Mechatronic Systems Identification

Lab 1 - Basics of Signal Processing

Khaldoun Fayad - 409597

Witold Surdej - 407100

09.03.2024

Description

The aim of this laboratory is for us to learn to learn the basics of signals and their main types, modification methods, and properties.

Task 1

Our task involved creating a signal by combining two sine waves, then analysing its frequency content through a Fourier transform using the FFT function. We adjusted the amplitude scaling in the Fourier transform to ensure the plot accurately represented the sine waves' amplitudes. Additionally, we aimed to identify the minimal signal length required at the same sampling frequency to distinguish both frequencies in the spectrum clearly, while preserving their correct amplitudes.

```
%% Task 1
fs = 500; % sampling frequency
N = 2000; % number of samples
t = (0:N-1)/fs; % time range
f1 = 30; %frequency one
f2 = 30.5; %frequency two
a1 = 1; % amplitude one
a2 = 2; % amplitude two
y = a1*sin(2*pi*f1*t) +
a2*sin(2*pi*f2*t);
figure(1)
plot(t,y)
xlabel('time (s)')
ylabel('amplitude')
```

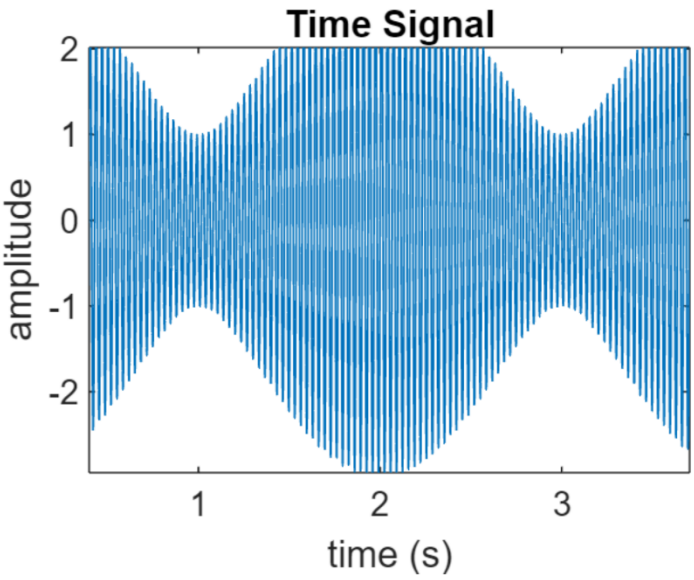

Figure 1 Graph of the sine wave.

```
% Creating the fourier transform
Y = fft(y);
N = length(Y);
df = fs/N; % frequency
resolution
fv = (0:N-1)*df; % frequency
vector
Nf = (abs(Y)/N*2);
figure(2)
stem(fv , abs(Y))
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([29,32])
```

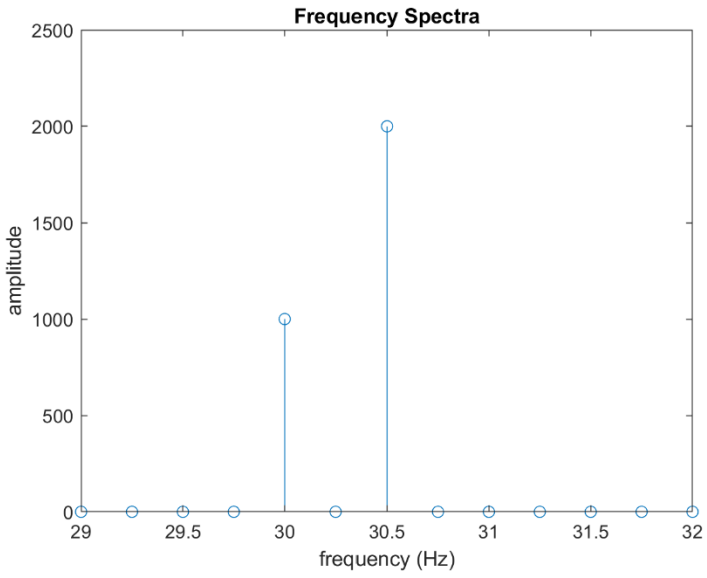

Figure 2 Graph of the frequency spectra.

Comments:

We first attempted to graph the continuous time signal and its frequency spectra using 500 samples, however, we noticed that we could not obtain the proper scale in the frequency domain and in turn the wave's frequencies were approximated. This mean that there is spectral leakage present in the system and an increase in the number of samples could fix it. Therefore, we started increasing the number of samples in order to obtain the proper frequency scale and managed to obtain the correct frequencies after changing the number of samples to 2000 samples with the same sampling frequency as shown in Figure 2. The increase of the number of samples meant an increase in the spectral resolution of the wave. The spectral resolution is the sampling rate at which the sensor (in case of a physical system) collects the information about the observed system. It is expressed by the following equation: dF=fs/N, where fs is sampling frequency and N is number of samples. As a result, the spectral resolution increases whenever dF decreases (ie. when the number of samples increases).

We also noticed in the time wave graph shown in Figure 1 a phenomenon wanes called Beats. The beating phenomenon in signal processing occurs when signals with frequencies that are close but not identical are combined, they produce a resultant waveform with a slowly varying amplitude (known as beating). The beat frequency is equal to the difference between the frequencies of the two original signals. Beating is commonly encountered in audio signals. It results in a distinct rise and fall in intensity, creating a noticeable pulsating effect. In our example due to the frequencies being close to each other the phenomenon leads to constructive and destructive interference patterns. By analysing beat frequencies, engineers can extract information about the characteristics of the original signals.

Task 2

Our task cantered on creating a signal from the sum of two cosine waves, calculating its Fourier transform, and analysing the amplitude-frequency spectrum. We further modified the signal by subsampling, retaining alternate samples, and conducted a similar Fourier analysis on this subsampled version to observe the effects on its spectrum.
Additionally, we explored variations of the original signal composition by using sine waves in one case and the difference of cosine waves in another, without specifying their frequencies or amplitudes. The analysis compared the amplitude and the real and imaginary components across all scenarios, aiming to understand how subsampling and changes in signal composition affect the spectral properties. This approach provided insights into the principles of signal processing and the impact of waveform combinations on a signal's spectral representation.

```
%% Task 2
fs = 1000; % sampling frequency
N = 1000; % number of samples
t = (0:N-1)/fs; % time range
f1 = 125; %frequency one
f2 = 375; %frequency two
y = cos(2*pi*f1*t) + cos(2*pi*f2*t);
figure(1)
plot(t,y)
title('Time Signal')
xlabel('time (s)')
```

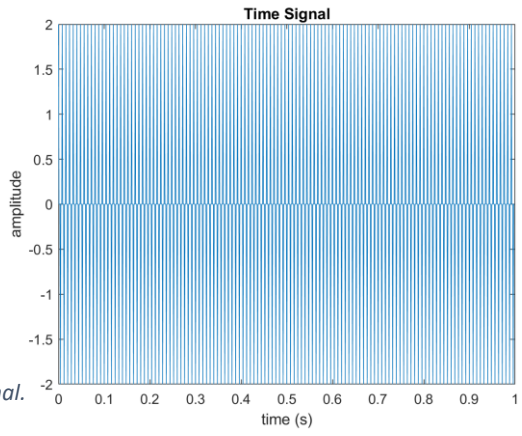

Figure 3 Time wave of first signal.

```
ylabel('amplitude')

% Creating the fourier transform
Y = fft(y);
N = length(Y);
df = fs/N; % frequency resolution
fv = (0:N-1)*df; % frequency
vector
Nf = (abs(Y)/N*2);
figure(2)
stem(fv , abs(Y))
title('Frequency Spectrum')
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([0,500])

% Subsampled Signal
Y1 = (y(1:2:end));
% Creating the fourier transform
Y = fft(Y1);
N = length(Y);
df = fs/N; % frequency resolution
fv = (0:N-1)*df; % frequency vector
Nf = (abs(Y)/N*2);
figure(2)
stem(fv , abs(Y))
title('Frequency Spectrum of
subsampled signal')
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([0,500])
```

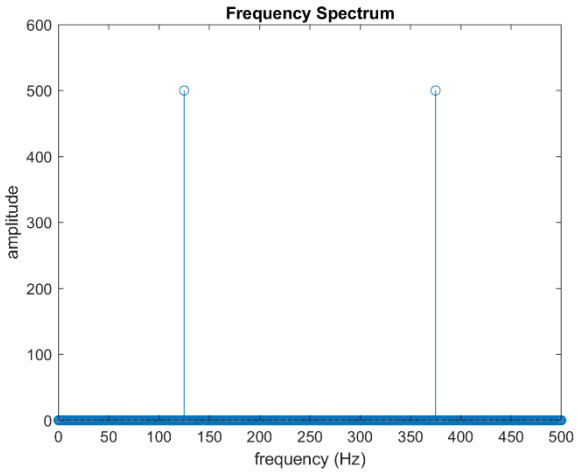

Figure 4 Frequency spectrum of the first signal

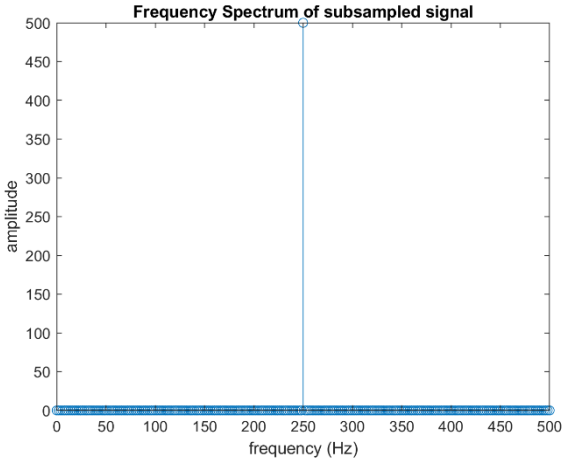

Figure 5 Frequency spectrum of subsampled signal.

```
% Part a
% First Signal
fs = 1000; % sampling frequency
N = 1000; % number of samples
t = (0:N-1)/fs; % time range
f1 = 125; %frequency one
f2 = 375; %frequency two
y1 = sin(2*pi*f1*t) +
sin(2*pi*f2*t);
figure(1)
plot(t,y1)
title('Time Signal')
xlabel('time (s)')
ylabel('amplitude')
```

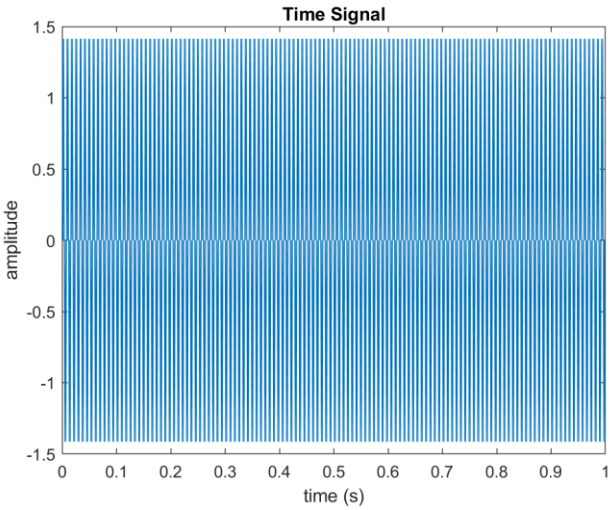

Figure 6 Time wave of second signal.

```
% Creating the fourier transform
Y = fft(y);
N = length(Y);
df = fs/N; % frequency resolution
fv = (0:N-1)*df; % frequency vector
Nf = (abs(Y)/N*2);
figure(2)
stem(fv , abs(Y))
title('Frequency Spectrum')
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([0,500])
```

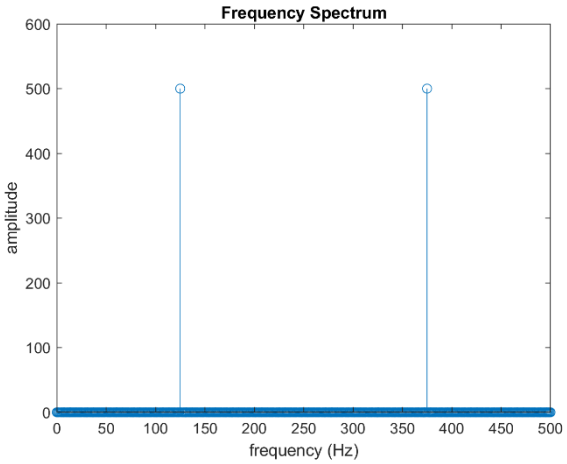

Figure 7 Frequency Spectrum of second signal.

```
% Subsampled Signal
Y1 = (y(1:2:end));
% Creating the fourier transform
Y = fft(Y1);
N = length(Y);
df = fs/N; % frequency resolution
fv = (0:N-1)*df; % frequency vector
Nf = (abs(Y)/N*2);
figure(2)
stem(fv , abs(Y))
title('Frequency Spectrum of
subsampled signal')
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([0,500])
```

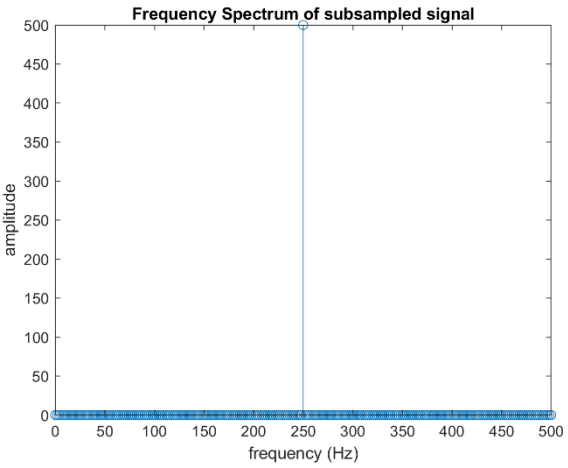

Figure 8 Frequency spectrum of second subsampled signal.

```
% Part b
% First Signal
fs = 1000; % sampling frequency
N = 1000; % number of samples
t = (0:N-1)/fs; % time range
f1 = 125; %frequency one
f2 = 375; %frequency two
y = cos(2*pi*f1*t) -
cos(2*pi*f2*t);
figure(1)
plot(t,y)
title('Time Signal')
xlabel('time (s)')
ylabel('amplitude')
```

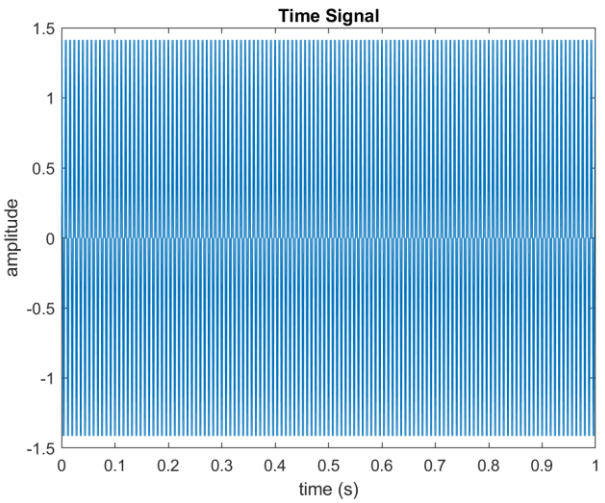

Figure 9 Time wave of third signal.

```
% Creating the fourier transform
Y = fft(y);
N = length(Y);
df = fs/N; % frequency resolution
fv = (0:N-1)*df; % frequency vector
Nf = (abs(Y)/N*2);
figure(2)
stem(fv , abs(Y))
title('Frequency Spectrum')
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([0,500])

% Subsampled Signal
Y1 = (y(1:2:end));
% Creating the fourier transform
Y = fft(Y1);
N = length(Y);
df = fs/N; % frequency resolution
fv = (0:N-1)*df; % frequency vector
Nf = (abs(Y)/N*2);
figure(2)
stem(fv , abs(Y))
title('Frequency Spectrum of
subsampled signal')
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([0,500])
```

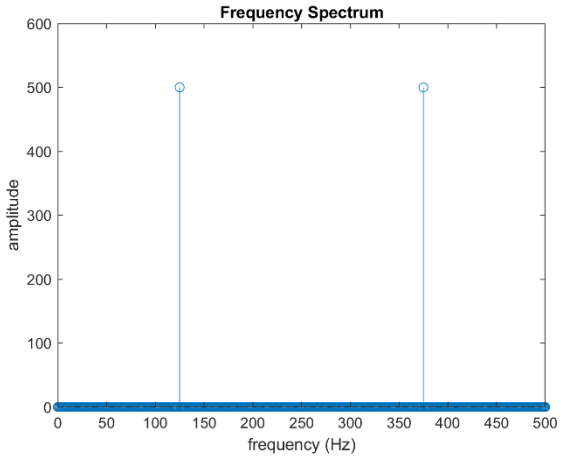

Figure 10 Frequency spectrum of the third signal.

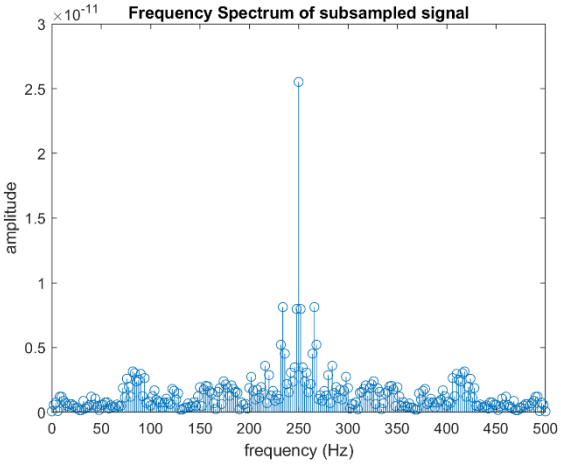

Figure 11 Frequency spectrum of third subsampled signal.

Comments:

When we first attempted to graph the signal and sampled, we noticed that in all three cases we have obtained the frequencies of the sum of the continuous time waves properly in the frequency spectrum, as shown in Figure 4, Figure 7, and Figure 10. However, in the first and second cases, where our time wave signal was the result of a sum of two signals, the subsampling operation that was done resulted in a mirroring operation and the frequencies added up to each other as shown in Figure 5, and Figure 8. Furthermore, for the third time wave signal, which was the result of the difference of two cosine signals, it appeared that the result was significantly lower in terms of amplitude, but the process of frequencies adding up was also witnessed in this case as well as shown Figure 11.

This clearly depicts the effect of insufficient sampling on properly displaying signal in the frequency domain, when omitting the proper guidelines and rules of sampling signals, the result could be completely unreliable and incorrect visualizations of the real signals would appear.

Task 3

Our task involved generating a signal over a one-second interval with a specified sampling rate, resulting in a set number of samples. This signal, labelled y1, was the sum of a sine wave and a cosine wave, each with distinct frequencies and amplitudes. We computed its Fourier transform and presented the spectrum on a logarithmic scale, adjusting the frequency range to span from zero to the Nyquist limit.

Following this, we applied a Hann window to the original signal in the time domain, creating a new signal y2. The Fourier transform of this windowed signal was then calculated, and its spectrum was displayed alongside the original signal's spectrum for comparison. This comparison aimed to identify differences in the spectra of the y1 and y2 signals, particularly focusing on the effects of windowing.

Additionally, we explored the necessity of windowing for the y1 signal and discussed alternative window functions that could be applied. A third signal, y3, was created by applying a different window to the original signal y1. The spectrum of this newly windowed signal was plotted and analysed to assess the properties and impact of the chosen window.

```
%% Task 3
fs = 100; % sampling frequency
N = 101; % number of samples
t = (0:N-1)/fs; % time range
f1 = 7; %frequency one
f2 = 20; %frequency two
a1 = 10; % amplitude one
a2 = 0.02; % amplitude two
y1 = a1*sin(2*pi*f1*t) +
a2*sin(2*pi*f2*t);
figure(1)
plot(t,y1)
xlabel('time (s)')
ylabel('amplitude')
```

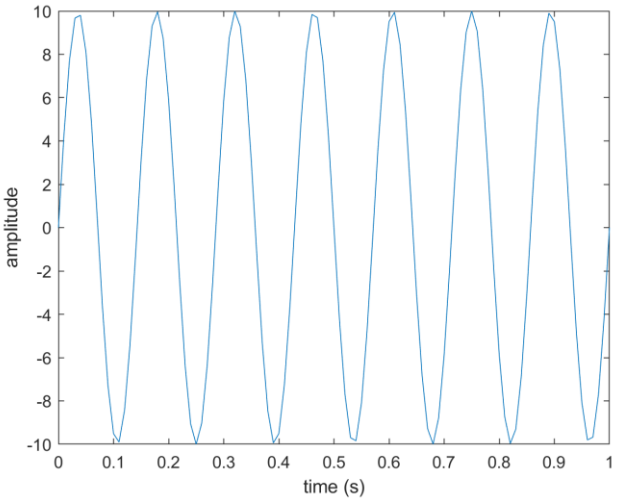

Figure 12 Graph of the time signal.

```
% Creating the fourier transform
Y = fft(y1);
N = length(Y);
df = fs/N; % frequency
resolution
fv = (0:N-1)*df; % frequency
vector
Nf = (abs(Y)/N*2);
figure(2)
stem(fv , abs(Y))
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([0,50])
figure(3)
semilogy(fv , abs(Y))
xlabel('frequency (Hz)')
ylabel('magnitude [db]')
xlim([0,50])
```

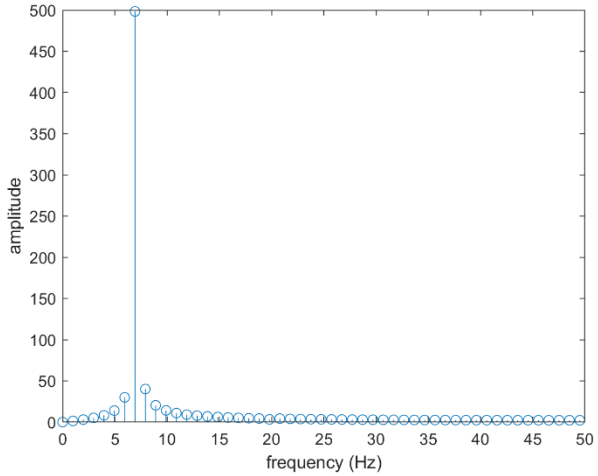

Figure 13 Graph of the frequency spectrum.

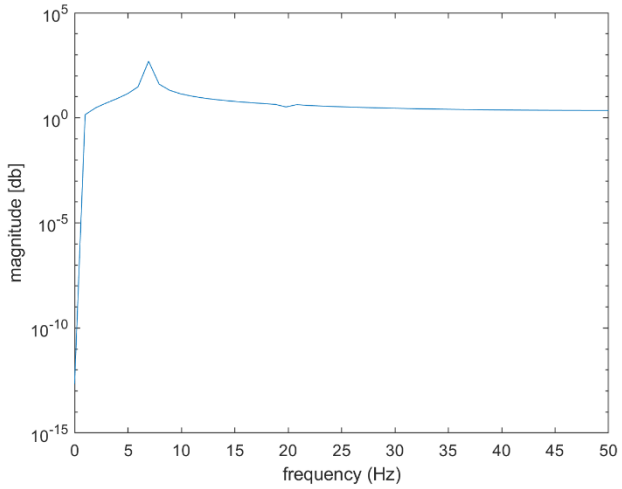

Figure 14 Logarithmic plot of the frequency spectrum.

```
% Creating the Hann windowed signal
y2 = y1.*hann(N)';
figure(4)
plot(t,y2)
xlabel('time (s)')
ylabel('amplitude')
```

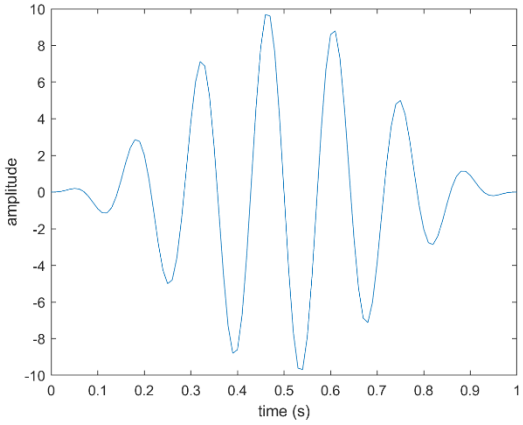

Figure 15 Graph of the Hann Windowed signal.

```
% Creating the Gaussian windowed
signal
y3 = y1.*gausswin(N)';
figure(5)
plot(t,y3)
xlabel('time (s)')
ylabel('amplitude')
```

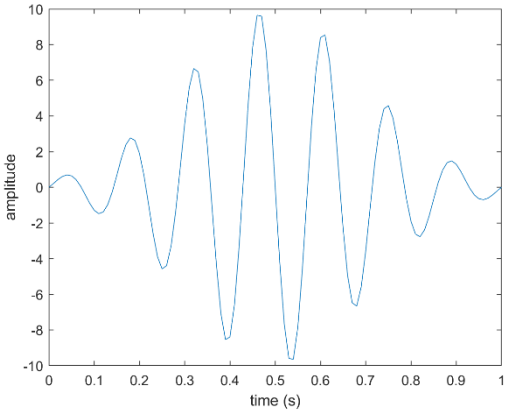

Figure 16 Graph of the Gaussian Windowed signal.

```
% Creating the fourier transform of the Hann Signal
Y2 = fft(y2);
N2 = length(Y2);
df2 = fs/N2; % frequency resolution
fv2 = (0:N2-1)*df2; % frequency vector
Nf = (abs(Y)/N*2);
```

```
% Creating the fourier transform of the Gaussian Signal
Y3 = fft(y3);
N3 = length(Y3);
df3 = fs/N3; % frequency resolution
fv3 = (0:N3-1)*df3; % frequency vector
Nf = (abs(Y)/N*2);

% Plotting the three signals
figure(6)
semilogy(fv2 , abs(Y2))
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([0,50])
hold on
figure (7)
semilogy(fv , abs(Y))
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([0,50])
figure(8)
semilogy(fv3 , abs(Y3))
xlabel('frequency (Hz)')
ylabel('amplitude')
xlim([0,50])
hold off
legend ('Hann Windowed Signal','Original Signal','Gaussian Widnowed Signal')
```

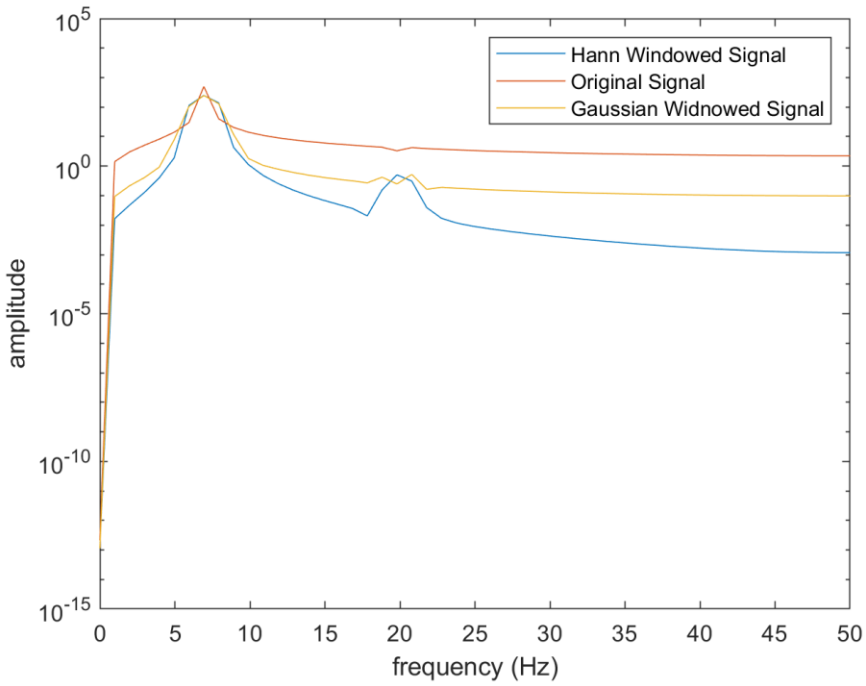

Figure 17 Graph of the logarithmic plot of the original signal and the two windowed signals.

Comments:

When we first attempted to graph the signal in the frequency domain using both the amplitude and logarithmic spectrum, as shown in Figure 13 and Figure 14, we noticed that when it comes to the low amplitude signal; its frequency is barely visible and we were not able to clearly identify whether such frequency is actually present or not in the signal. This is because of the spectral leakage that is present in the signal. However, the presence of windows allows us to fix that.

Window functions in signal processing are mathematical functions used to modify a signal by tapering its beginning and end to zero. This technique is applied to a finite segment of the signal to reduce edge effects or spectral leakage when performing a Fourier Transform or other spectral analysis. Essentially, windowing involves multiplying the signal by a window function, which is typically one in the middle and gradually decreases to zero towards the ends. The main purpose of using window functions is to mitigate the discontinuities at the boundaries of a signal segment. When a signal is abruptly cut off at the beginning and end of a segment, it introduces artificial discontinuities, as the Fourier Transform assumes the signal is periodic and repeats indefinitely. This can lead to spectral leakage, where energy from the signal's true frequencies spreads into adjacent frequencies, obscuring the true spectral content.

In our case, we multiplied the signal by two types of windows, the Hann window, and the Gaussian window. These windows were able to reduce the spectral leakage of the signal and were able clearly see the present frequencies in the signal. However, as it is visible in Figure 17, not all window functions are suitable for all waves. The Hann window was able to show the frequencies more than the Gaussian window and this is because of the window function equations and how they are able to modify in the signal. When faces with such issue, one has to clearly try to identify the signal that is most suited for the operation at hand.