EEE3030 Signal Processing and Machine Learning

Semester 1 Report

Sahas Talasila *230057896*

Contents

1 Plan for Report **2**
- 1.1 Task 1 Plan . . . 2
- 1.2 Task 2 Plan . . . 2
- 1.3 Task 3 Plan . . . 3
- 1.4 Task 4 Plan . . . 3
- 1.5 Task 5 Plan . . . 3

2 Introduction **4**

3 Results and Discussion **4**
- 3.1 Task 1 Procedure, Results and Discussion . . . 4
  - 3.1.1 Reading the Audio File and Statistics . . . 4
  - 3.1.2 Time Domain Analysis . . . 5
  - 3.1.3 Frequency Domain Analysis and FFT . . . 6
  - 3.1.4 Windowing Methods and Their Effects . . . 7
- 3.2 Task 2 . . . 8
- 3.3 Task 3 . . . 8
- 3.4 Task 4 . . . 8
- 3.5 Task 5 . . . 8

4 Conclusion **8**

5 References **9**

6 Appendix **9**

Abstract

This report presents the findings and methodologies employed in the EEE3030 Signal Processing and Machine Learning course. It encompasses a comprehensive analysis of signal processing techniques, machine learning algorithms, and their applications in various domains. The report details the experimental setups, data analysis, and results obtained from implementing different models. Key insights and conclusions drawn from the study are also discussed, highlighting the effectiveness of the approaches used.

Plan for Report

1.1 Task 1 Plan

- Explain what the original file is, which is an AM signal with noise.
- Show this, showing the time domain output, where we can clearly see three peaks at x, y, z times.
- Show the frequency domain output using my fft code.
- Show calculations for sampling period and frequency resolution, explaining their importance in the context of FFT analysis.
- Explain the need for normalisation of the amplitude values.
- Show my fft code snippet.
- Explain why FFT is needed for analysis.
- Explain why windows are needed, show code snippets for different windows used.
- Show plots of different windowing methods and explain spectral leakage and also compare with unwindowed signals.
- Show calculations for fmin and fmax, talk about how I initially tried to use the proper methods, but then with jeff's help, I used the fact that the message bandwidth is 4kHz to determine the cut off frequencies for the bandpass filter.

1.2 Task 2 Plan

- Explain what an FIR filter is and why it is needed for bandpass filtering.
- Explain what bandpass filtering is and why it is needed in this context.
- Show the conditions needed for the filter design as a table, similar to the style in the pdf.
- Show how I designed the FIR filter (hand written calculations).
- Then show code snippet for FIR design in MATLAB and explain the code.
- Show the frequency response of the designed filter and explain what is going on.
- Show the frequency respones being applied to the AM signal.
- Show the code as well for the FIR filtering (convolution).

- Explain what convolution is and why it is needed
- Show the mixed signal in time domain, explain what is going on.
- Show the mixed signal in frequency domain, explain what is going on.
- Show the AM signal and explain its significance, and show how we can see the carrier frequency and sidebands.

1.3 Task 3 Plan

- Explain what the square law is and why we need it for task 3.
- Show the code snippet for squaring the filtered signal.
- Show the output of the squared signal in time domain and frequency domain.
- Explain what the plots mean (2fc and how we can approximate fc as a multiple).
- Show the code for carrier signal with phi as 0 and when we multiply it with the BP filtered signal. Explain that we have done that for ease and explain the code.
- Show the mixed signal in time domain and frequency domain. Explain what is going on in the plots.

1.4 Task 4 Plan

- Explain what an IIR filter is and why it is needed for low pass filtering.
- Show the conditions needed for the filter design as a table, similar to the style in the pdf.
- Show how I designed the IIR filter (hand written calculations) for Butterworth as well.
- Then show code snippet for IIR design in MATLAB and explain the code, and how it relates to the above.
- Show image with frequency response and explain why it is useful.
- Show the code for IIR filtering and plotting the output signal in time and frequency domain, then explain what is going on.

1.5 Task 5 Plan

- Explain the importance of phi and how it affects the output signal.
- Show the code snippet for the iterative phi increase and explain how the code works.
- Explain how SNR is calculated and its importance.
- Show the output signal in time domain and frequency domain for the best phi value obtained
- Explain the results obtained and compare with original message signal.
- Show the last output signal plot against original message signal plot.

Introduction

Results and Discussion

3.1 Task 1 Procedure, Results and Discussion

3.1.1 Reading the Audio File and Statistics

For the first part of this task, we need to load in the audio signal and determine its parameters such as sampling frequency, number of samples, duration, and frequency resolution. From **LISTING 1**, we can see the MATLAB code used to load the audio signal and display its parameters.

```matlab
%% Load the AM signal
[signal, fs] = audioread('Sahas_Talasila.wav');
signal = signal(:);  % Ensure column vector

% Display basic information
fprintf('Sampling frequency: %d Hz\n', fs);
fprintf('Signal length: %d samples\n', length(signal));
fprintf('Duration: %.2f seconds\n', length(signal)/fs);

% Time Domain Analysis
t = (0:length(signal)-1) / fs;  % Time vector in seconds
```

This piece of code uses MATLAB's `audioread()` function to load the audio file named `Sahas_Talasila.wav`. The signal is then reshaped into a column vector to ensure consistency in further processing. The sampling frequency (`fs`) and the length of the signal (number of samples) are printed to the console, along with the duration of the signal calculated by dividing the number of samples by the sampling frequency. From **FIGURE X** and the list below, we can see the terminal output from the code above which reveals the following information:

FIGURE X: Terminal output showing signal parameters.

1. Sampling Frequency (f_s): 96000 Hz
2. Number of Samples (N): 244104 samples
3. Duration (T): 2.54 seconds
4. Frequency Resolution (Δf): 0.39 Hz

The sampling frequency tells us how many samples of the signal are taken per second, which is crucial for accurately representing the signal in the digital domain. The number of samples indicates the total data points in the signal, and the duration gives us the total time span of the audio signal. The frequency resolution is particularly important for FFT analysis as it determines how finely we can distinguish between different frequency components in the signal. A smaller delta value means a much higher resolution, so we can clearly distinguish between closely spaced frequency components in the FFT output.

Sampling frequency is calculated using the equation below:

$$f_s = \frac{1}{T_s} \tag{1}$$

where:

$$f_s = \text{sampling frequency (Hz)} \tag{2}$$

The sampling period (T_s) is the time interval between consecutive samples in the discrete signal.

The number of samples (N) is simply the total count of discrete data points in the signal, which can be obtained using the `length()` function in MATLAB.

Frequency resolution (Δf) is calculated using the formula:

$$\Delta f = \frac{f_s}{N} \tag{3}$$

Where:

$$\Delta f = \text{frequency resolution (Hz)} \tag{4}$$

We can see all of these values printed in the terminal output from the MATLAB code above (**FIGURE X**).

3.1.2 Time Domain Analysis

Previously, the reader has seen some of the statistics and the audio loading code for the AM signal. It is paramount that we see the spectrum of the signal in both time and frequency domains to understand its characteristics. Firstly, we will plot the time-domain representation of the signal using the following MATLAB code snippet:

```matlab
%% Time Domain Plot

% This code has been commented out to reduce plotting during
   automated runs. (remove this comment later)

t = (0:length(signal)-1) / fs;  % Time vector in seconds

figure('Position', [100 100 1200 400]);
plot(t, signal, 'b', 'LineWidth', 0.5);
xlabel('Time (s)', 'FontSize', 12);
ylabel('Amplitude', 'FontSize', 12);
title('Received AM Signal - Time Domain', 'FontSize', 14);
grid on;
xlim([0 max(t)]);
```

The code above (**LISTING X**), very simply generates the plot for the time-domain representation of the AM signal. It is very clear to see that there are three distinct peaks in the signal at different time intervals, indicating the presence of significant events or features in the signal at those times. These peaks appear at (very roughly) 0.5s, 1.5s, and 2.2s, which likely correspond to the modulated message signals (our three characters) within the AM waveform. This can be seen below in **FIGURE X**.

FIGURE SHOWING THE TIME DOMAIN SIGNAL

3.1.3 Frequency Domain Analysis and FFT

It is important to analyse the signal in the time domain, as we can clearly see the amplitude variations over time, which is characteristic of AM signals, which provides us some simple information. But, for the purpose of filtering and extracting the message signal, we need to look at the frequency domain representation of the signal. The frequency domain representation can be obtained using the Fast Fourier Transform (FFT). The following MATLAB code snippet is used to compute and plot the FFT of the signal:

```matlab
%% Frequency Domain Plot using FFT
% Frequency Domain Analysis - Manual FFT
N = length(signal);   % Number of samples
df = fs / N;          % Frequency resolution

% Compute FFT
signal_fft = fft(signal);

% Create frequency vector (only positive frequencies)
f = (0:N/2) * df;

% Take single-sided spectrum (positive frequencies only)
signal_fft_single = signal_fft(1:N/2+1);

% Amplitude normalisation (convert to actual amplitudes)
signal_amplitude = abs(signal_fft_single) / N;
signal_amplitude(2:end-1) = 2 * signal_amplitude(2:end-1);  %
    Double non-DC components

% Convert to dB scale
signal_dB = 20 * log10(signal_amplitude + eps);  % eps prevents
    log(0)

fprintf('\nFrequency resolution: %.2f Hz\n', df);

%% Plot Frequency Spectrum
figure('Position', [100 100 1200 500]);
plot(f/1000, signal_dB, 'b', 'LineWidth', 1);
xlabel('Frequency (kHz)', 'FontSize', 12);
ylabel('Magnitude (dB)', 'FontSize', 12);
title('Received AM Signal - Frequency Domain', 'FontSize', 14);
grid on;
xlim([0 fs/2000]);  % Full range to Nyquist
```

The code above (**LISTING X**) computes the FFT of the AM signal and plots its magnitude spectrum in decibels (dB) against frequency in kilohertz (kHz). The FFT is computed using MATLAB's built-in `fft()` function, and the resulting frequency components are normalized to obtain the actual amplitude values.

The frequency vector is created to correspond to the positive frequencies only, and the amplitude is converted to a dB scale for better visualisation. We have also printed the frequency resolution to the console for reference. In addition, we only look at the positive frequencies since the FFT output is symmetric for real-valued signals.

Before discussing the results, it is important to understand the concept of frequency resolution in the context of FFT analysis.

The Fast Fourier Transform (FFT) is a method that quickly converts a signal from the time domain (how the signal changes over time) into the frequency domain (what frequencies are inside the signal).

Why is the FFT important?

- It shows which frequencies are present in a signal.
- It is much more useful for analyzing signals than just looking at them over time.
- It helps us design filters to isolate or remove certain frequencies.

Normalisation and scaling are needed for the FFT output to ensure that the amplitude values accurately represent the signal's strength at each frequency. Without normalisation, the FFT output can be misleading, as the raw FFT values depend on the number of samples and the specific implementation of the FFT algorithm. By normalising the FFT output, we ensure that the amplitude values are consistent and comparable across different signals and sampling conditions. It also makes it easier to interpret the results, especially when converting to a logarithmic scale (dB).

FIGURE SHOWING THE FREQUENCY DOMAIN SIGNAL

We can see from the frequency domain plot above (**FIGURE X**) that there are significant peaks in the spectrum, which correspond to the carrier frequency and its sidebands. These peaks indicate the presence of the modulated message signals within the AM waveform. The peaks are located at **X FREQUENCIES** The carrier frequency can be identified as the highest peak in the spectrum, while the sidebands are located symmetrically around the carrier frequency.

This frequency domain representation is crucial for designing filters to isolate and extract the message signals from the noisy AM signal. Initially, I attempted to determine the cut-off frequencies for the bandpass filter by analyzing the FFT of the AM signal.

However, this approach proved to be ineffective due to the high level of noise present in the signal, which obscured the relevant frequency components. Consequently, I resorted to a trial-and-error method to identify suitable cut-off frequencies for the bandpass filter. This was only possible after some guidance, wherein I decided to set the ranges to $\pm B$, where B is the message bandwidth of 4 kHz. This approach allowed me to effectively isolate the desired frequency components for further processing.

3.1.4 Windowing Methods and Their Effects

When performing FFT analysis, windowing is a technique used to reduce spectral leakage, which occurs when the signal being analyzed is not perfectly periodic within the observation window. Spectral leakage can distort the frequency representation of the signal, making it difficult to accurately identify and analyze its frequency components. By applying a window function to the signal before performing the FFT, we can mitigate the effects of spectral leakage.

3.2 Task 2

- Explain the cut off frequencies used for the filters.
- Include calculations for filter design.
- Show the FIR filter design code.
- Show convolution code.
- Show the frequency response of the filter and show the expected value.
- Show that I have applied the filter to the signal.

3.3 Task 3

- Square the filtered signal for a better response.
- Show code and plots for the envelope detection.
- Show the carrier frequency and the message frequency.
- Show fc times AM signal plot.
- Show the mixed plot in time domain AND frequency domain.

3.4 Task 4

- Show the code for the correctly designed IIR filter.
- Apply to output of task 3.
- Show the frequency response of the filter.
- Show the final output signal in time domain and frequency domain.
- Compare the output signal with the original message signal.
- Show convolution code if used.
- Explain the outputs

3.5 Task 5

- Show plot of output signal against original message signal.
- Calculate and show the SNR of the output signal.
- Explain the results obtained.
- Show that I have attempted to change the phase to increase SNR (using the incremental phase increase).

Conclusion

explain what was done in the report and summarise the results obtained. Talk about any challenges faced and how they were overcome. Discuss any potential improvements or future work that could be done based on the findings of the report. Need to add some stuff on how I initially tried to find the upper and lower limits for the bandpass filter by looking at the fft of the AM signal but this did not work well as there was too much noise. So I had to use trial and error to find suitable cut off frequencies for the bandpass filter.

References

References that I have used in the report. (articles, MATLAB documentation, textbooks etc.)

Appendix

Include some flowcharts for code design if possible. Include entire code listings if possible or split for the tasks. (code snippet for task 1, task 2 etc.) Include conv.m and iir filter design code.