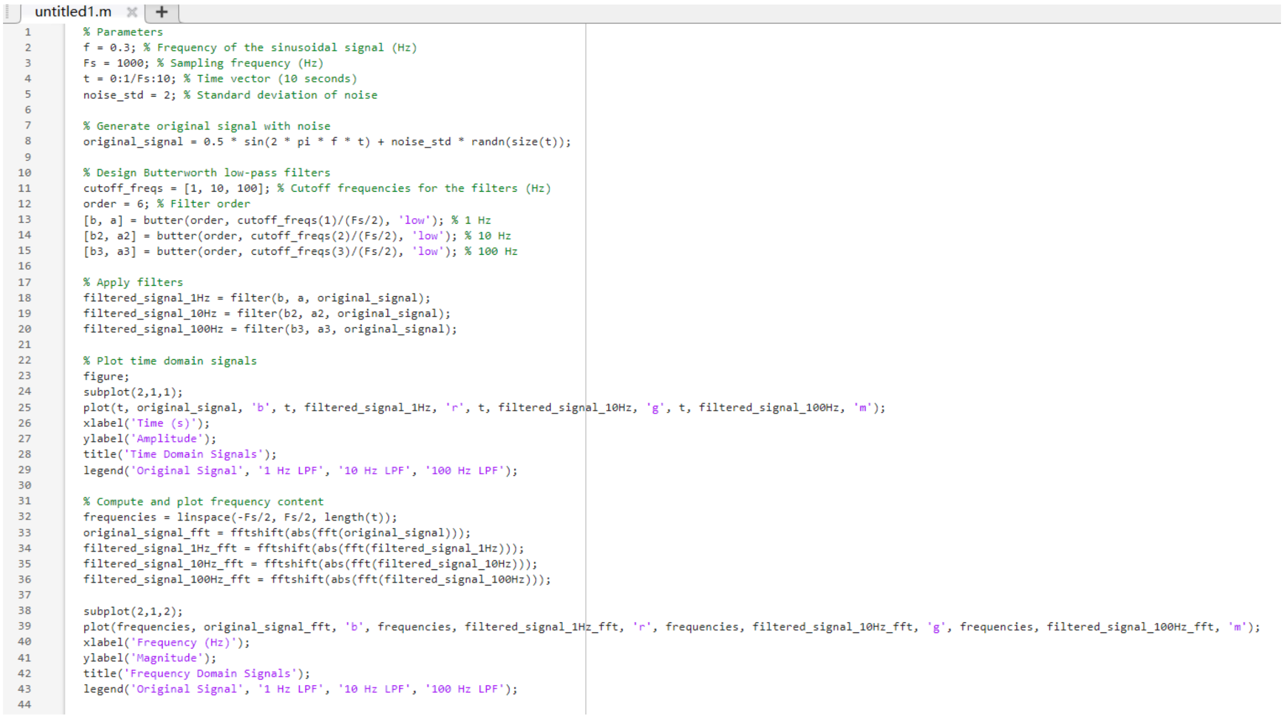
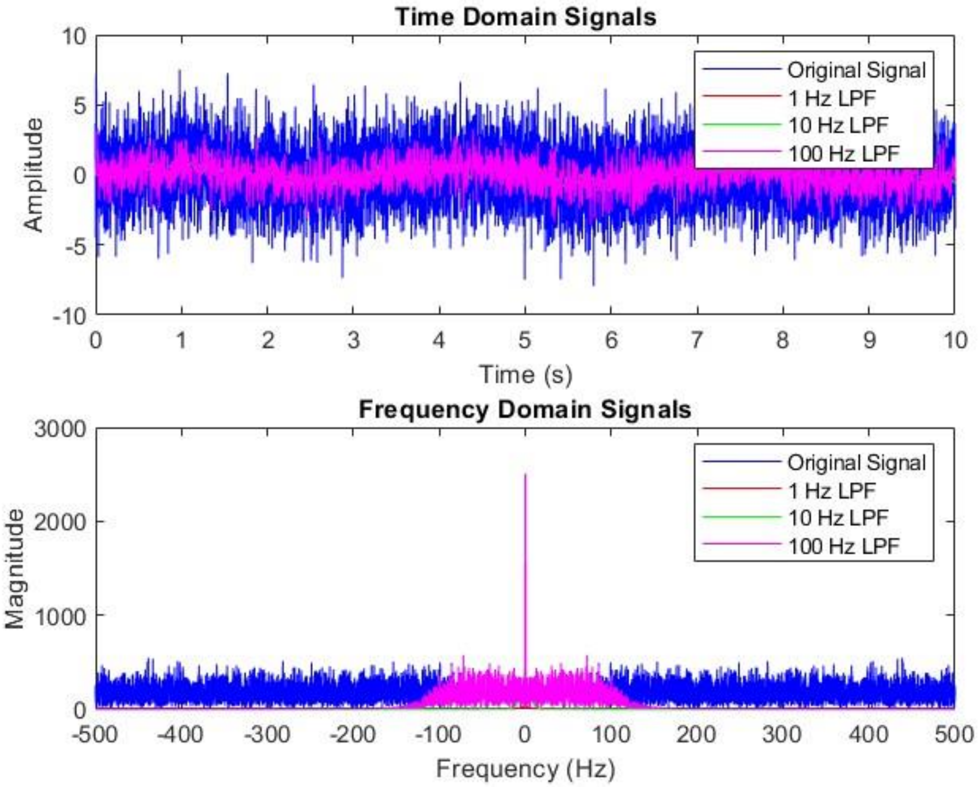
HW-5 noisy signals filtered by low pass filters

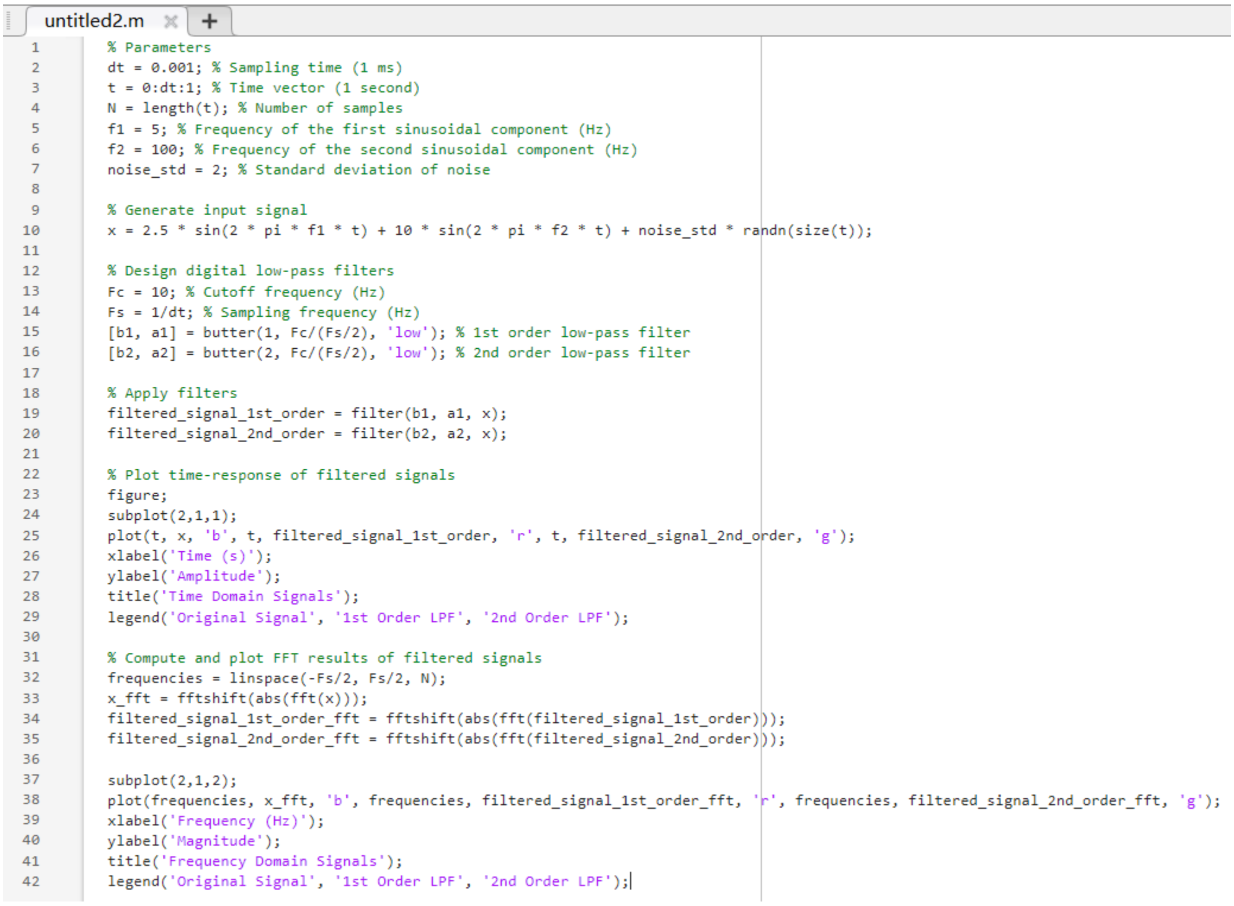
Matlab simulations -1



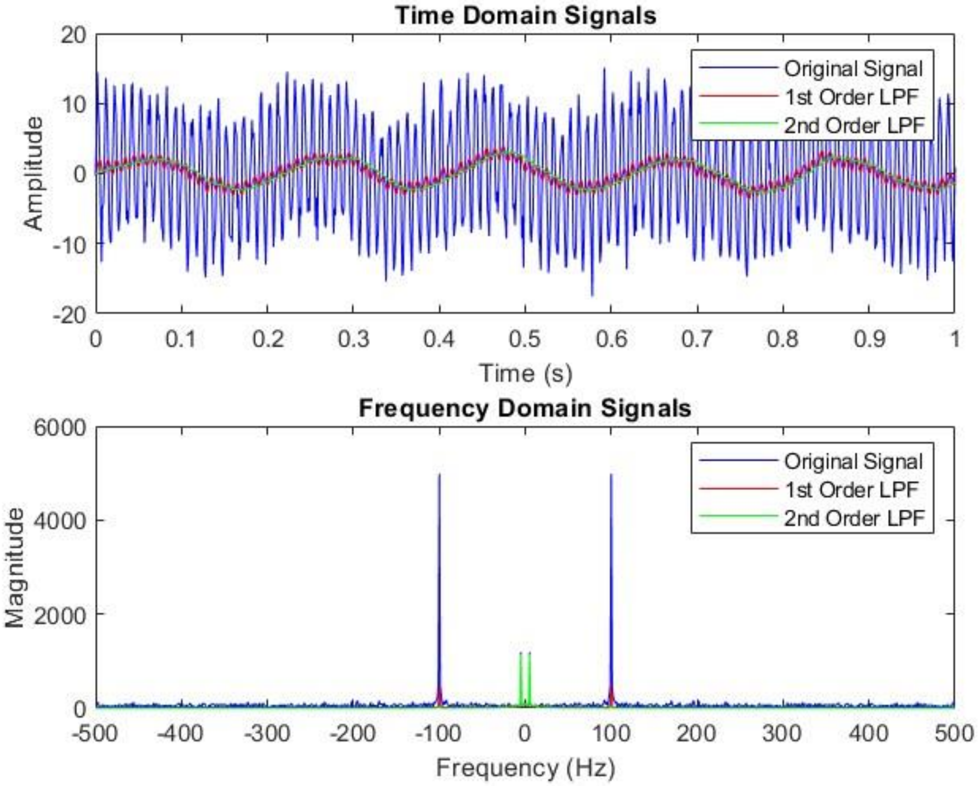
```matlab
% Parameters
f = 0.3; % Frequency of the sinusoidal signal (Hz)
Fs = 1000; % Sampling frequency (Hz)
t = 0:1/Fs:10; % Time vector (10 seconds)
noise_std = 2; % Standard deviation of noise

% Generate original signal with noise
original_signal = 0.5 * sin(2 * pi * f * t) + noise_std * randn(size(t));

% Design Butterworth low-pass filters
cutoff_freqs = [1, 10, 100]; % Cutoff frequencies for the filters (Hz)
order = 6; % Filter order
[b, a] = butter(order, cutoff_freqs(1)/(Fs/2), 'low'); % 1 Hz
[b2, a2] = butter(order, cutoff_freqs(2)/(Fs/2), 'low'); % 10 Hz
[b3, a3] = butter(order, cutoff_freqs(3)/(Fs/2), 'low'); % 100 Hz

% Apply filters
filtered_signal_1Hz = filter(b, a, original_signal);
filtered_signal_10Hz = filter(b2, a2, original_signal);
filtered_signal_100Hz = filter(b3, a3, original_signal);

% Plot time domain signals
figure;
subplot(2,1,1);
plot(t, original_signal, 'b', t, filtered_signal_1Hz, 'r', t, filtered_signal_10Hz, 'g', t, filtered_signal_100Hz, 'm');
xlabel('Time (s)');
ylabel('Amplitude');
title('Time Domain Signals');
legend('Original Signal', '1 Hz LPF', '10 Hz LPF', '100 Hz LPF');

% Compute and plot frequency content
frequencies = linspace(-Fs/2, Fs/2, length(t));
original_signal_fft = fftshift(abs(fft(original_signal)));
filtered_signal_1Hz_fft = fftshift(abs(fft(filtered_signal_1Hz)));
filtered_signal_10Hz_fft = fftshift(abs(fft(filtered_signal_10Hz)));
filtered_signal_100Hz_fft = fftshift(abs(fft(filtered_signal_100Hz)));

subplot(2,1,2);
plot(frequencies, original_signal_fft, 'b', frequencies, filtered_signal_1Hz_fft, 'r', frequencies, filtered_signal_10Hz_fft, 'g', frequencies, filtered_signal_100Hz_fft, 'm');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Frequency Domain Signals');
legend('Original Signal', '1 Hz LPF', '10 Hz LPF', '100 Hz LPF');

```



Matlab simulations -2



```matlab
% Parameters
dt = 0.001; % Sampling time (1 ms)
t = 0:dt:1; % Time vector (1 second)
N = length(t); % Number of samples
f1 = 5; % Frequency of the first sinusoidal component (Hz)
f2 = 100; % Frequency of the second sinusoidal component (Hz)
noise_std = 2; % Standard deviation of noise

% Generate input signal
x = 2.5 * sin(2 * pi * f1 * t) + 10 * sin(2 * pi * f2 * t) + noise_std * randn(size(t));

% Design digital low-pass filters
Fc = 10; % Cutoff frequency (Hz)
Fs = 1/dt; % Sampling frequency (Hz)
[b1, a1] = butter(1, Fc/(Fs/2), 'low'); % 1st order low-pass filter
[b2, a2] = butter(2, Fc/(Fs/2), 'low'); % 2nd order low-pass filter

% Apply filters
filtered_signal_1st_order = filter(b1, a1, x);
filtered_signal_2nd_order = filter(b2, a2, x);

% Plot time-response of filtered signals
figure;
subplot(2,1,1);
plot(t, x, 'b', t, filtered_signal_1st_order, 'r', t, filtered_signal_2nd_order, 'g');
xlabel('Time (s)');
ylabel('Amplitude');
title('Time Domain Signals');
legend('Original Signal', '1st Order LPF', '2nd Order LPF');

% Compute and plot FFT results of filtered signals
frequencies = linspace(-Fs/2, Fs/2, N);
x_fft = fftshift(abs(fft(x)));
filtered_signal_1st_order_fft = fftshift(abs(fft(filtered_signal_1st_order)));
filtered_signal_2nd_order_fft = fftshift(abs(fft(filtered_signal_2nd_order)));

subplot(2,1,2);
plot(frequencies, x_fft, 'b', frequencies, filtered_signal_1st_order_fft, 'r', frequencies, filtered_signal_2nd_order_fft, 'g');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Frequency Domain Signals');
legend('Original Signal', '1st Order LPF', '2nd Order LPF');
```


Time Domain Signals
Amplitude
Time (s)
Original Signal
1st Order LPF
2nd Order LPF
Frequency Domain Signals
Magnitude
Frequency (Hz)
Original Signal
1st Order LPF
2nd Order LPF