**EE430 Lab 3 Writeup**

Section I – Narrative

Problem 1)

The majority of this problem involved code, which can be found in full in section 3, “source”. Comments for different sections are placed in the proper areas. Resultant graphs are included in section 2, “figures”.

Problem 2)

1. In order to determine which sinusoids the provided audio files had within them, each signal had the FFT function performed on it, and the power spectrum was observed. For signals 1 & 3, there were obvious peaks at 1000 and 880 Hz, respectively. The other two signals were too noisy to tell easily, and so the original signal for both was filtered using a FIR filter of length 500, with the assumption being that this should be adequate to ‘cut out’ any errant noise. The resulting waveforms still had no obvious peaks, and it was decided that they were both likely just noise. The generated figures are included in Section 2.
2. After putting the provided guitar note file through the FFT function and plotting it, two peaks were observed in the power spectrum: one at 440 Hz, and another at 880 Hz. This insinuates a fundamental frequency of 440 Hz, which is the musical note ‘A’, and is considered ‘pitch standard’. Considering the tone was perfectly on top of said frequency, it can be said that the guitar is not only in tune, but exceptionally so. Generated plots for this section are included in the second portion of this document.

Problem 3)

1. The requested filters were designed using the ‘fir1’ function. Graphs of the frequency responses are in the “figures” section of this document. It was noticed that a filter of a longer length compressed the response.
2. In the original filtering of the generated signal, the end result looked incorrect. The assumed cutoff of pi/8 (which had been calculated as corresponding to 500 Hz) was not cutting off the correct frequencies, with the exception of the highpass signal. After some recalculation, it appeared that a value of pi actually corresponded to the sampling frequency, and the values for the filters were adjusted appropriately. This resulted in a much better looking graph, with the signals present only being the desired ones. It is hypothesized that the original fir function was misunderstood.
3. When compared to the original noise, the filtered noise sounded much deeper in tone, and almost seemed ‘hallow’. This would appear to be due to the removal of both the high and low frequencies from the noise, which is exactly what a bandpass filter is supposed to do.

Problem 4)

1. The power spectrum of the provided morse code signal was analyzed, and the carrier signal appeared to be at 2700 Hz, as determined by looking at the peak value of the graph.
2. While attempting to design an effective filter for the signal, it was noticed that a shorter filter was more effective at maintaining the peak power of the signal, but left the surrounding noise mostly untouched. The longer the filter, the less apparent noise, but the carrier signal was also reduced as a consequence. However, over several trials and attempts, I was unable to design a filter that both cut out surrounding noise and left the main signal adequately untouched as to be able to clearly plot it.
3. As mentioned above, the original filter never worked adequately, however, the morse code was cleaned up enough to hear through the noise, and appeared to be the tone for S.O.S.

Problem 5)

This problem almost entirely used functions, which were written and are included at the fourth section of this document. When inputting the sequence ‘65456665556666545666655654’ into the sequence dialer, the tone did indeed sound similar to the classic tune of “Mary Had a Little Lamb”, but other sequences found online did not seem to sound quite right. It is believed that this may be due to the note spacing being constant, and a lack of inflection.

The spectrogram of the signal, included in the following section, clearly showed that the signals were working as intended.

Problem 6)

1. In order to decode a DTMF tone, three different functions were first created. The first normalized signals, ensuring that there would be no DC component by subtracting the mean value of the signal from itself, and then the maximum values were divided in such a way as to ensure that the maximum value of the waveform was not above ‘1’. Two smaller functions were also used with the signal, one that would seek the first value above a threshold, and another that would do the same but starting from the end of the signal. Combined, the values returned from this process could be used to ‘trim’ the signal to only the relevant part. Finally, code was constructed that would analyze the FFT of the signal in order to find both the high and low frequencies present in the signal. The output from the FFTs and code can be found in the relevant portions of this document,
2. The code was used to find the digit pressed, start time, and end time for six different signals as provided by the laboratory instructor. The output of the function is included on the following page.

=====REPORTING DTMF DATA FOR FILE 1=====

Digit : 2

Start Time : 0.200250 Seconds

End Time : 0.599250 Seconds

=====REPORTING DTMF DATA FOR FILE 2=====

Digit : 4

Start Time : 0.100250 Seconds

End Time : 0.599125 Seconds

=====REPORTING DTMF DATA FOR FILE 3=====

Digit : 9

Start Time : 0.300250 Seconds

End Time : 0.999750 Seconds

=====REPORTING DTMF DATA FOR FILE 4=====

Digit : #

Start Time : 0.200045 Seconds

End Time : 0.699728 Seconds

=====REPORTING DTMF DATA FOR FILE 5=====

Digit : 8

Start Time : 0.300045 Seconds

End Time : 1.099546 Seconds

=====REPORTING DTMF DATA FOR FILE 6=====

Digit : 5

Start Time : 0.200045 Seconds

End Time : 0.700045 Seconds