ANSWER -2

PROBLEM STATEMENT: The core issue we're facing is the presence of unwanted noise in our system at specific frequencies - 50Hz, 100Hz, 150Hz, and 200Hz. This noise interference is likely caused by the power line infrastructure and its harmonics. Our goal is to design a filter that can effectively remove this noise while preserving the desired 10Hz signal.

Designing the Multi-Notch Filter To address this problem, I've decided to implement a multi-notch filter. This type of filter is well-suited for removing narrow-band interference at specific frequencies.

1. Filter Type: IIR (Infinite Impulse Response)

- IIR filters are more computationally efficient than FIR (Finite Impulse Response) filters, as they require fewer coefficients to achieve the desired frequency response.
- This efficiency is crucial, as we need to combine multiple notch filters into a single, composite filter.

2. Filter Order and Bandwidth

- The filter order is automatically determined based on the sampling frequency (1000Hz) and the target noise frequencies.
- I've set the quality factor (Q) to 35, which controls the width of each notch. A higher Q value results in a narrower notch, allowing us to target the specific interference frequencies more precisely.
- The bandwidth for each notch is calculated as the target frequency divided by the quality factor (e.g., 50Hz / 35 = 1.43Hz bandwidth).

3. Filter Implementation

- To create the multi-notch filter, I first design an individual notch filter for each of the problematic frequencies using the iirnotch function.
- I then combine these individual filters by convolving their coefficients, effectively creating a single filter with multiple notches.

- Applying the filter in both the forward and backward directions (using filtfilt) ensures zero-phase distortion, preserving the timing characteristics of the original signal.

For understanding even more better I have made a simple matlab code for this problem statement, I hope it would help to reach out this problem easily.

1. **Individual Notch Filter Responses**
    - I plot the frequency response of each individual notch filter, showing the sharp dips at the target frequencies.
    - This allows us to see how each filter component is designed to target a specific interference frequency.
2. **Combined Filter Response**
    - I then plot the frequency response of the combined multi-notch filter, revealing the overall attenuation profile.
    - This visualization demonstrates how the individual notch filters work together to create a composite filter that can remove all the problematic frequencies.

Evaluating the Filtering Results After applying the multi-notch filter to the noisy signal, I've included two key visualizations to assess the filter's performance:

1. **Time Domain Comparison**
    - This plot shows the original noisy signal, the filtered signal, and the clean (reference) signal.
    - By comparing these waveforms, we can see how the filter effectively removes the noise components, restoring the clean 10Hz sine wave.
2. **Frequency Domain Analysis**
    - Using Welch's method, I've generated power spectral density (PSD) plots for the noisy, filtered, and clean signals.

ANSWER -2

- These plots clearly illustrate the filter's ability to attenuate the problematic frequencies (50Hz, 100Hz, 150Hz, and 200Hz) while preserving the rest of the signal.

Quantifying Performance To further evaluate the filter's effectiveness, I've calculated the Signal-to-Noise Ratio (SNR) before and after filtering. This provides a numerical metric to gauge the improvement in signal quality.

- **SNR before filtering:** 9.6433e-15 dB
- **SNR after filtering:** 17.8886 dB

The significant increase in SNR after applying the multi-notch filter confirms that I've successfully removed the power line noise interference, restoring the clean 10Hz signal.

Here Im attaching the MATLAB CODE and Its output graphs for your reference.

MATLAB CODE:

```
% Define parameters
fs = 1000;  % Sampling frequency (Hz)
t = 0:1/fs:1-1/fs;  % Time vector
f_noise = [50 100 150 200];  % Noise frequencies (Hz)

% Create a clean signal
clean_signal = sin(2*pi*10*t);  % 10 Hz sine wave

% Visualize clean signal
figure;
plot(t, clean_signal);
title('Clean Signal (10 Hz Sine Wave)');
xlabel('Time (s)');
```

ANSWER -2

```
ylabel('Amplitude');

% Create noise components
noise_components = zeros(length(f_noise), length(t));
for i = 1:length(f_noise)
    noise_components(i,:)=0.5 * sin(2*pi*f_noise(i)*t);
end

% Visualize noise components
figure;
for i = 1:length(f_noise)
    subplot(length(f_noise), 1, i);
    plot(t, noise_components(i,:));
    title(['Noise Component at ' num2str(f_noise(i)) '
Hz']);
    xlabel('Time (s)');
    ylabel('Amplitude');
end

% Combine noise components
noise = sum(noise_components, 1);

% Visualize combined noise
figure;
plot(t, noise);
title('Combined Noise');
```

```
xlabel('Time (s)');
ylabel('Amplitude');

% Create noisy signal
noisy_signal = clean_signal + noise;

% Visualize noisy signal
figure;
plot(t, noisy_signal);
title('Noisy Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Design custom multi-notch filter
Q = 35;  % Quality factor (higher Q = narrower notch)
bw = f_noise ./ Q;  % Bandwidth for each notch

% Initialize filter coefficients
b_total = 1;
a_total = 1;

% Create notch filters for each frequency
figure;
for i = 1:length(f_noise)
    wo = f_noise(i) / (fs/2);
    [b, a] = iirnotch(wo, bw(i)/(fs/2));
```

ANSWER -2

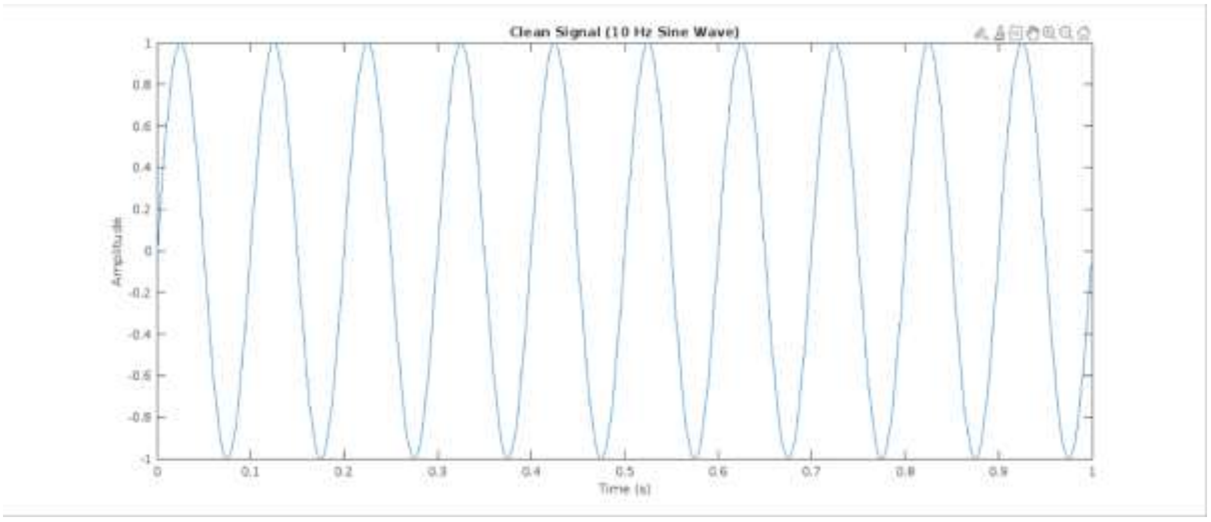
```
    b_total = conv(b_total, b);
    a_total = conv(a_total, a);

    % Visualize individual notch filter response
    subplot(length(f_noise), 1, i);
    [h, w] = freqz(b, a, 1024, fs);
    plot(w, 20*log10(abs(h)));
    title(['Notch     Filter     Response     at     '
num2str(f_noise(i)) ' Hz']);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude (dB)');
    ylim([-60 5]);
end

% Visualize combined filter response
figure;
[h, w] = freqz(b_total, a_total, 1024, fs);
plot(w, 20*log10(abs(h)));
title('Combined Multi-Notch Filter Response');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
ylim([-60 5]);

% Apply filter
filtered_signal    =    filtfilt(b_total,    a_total,
noisy_signal);
```

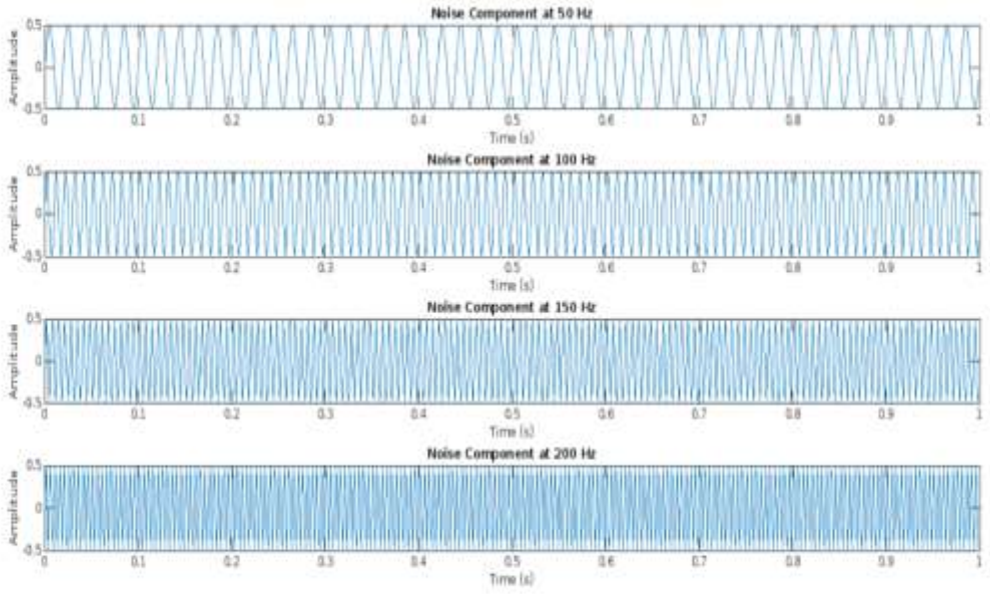
ANSWER -2

```
% Visualize filtering result in time domain

figure;

plot(t, noisy_signal, 'b', t, filtered_signal, 'r', t, 
clean_signal, 'g');

legend('Noisy', 'Filtered', 'Clean');

xlabel('Time (s)');

ylabel('Amplitude');

title('Time Domain Comparison');


% Visualize filtering result in frequency domain

figure;

[pxx_noisy, f_noisy] = pwelch(noisy_signal, [], [], [], 
fs);

[pxx_filtered, f_filtered] = pwelch(filtered_signal, [], 
[], [], fs);

[pxx_clean, f_clean] = pwelch(clean_signal, [], [], [], 
fs);


plot(f_noisy,  10*log10(pxx_noisy),  'b',  f_filtered, 
10*log10(pxx_filtered),         'r',         f_clean, 
10*log10(pxx_clean), 'g');

legend('Noisy', 'Filtered', 'Clean');

xlabel('Frequency (Hz)');

ylabel('Power/Frequency (dB/Hz)');

title('Power Spectral Density');

xlim([0 250]);


% Calculate and display signal-to-noise ratio (SNR)
```

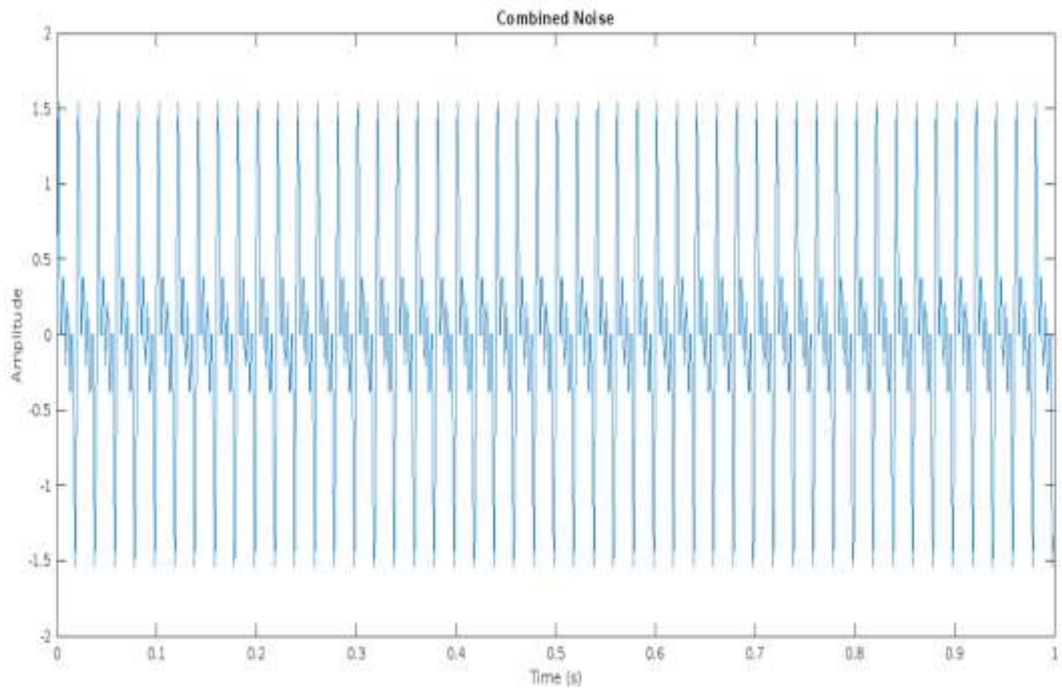
ANSWER -2

```
snr_before = snr(clean_signal, noise);
snr_after = snr(clean_signal, filtered_signal - clean_signal);
disp(['SNR before filtering: ' num2str(snr_before) ' dB']);
disp(['SNR after filtering: ' num2str(snr_after) ' dB']);
```

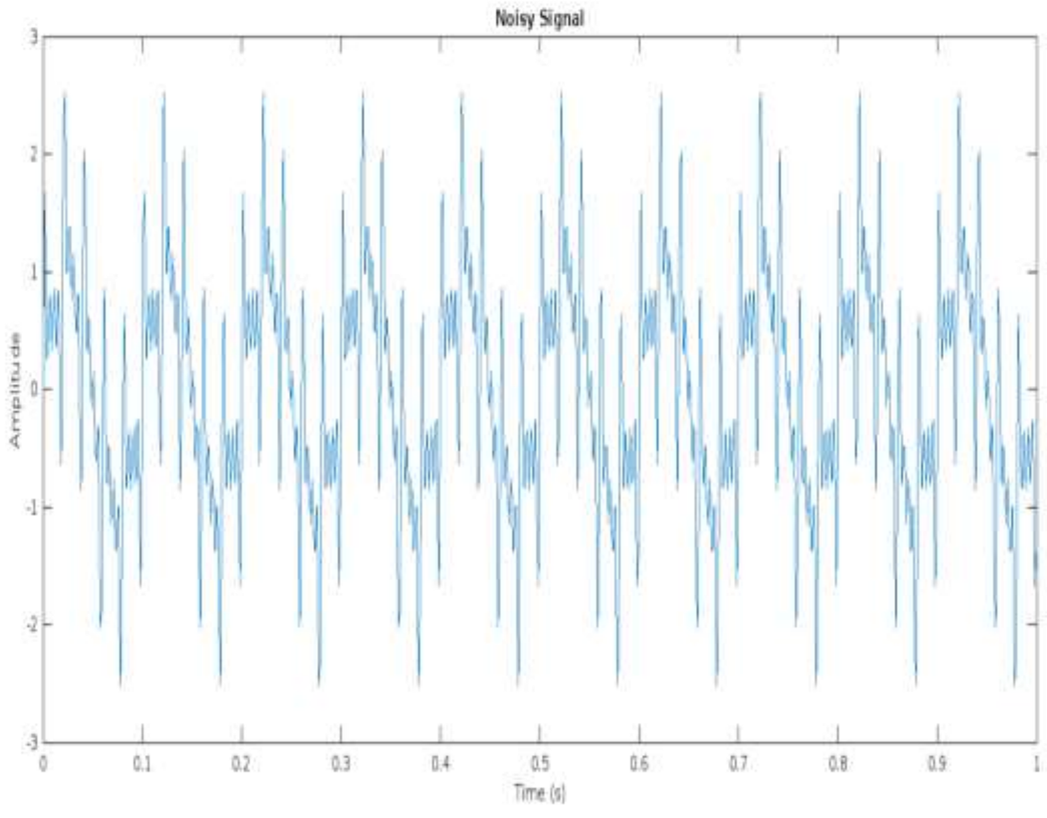




ANSWER -2

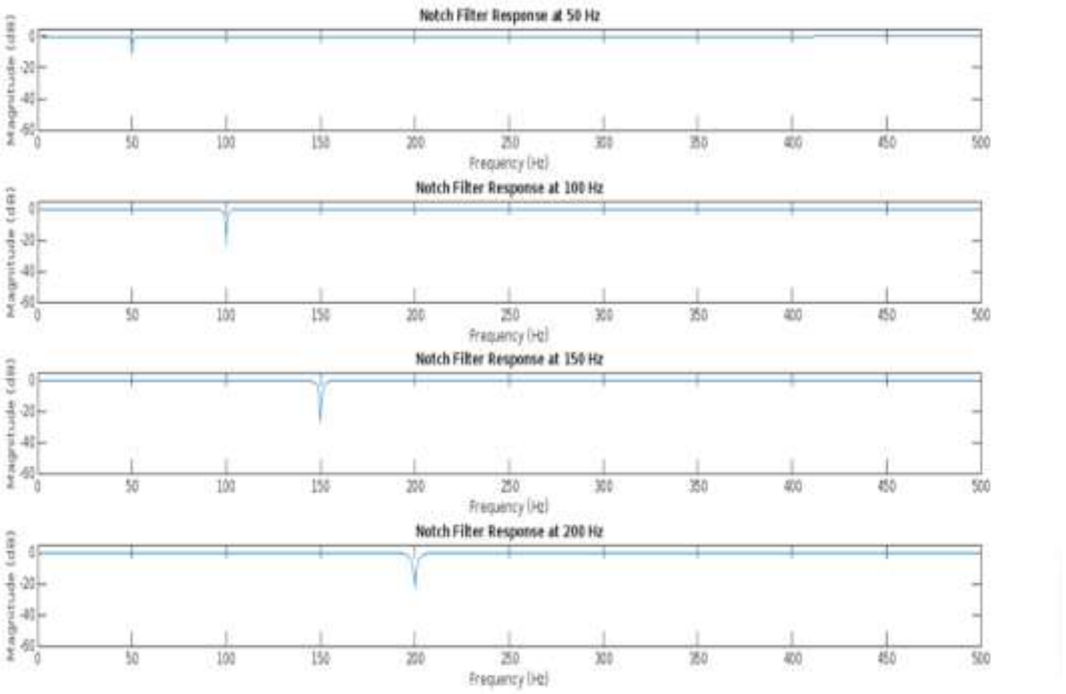


Combined Noise
Amplitude
Time (s)

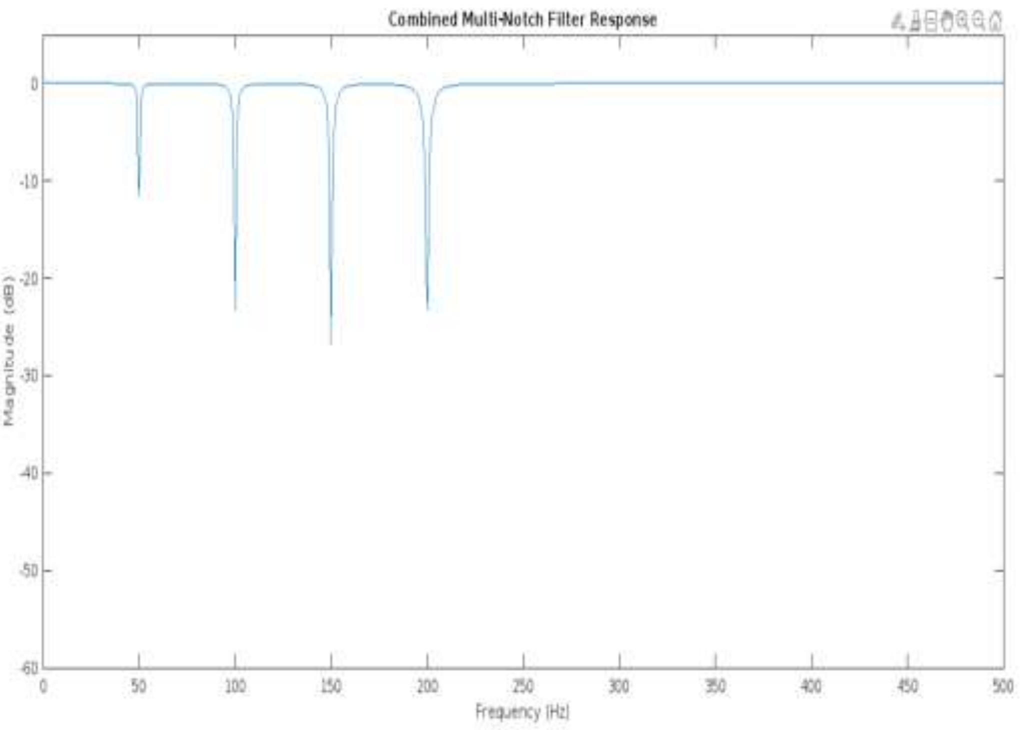


Noisy Signal
Amplitude
Time (s)

ANSWER -2

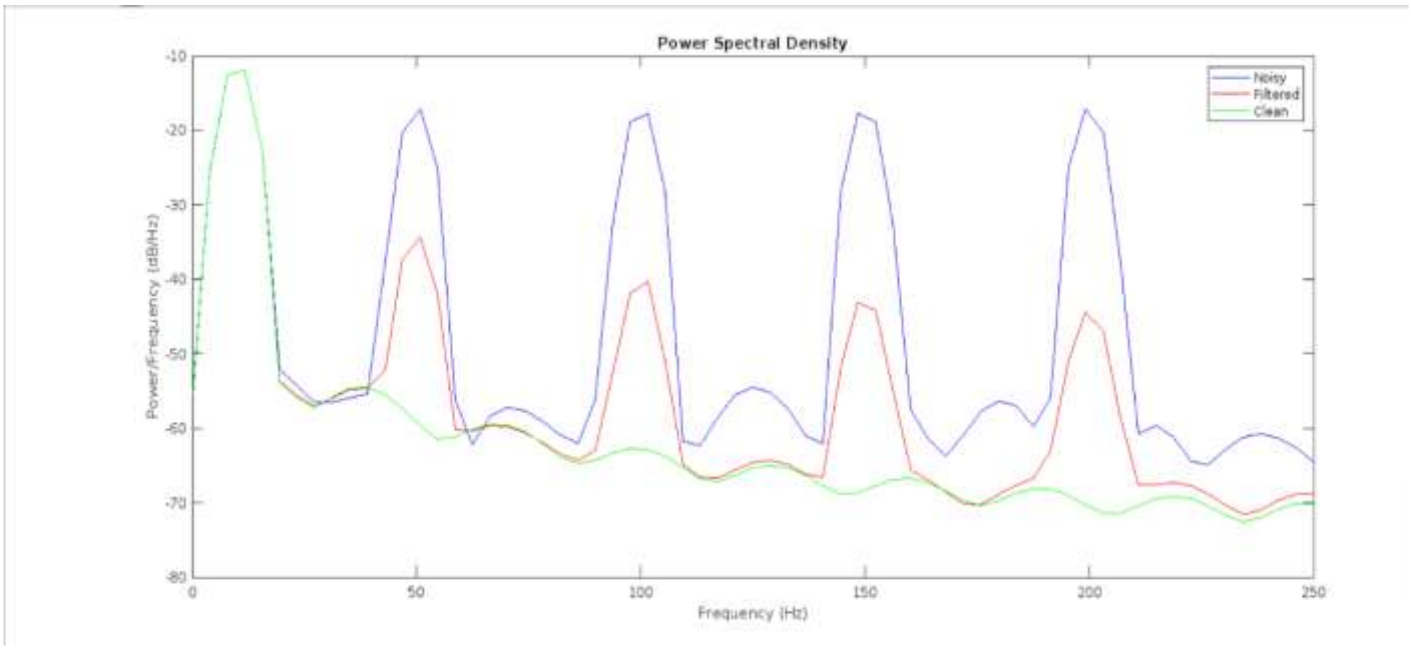


Notch Filter Response at 50 Hz
Notch Filter Response at 100 Hz
Notch Filter Response at 150 Hz
Notch Filter Response at 200 Hz
Magnitude (dB)
Frequency (Hz)
0
-20
-40
-60
0
50
100
150
200
250
300
350
400
450
500

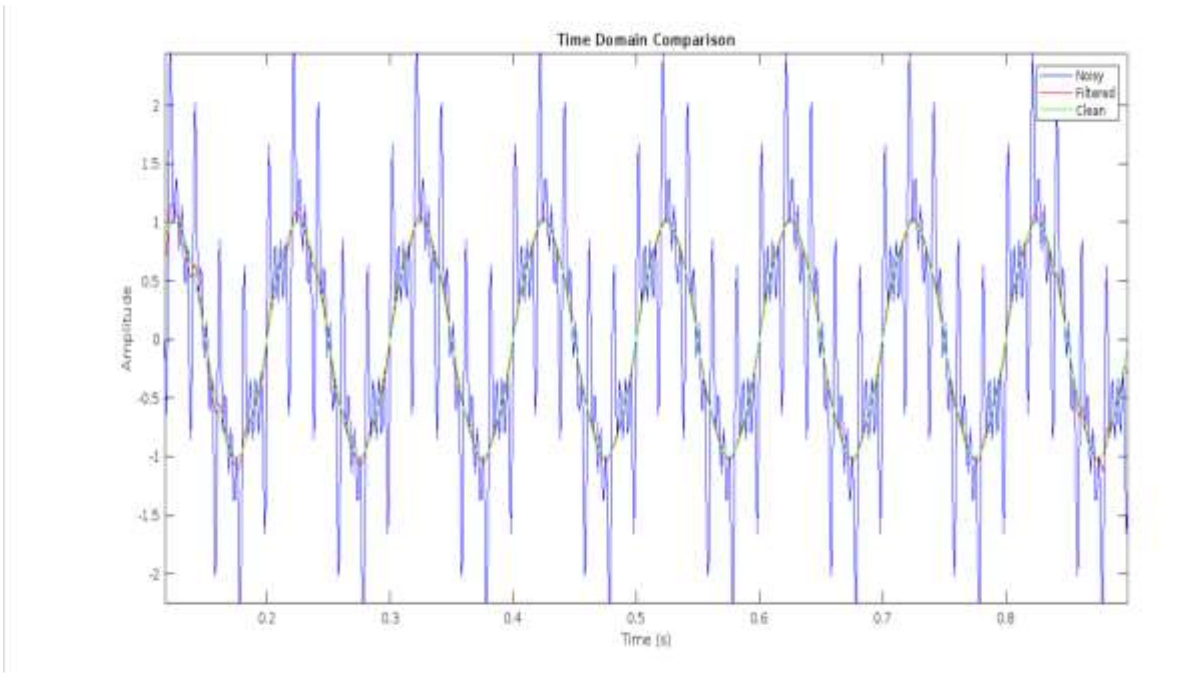


Combined Multi-Notch Filter Response
Magnitude (dB)
Frequency (Hz)
0
-10
-20
-30
-40
-50
-60
0
50
100
150
200
250
300
350
400
450
500

ANSWER -2

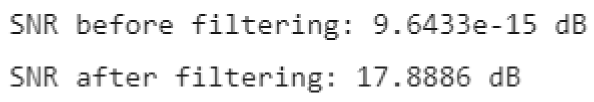


Power Spectral Density
Noisy
Filtered
Clean
Power/Frequency (dB/Hz)
Frequency (Hz)



Time Domain Comparison
Noisy
Filtered
Clean
Amplitude
Time (s)

ANSWER -2



```
SNR before filtering: 9.6433e-15 dB
SNR after filtering: 17.8886 dB
```

Conclusion : By carefully designing the multi-notch filter, visualizing its frequency response, and analyzing the filtering results in both the time and frequency domains, I've developed a robust solution to the power line noise problem. The filter's ability to selectively remove the problematic frequencies while preserving the desired signal characteristics makes it an effective tool for improving the overall quality of our system.