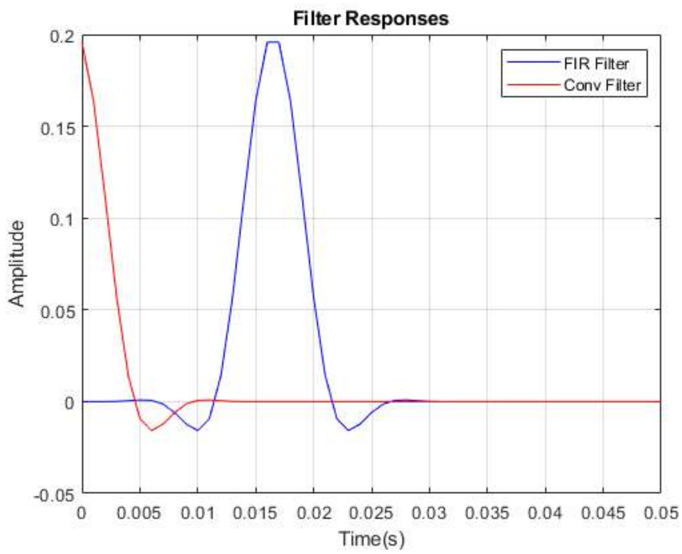
Question 10:

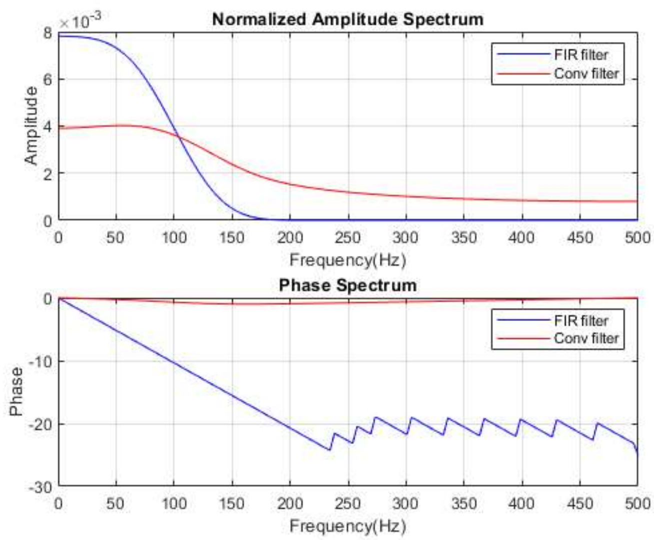
```
s = load('sawth.mat'); % loading the dataset
signal = s.x;

fc = 40; % cutoff frequency
fs = 1000;
ts = 1/fs;
order = 65;

L = length(signal);
t = (0:L-1)/fs;

wc = fc/(fs/2);
b = fir1(order, wc, 'low', blackmanharris(order+1));

% Adding the filter in two ways
filt_FIR = filter(b, 1, signal);
filt_conv = conv(signal, b, 'same');

figure;
plot(t, signal); hold on; plot(t, filt_FIR); hold on; plot(t, filt_conv); hold off;
xlabel('Time(s)'); ylabel('Amplitude'); grid on;
legend('Sawtooth wave','Causal Filter', 'Non-causal filter');

% from the plots above, it can be seen that due to the introduced delay
% from the casaul filter, the peaks show up later in time using the FIR
% filter. However, in the plot where the filtering is done through
% convolution, the output of the filtered signal is occuring before the
% actual Sawtooth wave signal which indicates non-causality.
```

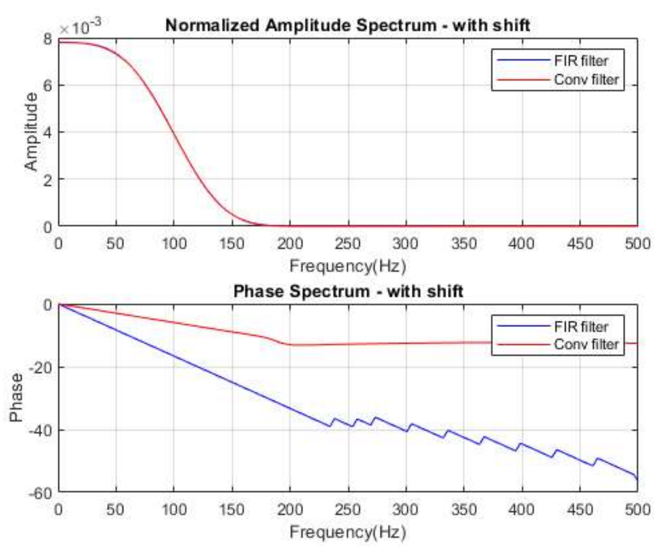


Question 11:

```
fs = 1000;
ts = 1/fs;
fc = 100;
wc = fc/(fs/2);

order = 33;

fir = fir1(order, wc, 'low', blackmanharris(order+1));

impulse = zeros(1, 256);
impulse(1) = 1;

filt_FIR = filter(fir, 1, impulse);
filt_conv = conv(impulse, fir, 'same');

t = (0:length(impulse)-1)*ts;
index = find(t == 0.05);

% plotting the filter responses:
figure;
plot(t(1:index), filt_FIR(1:index),'-b'); hold on;
plot(t(1:index), filt_conv(1:index), '-r'); hold off;
title('Filter Responses'); xlabel('Time(s)'); ylabel('Amplitude');
legend('FIR Filter','Conv Filter'); grid on;

L = length(impulse);
FIR_ft = fft(filt_FIR)/L;
conv_ft = fft(filt_conv)/L;
f = fs*(0:(L/2))/L;

% plotting the amplitude spectrums:
spec_FIR = abs(FIR_ft(1:L/2+1));
spec_conv = abs(conv_ft(1:L/2+1));

phase_FIR = unwrap(angle(FIR_ft(1:L/2+1)));
phase_conv = unwrap(angle(conv_ft(1:L/2+1)));

figure;
subplot(2,1,1);
plot(f, 2*spec_FIR,'-b'); hold on; plot(f, 2*spec_conv,'-r'); hold off;
title('Normalized Amplitude Spectrum'); xlabel('Frequency(Hz)'); ylabel('Amplitude');
grid on; legend('FIR filter','Conv filter');

subplot(2,1,2);
plot(f, phase_FIR,'-b'); hold on; plot(f, phase_conv,'-r'); hold off;
title('Phase Spectrum'); xlabel('Frequency(Hz)'); ylabel('Phase');
grid on; legend('FIR filter','Conv filter');

% Adding the shift:
impulse2 = [zeros(1,10) 1 zeros(1,245)];
filt2_FIR = filter(fir, 1, impulse2);
filt2_conv = conv(impulse2, fir, 'same');

FIR_ft2 = fft(filt2_FIR)/L;
conv_ft2 = fft(filt2_conv)/L;

spec2_FIR = abs(FIR_ft2(1:L/2+1));
spec2_conv = abs(conv_ft2(1:L/2+1));

phase2_FIR = unwrap(angle(FIR_ft2(1:L/2+1)));
phase2_conv = unwrap(angle(conv_ft2(1:L/2+1)));

figure;
subplot(2,1,1);
plot(f, 2*spec2_FIR,'-b'); hold on; plot(f, 2*spec2_conv,'-r'); hold off;
title('Normalized Amplitude Spectrum - with shift'); xlabel('Frequency(Hz)'); ylabel('Amplitude');
grid on; legend('FIR filter','Conv filter');

subplot(2,1,2);
plot(f, phase2_FIR,'-b'); hold on; plot(f, phase2_conv,'-r'); hold off;
title('Phase Spectrum - with shift'); xlabel('Frequency(Hz)'); ylabel('Phase');
grid on; legend('FIR filter','Conv filter');
```



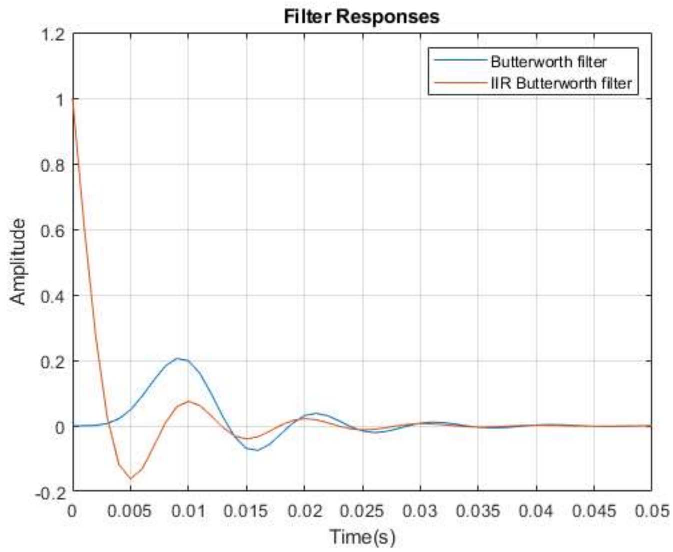


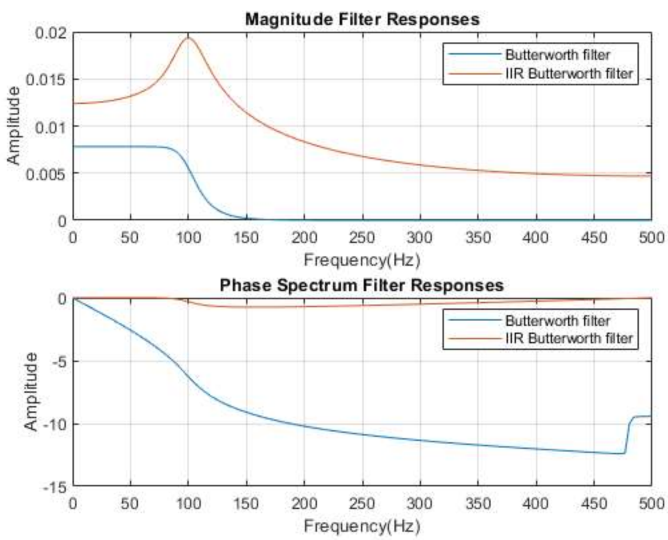


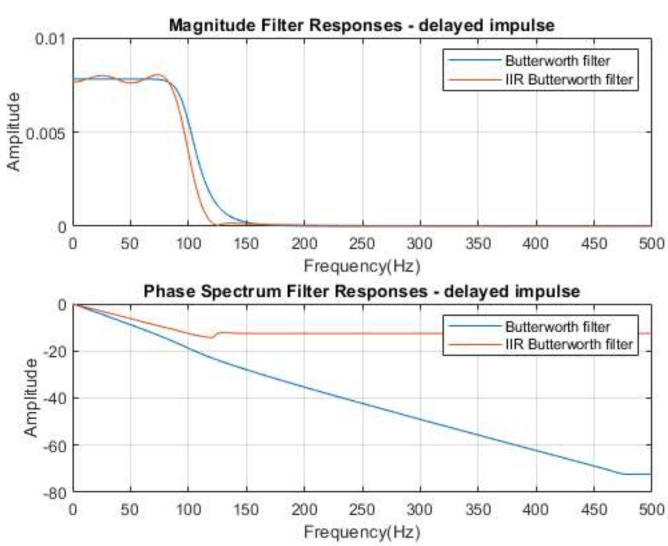
Question 12:

```
fc = 100;
fs = 1000;
ts = 1/fs;
wc = fc/(fs/2);

impulse = [1 zeros(1, 255)];
```

```
L = length(impulse);
[b, a] = butter(8, wc);
y1 = filter(b, a, impulse);
y2 = filtfilt(b, a, impulse);

t = (0:length(impulse)-1).*ts;
index = find(t == 0.05);

% plotting the filter responses:
figure;
plot(t(1:index), y1(1:index)); hold on; plot(t(1:index), y2(1:index)); hold off;
title('Filter Responses'); xlabel('Time(s)'); ylabel('Amplitude');
legend('Butterworth filter','IIR Butterworth filter'); grid on;

% Defining the magnitude and phase response and plotting them:
bw_ft = fft(y1)/L;
iirbw_ft = fft(y2)/L;

f = fs*(0:(L/2))/L;
bw_ft = bw_ft(1:(L/2)+1);
iirbw_ft = iirbw_ft(1:(L/2)+1);

bw_mag = abs(bw_ft).*2;
iirbw_mag = abs(iirbw_ft).*2;

bw_phase = unwrap(angle(bw_ft(1:(L/2)+1)));
iirbw_phase = unwrap(angle(iirbw_ft(1:(L/2)+1)));

figure;
subplot(2,1,1); plot(f, bw_mag); hold on; plot(f, iirbw_mag); hold off;
title('Magnitude Filter Responses'); xlabel('Frequency(Hz)'); ylabel('Amplitude');
legend('Butterworth filter','IIR Butterworth filter'); grid on;

subplot(2,1,2); plot(f, bw_phase); hold on; plot(f, iirbw_phase); hold off;
title('Phase Spectrum Filter Responses'); xlabel('Frequency(Hz)'); ylabel('Amplitude');
legend('Butterworth filter','IIR Butterworth filter'); grid on;

% showing absence of ripples:
impulse2 = [zeros(1,20) 1 zeros(1, 235)];

y3 = filter(b,a,impulse2);
y4 = filtfilt(b, a, impulse2);
bw_ft2 = fft(y3)/L;
iirbw_ft2 = fft(y4)/L;

f3 = fs*(0:(L/2))/L;
bw_ft2 = bw_ft2(1:(L/2)+1);
iirbw_ft2 = iirbw_ft2(1:(L/2)+1);

bw_mag2 = abs(bw_ft2).*2;
iirbw_mag2 = abs(iirbw_ft2).*2;

bw_phase2 = unwrap(angle(bw_ft2(1:(L/2)+1)));
iirbw_phase2 = unwrap(angle(iirbw_ft2(1:(L/2)+1)));

figure;
subplot(2,1,1); plot(f3, bw_mag2); hold on; plot(f3, iirbw_mag2); hold off;
title('Magnitude Filter Responses - delayed impulse'); xlabel('Frequency(Hz)'); ylabel('Amplitude');
legend('Butterworth filter','IIR Butterworth filter'); grid on;

subplot(2,1,2); plot(f3, bw_phase2); hold on; plot(f3, iirbw_phase2); hold off;
title('Phase Spectrum Filter Responses - delayed impulse'); xlabel('Frequency(Hz)'); ylabel('Amplitude');
legend('Butterworth filter','IIR Butterworth filter'); grid on;
```







Question 13:

data given:

```
N = 1000;
SNR = -12;
fs = 1000;
ts = 1/fs;
```

```
f1 = 100;
f2 = 240;
f3 = 280;
f4 = 400;
t = (1:N)/fs;

sine_wave = sig_noise([f1, f2, f3, f4], SNR, 1000);

figure;
subplot(3,2,1);
plot(t, sine_wave);
title('Four Sinusoids'); xlabel('Time(s)'); ylabel('Amplitude'); grid on;

subplot(3,2,2);
pwelch(sine_wave,N,[],[],fs);

[Pxx_17,F_17] = pyulear(sine_wave,17,1024,N);
subplot(3,2,3);
plot(F_17, Pxx_17);
title('Magnitude Spectrum-order=17'); xlabel('Frequency(Hz)');
ylabel('Amplitude'); grid on;

[Pxx_20,F_20] = pyulear(sine_wave,20,1024,N);
subplot(3,2,4);
plot(F_20, Pxx_20);
title('Magnitude Spectrum-order=20'); xlabel('Frequency(Hz)');
ylabel('Amplitude'); grid on;

[Pxx_30,F_30] = pyulear(sine_wave,30,1024,N);
subplot(3,2,5);
plot(F_30, Pxx_30);
title('Magnitude Spectrum-order=30'); xlabel('Frequency(Hz)');
ylabel('Amplitude'); grid on;

[Pxx_50,F_50] = pyulear(sine_wave,50,1024,N);
subplot(3,2,6);
plot(F_50, Pxx_50);
title('Magnitude Spectrum-order=50'); xlabel('Frequency(Hz)');
ylabel('Amplitude'); grid on;
```