ECE 251 : Signals and Systems Fundamentals

Course Project

Fall 2024

Presented to: Dr Michael Ibrahim and Eng. Yaseen Salah

Presented by:

1- Marwan Ahmed Khairy Ibrahim Elsahy - 22P0201

2- Mohamed Mohsen Mostafa Zaima - 22P0147

3- Omar Ahmed Abdelmohsen Dardir – 22P0218

4- Roger Sherif Selim Salama – 22P0112

5- Saif El Din Wael Mohamed – 22P0191

Acknowledgments:

We would like to express our sincere gratitude to Dr. Michael Ibrahim and Eng. Yaseen Salah for their exceptional guidance and support throughout the ECE 251: Signals and Systems Fundamentals course. Their expertise and dedication played a pivotal role in enhancing our understanding of complex concepts and successfully completing this project. Dr. Michael's ability to present theoretical foundations with clarity and practical relevance has been truly inspiring, while Eng. Yaseen's constructive feedback and hands-on assistance were invaluable during the implementation of our work. Their contributions have not only enriched our knowledge but also motivated us to explore the field of signal processing with greater enthusiasm and depth.

Contribution List

Contributors	Tasks
Marwan Ahmed	Signal Generation and Storage
Mohamed Mohsen	Butterworth High-Pass Filter Design and Implementation
Omar Ahmed	Butterworth Low-Pass Filter Design and Implementation & Report Preparation
Roger Sherif	Frequency Spectrum Analysis & Report Preparation
Saif El Din Wael	Signal Plotting and Analysis

Table of Contents

List of Figures

1.0 INTRODUCTION

This project focuses on analyzing and processing a composite signal x(t)x(t)x(t), composed of four cosine waves with frequencies f1 = 500 Hz, f2 = 1000 Hz, f3 = 1500 Hz, and f2 = 2000 Hz. The main objectives are to generate the signal, compute its energy, analyze its frequency spectrum, and apply both low-pass and high-pass Butterworth filters to observe the impact of frequency-selective filtering on the signal.

The project systematically implements signal processing tasks, starting with signal generation and energy computation in the time and frequency domains. Parseval's theorem is verified to confirm energy consistency across domains. Next, Butterworth filters are designed with a specific cutoff frequency to isolate frequency components and assess the signal's transformation after filtering. The results are visualized through plots of the signal and its frequency spectrum, along with the magnitude and phase responses of the filters.

Additionally, the processed signals are stored as audio files, providing a tangible way to observe the impact of filtering. This approach ensures the practical utility of the project outcomes, particularly for applications involving audio and communication signals. Through a series of systematic steps, the project demonstrates the intricate relationships between time-domain and frequency-domain signal representations and the efficacy of filtering techniques.

2.0 MAIN OBJECTIVES

The primary objectives of this project are as follows:

1. Generate and analyze a composite signal x(t) consisting of multiple cosine components.
2. Compute the energy of the signal in both the time and frequency domains and verify Parseval's theorem.
3. Design and apply Butterworth low-pass and high-pass filters to the signal.
4. Compare the filtered signals to the original signal in both time and frequency domains.
5. Store and document results with plots, audio files, and computations.

3.0 INPUT SIGNAL

3.1 Signal Generation

Initially, we start by initializing the 4 frequencies used for our signals with their respective values, we then proceed by defining our sampling frequency, for this signal we chose to have our sampling frequency at 10 KHz. The sampling frequency 'fs' must be greater than twice the highest frequency 'h' in the signal to satisfy the Nyquist-Shannon Sampling Theorem. This principle ensures that the signal can be accurately reconstructed from its samples without any loss of information, avoiding an issue known as aliasing. In our case, the signal's highest frequency is 2000 Hz, so to satisfy this condition fs > 2fh, fs must be greater than 4 kHz, so we chose 10 kHz as our sampling frequency.

We then defined two different time ranges, one for the signal to perform accurate calculations, so we chose a wide time range, and another for plotting, where we want the plot to be easily visible so a small time range was defined for it. Their respective signals x and x_plot are also declared.

```
% Parameters
f1 = 500;
f2 = 1000;
f3 = 1500;
f4 = 2000;

fs = 10000;  % Sampling frequency
ts = 1/fs;
t = 0:ts:10;  % Time vector
t_plot = 0:ts:0.01;  % Short time vector for plotting
x = cos(2*pi*f1*t) + cos(2*pi*f2*t) + cos(2*pi*f3*t) + cos(2*pi*f4*t);
x_plot = cos(2*pi*f1*t_plot) + cos(2*pi*f2*t_plot) + cos(2*pi*f3*t_plot) + cos(2*pi*f4*t_plot);
```

Figure 1 Declaration of variables

The next sequence of steps included saving the signal into an audio file, the signal was normalized by dividing by 4 and remultiplying to prevent clipping.

The signal is then plotted for visualization across time. The t_plot and x_plot variables were used to enhance readability of the graph.

```
% Step 2: Save the signal to a WAV file
filename = 'signal_1.wav';
x = x / 4;  % Normalize to prevent clipping
audiowrite(filename, x, fs);  % Save the normalized signal
%sound(x, fs);

x = x * 4;  % Restore original amplitude

% Step 3: plot the signal
figure;
plot(t_plot, x_plot);
title('Generated Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Step 4: Compute the energy of the signal in the time domain
energy_time = sum(x.^2) / length(x);

```

Figure 2 Audio file and signal plotting

3.2 Input Signal Energy Calculation in time domain

The next step calculates the energy of the signal in the time domain by squaring each of the discrete values of the signal 'x' and summing them.

Since x(t) is a periodic signal with an infinite duration in theory, the computed energy without normalization would grow with the signal's length. Dividing by the number of samples normalizes the energy, making it independent of the signal's length. Which is why we divide by length(x), which is the number of samples stored in array x.

3.3 Signal Visualization

The following image shows the signal plotted across time.

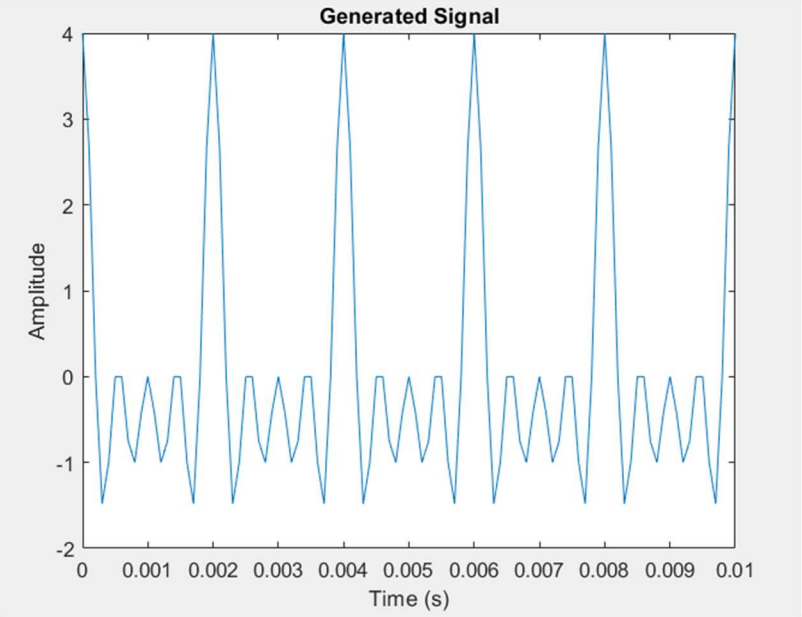

Figure 3 Signal in time domain

3.4 Input Signal Frequency Spectrum Analysis

We start here by finding the fundamental frequency which is used to extract exactly one period of the signal, which is needed for the next steps.
The FFT function is then used to transform the signal into its frequency domain, note that only one single period is used to compute the frequency domain, hence, x_periodic is used.
FFTShift function is used to shift the frequencies to be centered at the DC component at 0 Hz.
The frequency spectrum is then plotted.

```
%Extract one fundamental period of the signal
gcd_freq = gcd(gcd(f1, f2), gcd(f3, f4));  % GCD of frequencies
T_fundamental = 1 / gcd_freq;  % Fundamental period
N0 = round(T_fundamental * fs);  % Number of samples in one period
x_periodic = x(1:N0);  % Extract one period

% Step 5: Compute the frequency spectrum using the FFT
X_f = fft(x_periodic);

% Frequency vector
f = (-fs/2):(fs/length(x_periodic)):(fs/2 - fs/length(x_periodic));

% Shift the spectrum to center the zero frequency component
X_f_shifted = fftshift(X_f) / length(x_periodic);

% Step 6: Plot the magnitude of the frequency spectrum
figure;
stem(f, abs(X_f_shifted), 'marker', 'none');
title('Magnitude of Frequency Spectrum X(f)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

Figure 4 Finding the frequency spectrum of the signal

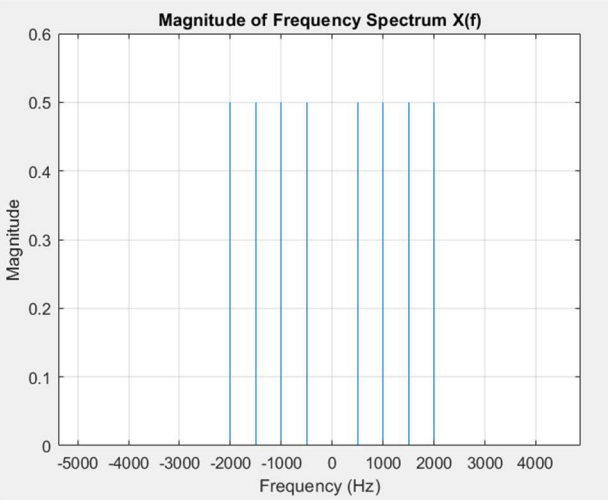

Figure 5 Signal X(f) in frequency domain

Figure 2 shows the frequency spectrum of the signal x(f). From our observations, we can deduce that the set of frequencies on the graph are indeed the frequencies defined in our signal (500, 1000, 1500, 2000 Hz).

We can also verify the magnitude of each frequency where the coefficients of the cosine functions are all 1, when plotted, we can see magnitude of 0.5 in each frequency in both negative and positive frequencies, summed it would give us a magnitude of 1 for each frequency.

3.5 Input Signal Energy Calculation in Frequency Domain

Step 7 includes calculating the energy from the frequency spectrum and also verifying Parseval's theorem which states that the energy must be equal when calculated from the time and frequency domains. Normalization by dividing the energy by the number of samples was not needed here since the shifted X in frequency (X_f_shifted) is already defined for exactly one period.

```
% Step 7:
% Compute the energy in the frequency domain
energy_freq = sum(abs(X_f_shifted).^2);

% Verify Parseval's theorem
fprintf('X Energy in Time Domain: %.5f\n', energy_time);
fprintf('X Energy in Frequency Domain: %.5f\n', energy_freq);

if abs(energy_time - energy_freq) < 0.001
    disp('Parseval's theorem is verified!');
else
    disp('Parseval's theorem is NOT verified.');
end
```

Figure 6 Verification of Parseval's theorem

3.6 Verifying Parseval's Theorem Using The Input Signal

```
X Energy in Time Domain: 2.00014
X Energy in Frequency Domain: 2.00000
Parseval's theorem is verified!
```

Figure 7 Energy calculated in time and frequency domains

The energy from both the time and frequency domains can be verified to be equal which proves Parseval's theorem. A condition was added for potential errors to adjust for slight miscalculations.

4.0 LOW-PASS FILTER DESIGN, IMPLEMENTATION AND VERIFICATION.

4.1 Designing a Butterworth Low-Pass Filter

Designing a Butterworth low-pass filter was relatively easy. We use the built in Butter function which takes the order and the normalized cut-off frequency. The frequency is normalized with respect to half the sampling frequency because lower than that and the filter will allow too much of the signal through reducing its effect. We also specified in the function that we are designing a low pass filter.

The butter function outputs an array of filter coefficients (B,A). The B is the numerator and A is the denomerator.

```
% Step 8: Designing a butterworth lpf:
fc = 1250 %cut-off freq
Wn = fc / (fs / 2);
% normalize cut-off w.r.t to half samlpling freq
order = 20;
%here I design the filter
[B, A] = butter(order, Wn, 'low');
```

Figure 8 Designing a butterworth low pass filter (LFT)

To verify that the filter we've created is functioning correctly we used another function called freqz which is used to get the frequency response of filters on matlab.

Along with the frequency coefficients, we pass to the function the sampling frequency and the number of frequency points which specifies how many points are used to calculate the frequency response. We used a reasonable number of 4000 to ensure the graph is smooth enough.

```
81    % Frequency response
82    [H, w] = freqz(B, A, 4000, fs);
83
84    % Step 9:
85    % Magnitude plot
86    figure;
87    subplot(2, 1, 1);
88    plot(w, abs(H));  % Plot the magnitude of H
89    title('y1 Magnitude Response of the Filter');
90    xlabel('Frequency (Hz)');
91    ylabel('Magnitude');
92    grid on;
93
94    % Phase plot without unwrapping to see the raw phase
95    subplot(2, 1, 2);
96    plot(w, unwrap(angle(H)));  % Plot the phase of H (in radians)
97    title('y1 Phase Response of the Filter');
98    xlabel('Frequency (Hz)');
99    ylabel('Phase (radians)');
100   grid on;
```

Figure 9 Plotting the LPF

We extracted the magnitude and phase of the frequency response and plotted them.

Here's the graphs that we got:

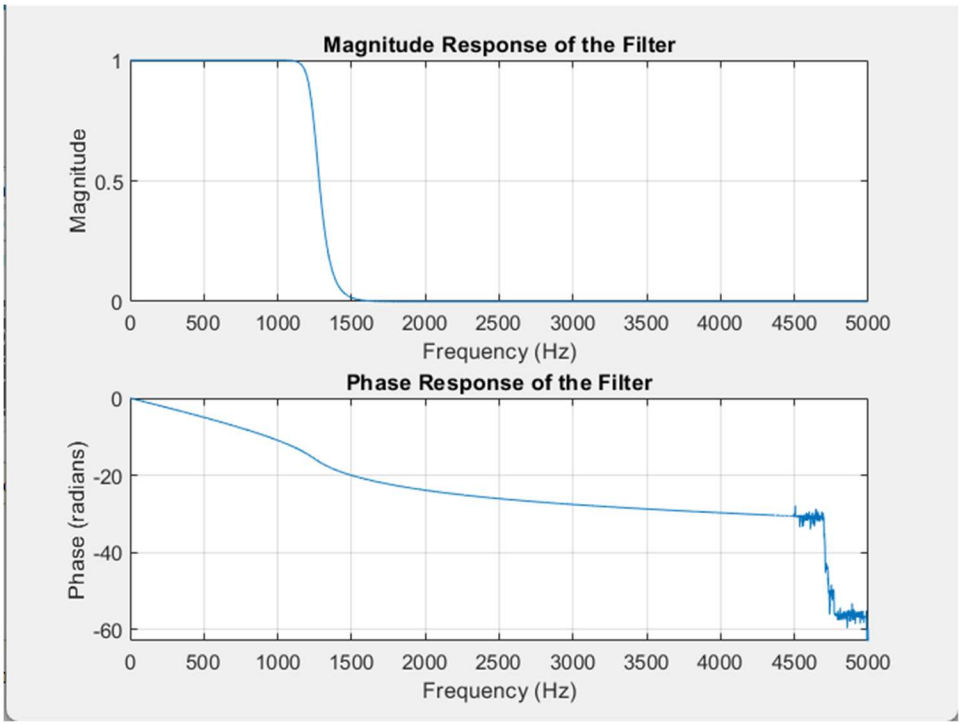

Figure 10 Magnitude and Phase responses of LPF

The graph of the magnitude of the frequency response proves that the order is high because of the steepness and sharp drop right before the cutoff frequency.

Now that we've verified that the filter is working as expected, we pass our original signal through the filter and observe the output y1.

```
% Step 10:
y = filter(B, A, x);
y_plot = filter(B, A, x_plot);

% Step 11: Save the signal to a WAV file
filename = 'signal_2.wav';
y = y / 4;  % Normalize to prevent clipping
audiowrite(filename, y, fs);  % Save without normalization to preserve amplitude
%sound(y, fs);
```

Figure 11 Applying the filter on our signal and storing audio file

4.2 Y1 Signal Creation

We used the filter function applies the digital filter onto the signal. We pass the original signal and the filter coefficients and store the output y. We also did the same thing using x_plot and y_plot to show the appropriate range during plotting.

4.3 Y1 Storage in Audio File and Plotting

We also saved the output as a sound file using audiowrite function again.

Here's the plotted signal before and after passing it through the digital filter:

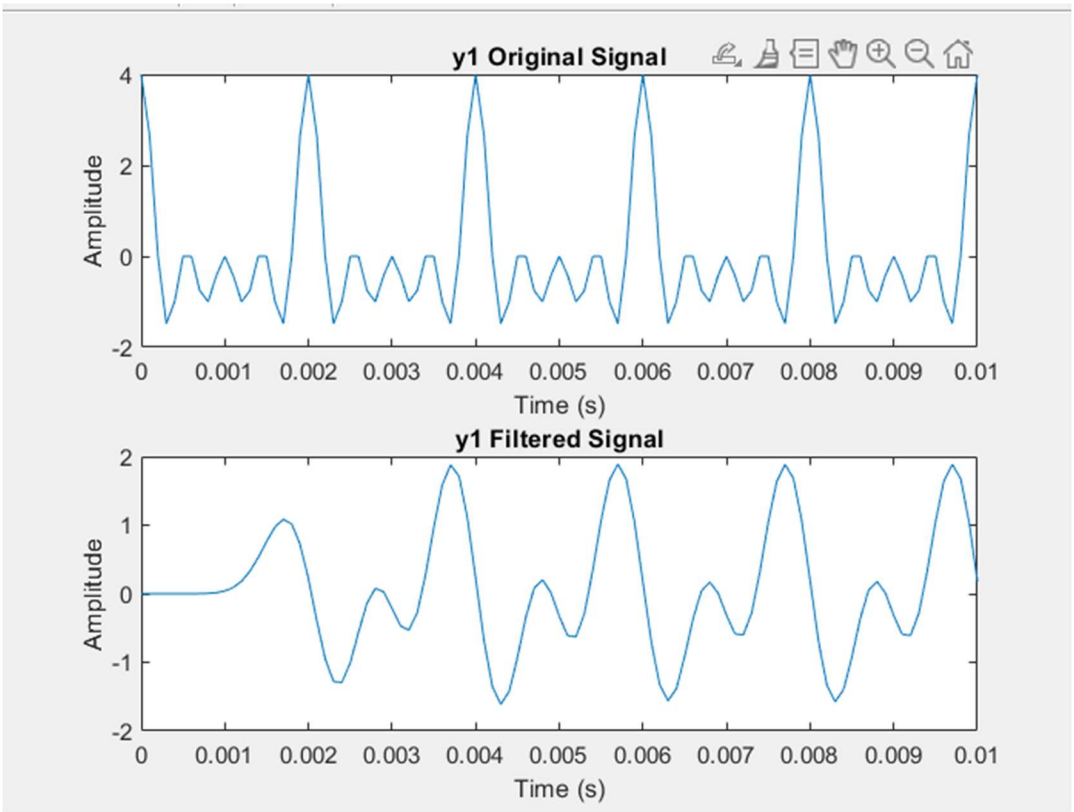

Figure 12 Signal before and after being passed to the LPF

We can notice from the graph that the output signal is different, but they have the same fundamental period. More importantly we see that there is a delay in the signal starting from time zero as it has not yet stabilized. This delay increases as the order of the Butterworth low pass filter increases.

The energy was calculated and noted down again to be used later on.

4.4 Y1 Fourrier Transform and Frequency Spectrum Analysis

Using FFT to show this signal is in the frequency domain caused a lot of unexpected results at the beginning as this delay produced a DC component in the frequency domain so we had to take another period later on in the signal when It has stabilized where we can clearly see the filtered frequencies.

Here's the code:

```
133    %Fundamental Period Calculation
134    gcd_freq = gcd(gcd(f1, f2), gcd(f3, f4));  % GCD of frequencies
135    T_fundamental = 1 / gcd_freq;  % Fundamental period
136    N0 = round(T_fundamental *fs);  % Number of samples in one period
137    y_periodic = y(10*N0:11*N0-1);  % Extract one period
138    %======================
139
140    % Step 14: Compute the frequency spectrum using the FFT
141    Y_f = fft(y_periodic);
142
143    % Frequency vector
144    f = (-fs/2):(fs/length(y_periodic)):(fs/2 - fs/length(y_periodic));
145
146    % Shift the spectrum to center the zero frequency component
147    Y_f_shifted = fftshift(Y_f)/length(y_periodic);
148
```

Figure 13 Converting the low pass filtered signal to the frequency domain

Here's the output:

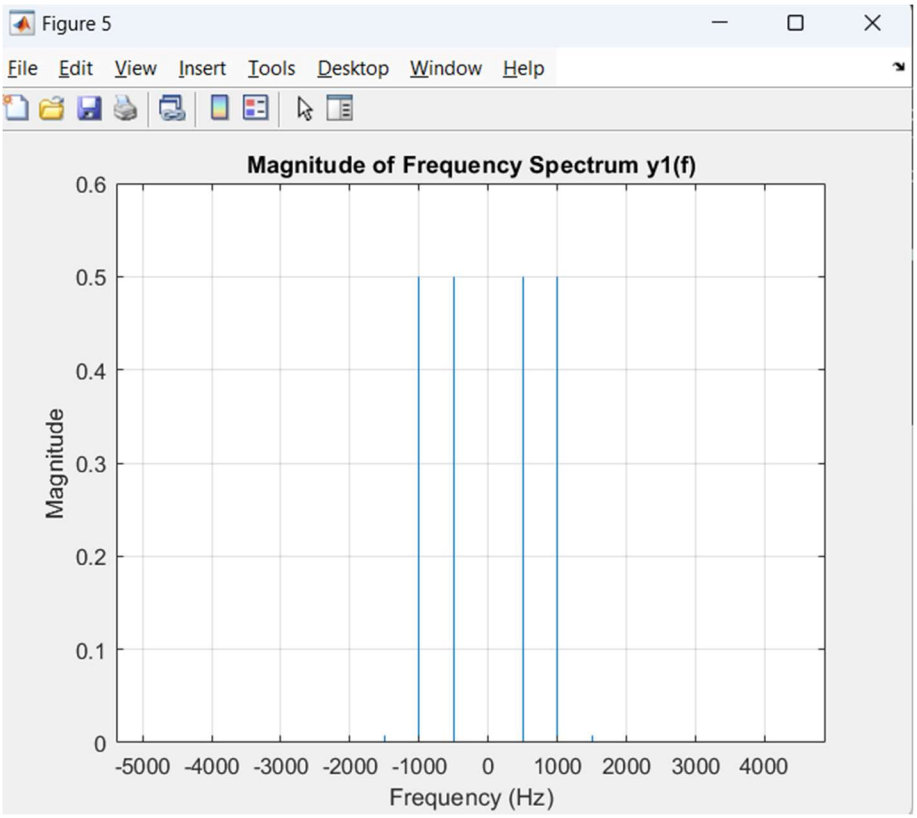

Figure 14 Frequency spectrum of low pass filtered signal

The magnitude at frequencies lower than the cutoff frequency is the same as the original signal and the frequencies above 1250 Hz are filtered. This shows that the filter is working perfectly.

4.5 Y1 Verifying Parseval's Theorem

After computing the energy using the frequency domain we could prove parseval's theorem again.

```
Y1 Energy in Time Domain: 0.99991
Y1 Energy in Frequency Domain: 1.00010
Parseval's theorem is verified!
```

Figure 15 Energy in time and frequency domains for the filtered signal

5.0 BUTTERWORTH HIGH PASS FILTER DESIGN, IMPLEMENTATION AND VERIFICATION

5.1 Butterworth High Pass Filder design

The next step was to design a Butterworth high pass digital filter, the same exact functions were used to design the filter and plot the frequency response. The only difference is we specified to the butter function that the filter is "high" meaning high pass.

Here's where the mentioned difference occurs:

```
% Step 17:Designing a butterworth :
fc = 1250 %cut-off freq
Wn = fc / (fs / 2);
% normalize cut-off w.r.t to half samlpling freq
order = 20;
%here I design the filter
[B1, A1] = butter(order, Wn, 'high');
```

Figure 16 Designing a butterworth high pass filter (HPF)

The magnitude and phase of the frequency response of the new filter were plotted.

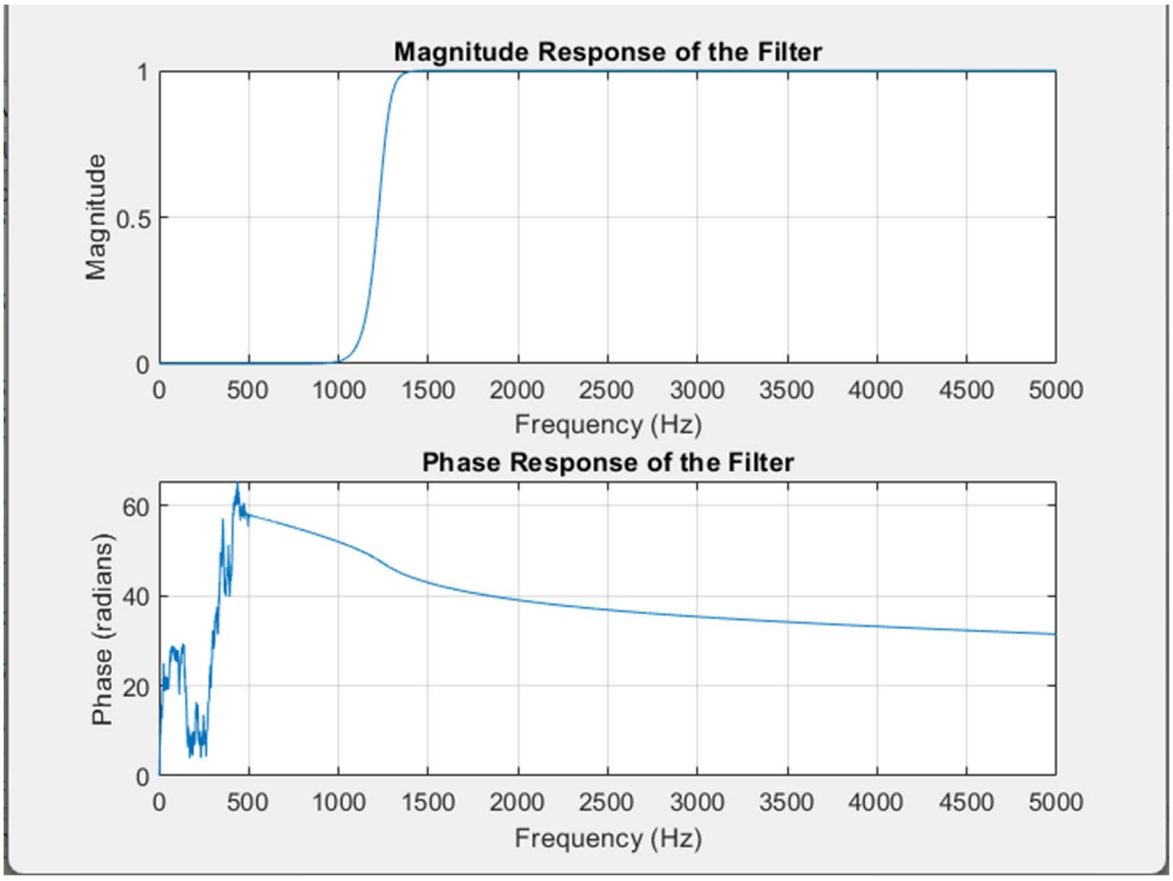

Figure 17 Magnitude and Phase responses of HPF

The functionality of the digital filter is verified from the magnitude graph as it is 0 before the cutoff frequency and right as it approaches the cutoff frequency it rises to 1 with a steep curve.

5.2 Y2 signal Creation and Plotting

The original signal was passed through the new filter and once again there are expectedly unexpected spikes at the start indicating that the signal hasn't stabilized yet.

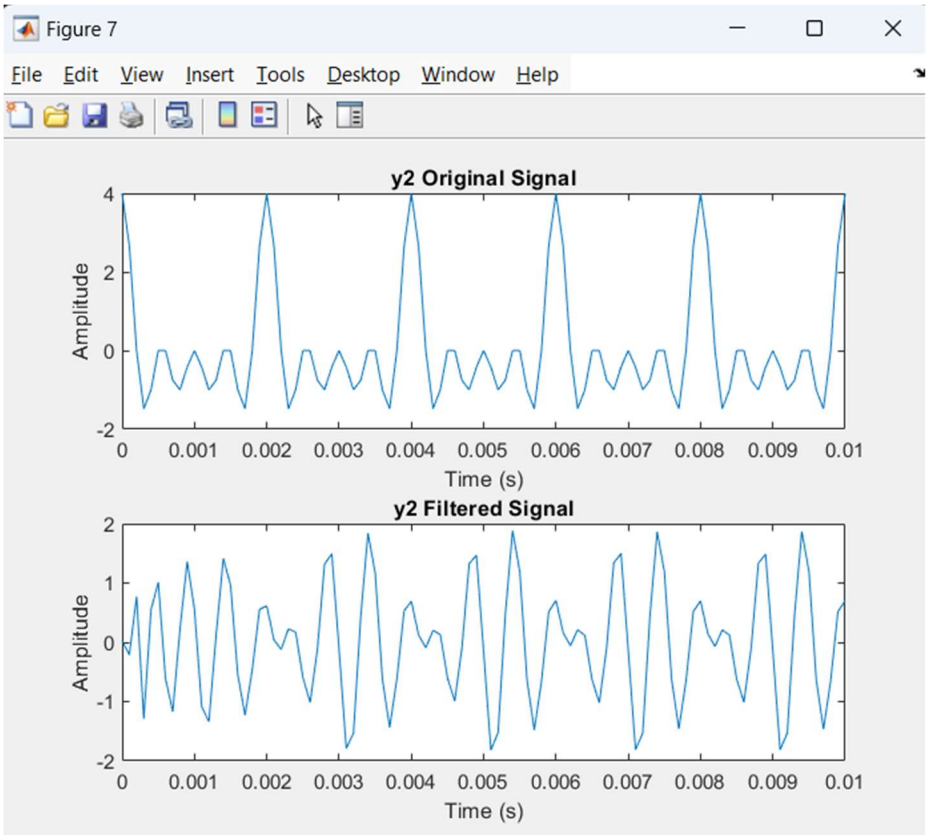

Figure 18 Signal before and after being passed to the HPF

5.3 Y2 Signal Energy Calculation and Storage in Audio File

The energy of the output signal is calculated and noted to once again prove parseval's theorem.

The signal y2 is saved in an audio file just like the other signals.

5.4 Y2 Signal Frequency Spectrum Analysis

Y2 is transformed to the frequency domain once again using FFT. The period was chosen was after time zero to avoid the instability at the beginning and the results are as shown below.

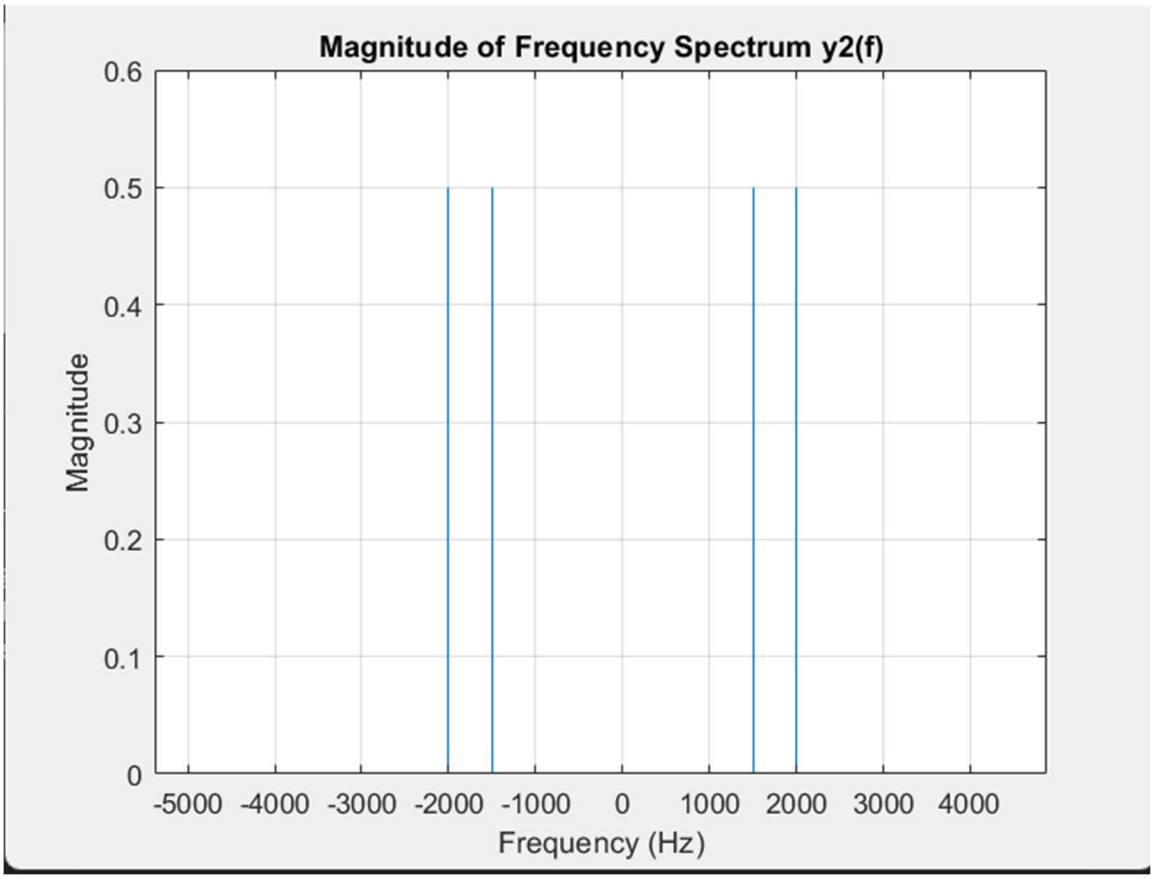

Figure 19 Frequency spectrum of high pass filtered signal

This graph was just as expected as low frequences below the cut-off frequency were filtered out and higher frequencies pass through with the same magnitude.

5.5 Y2 Verifying Parseval's Theorem

The last step is to calculate the energy of this signal using the frequency domain and once again prove Parseval's theorem and it was proved.

```
Y2 Energy in Time Domain: 0.99985
Y2 Energy in Frequency Domain: 0.99990
Parseval's theorem is verified!
```

Figure 20 Energy in time and frequency domains after signal being passed to HPF

6.0 Conclusion

The project successfully achieved its objectives, providing a thorough exploration of signal analysis and filtering techniques using a composite signal. The generation of x(t) and its energy computation in both time and frequency domains highlighted the consistency of energy representation, as verified through Parseval's theorem. This validation reinforced the theoretical principles underlying the analysis.

The application of Butterworth low-pass and high-pass filters demonstrated their ability to isolate specific frequency components effectively. By designing filters with a cutoff frequency of 1.25 kHz and visualizing their magnitude and phase responses, the project illustrated the influence of frequency-selective filtering on signal characteristics. The filtered signals, both in time and frequency domains, showcased clear distinctions, affirming the precision of the filtering process.

Storing the signals as audio files further emphasized the practical implications of the project, enabling an auditory comparison of the original and filtered signals. Overall, the project provided deep insights into the interplay between time-domain signals, frequency spectra, and filtering, offering valuable knowledge for real-world signal processing applications.

7.0 References

- S. Haykin and B. Van Veen, *Signals and Systems*, 2nd ed. Hoboken, NJ, USA: Wiley, 2003.
- MATLAB Documentation, MathWorks, Natick, MA, USA. [Online]. Available: https://www.mathworks.com/help/matlab
- "Fourier Transform in MATLAB," GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/fourier-transform-in-matlab/. [Accessed: Dec. 27, 2024].