ELEC1200 Project Report

PART I

Description

The first part of the project is a simple speech recognition program. Two chosen single syllabus English words, “bar” and “door”, are stored in two .wav files “word1.wav” and “word2.wav” respectively. When the program is executed, it will prompt the user to choose between “loading one of the existing .wav files” and “speaking and recording on the scene”. Then the program will output which word was loaded or was spoken by the user. This is only a binary word recognition program, as it can only decide whether “it’s word1 or word2”. It is unable to detect and decode speech signal into any arbitrary English word.

Technique/Algorithm

The main idea of this word recognition program is to make use of the difference between frequency spectrums of different words. The three important pieces of data are the signal of “word1.wav”, the signal of “word2.wav” and the signal originated from the user’s decision (Either a speech signal recorded on the scene, or the same as one of word1.wav and word2.wav), all of which are represented in the time domain. I will preform the Fourier Transform on all of the three signals using the `fft()` function provided by MATLAB in order to obtain the frequency spectra of the three signals. As the result of the Fourier Transform includes both the magnitude and phase, yet I am

only interested in the magnitude, I will add `abs()` to only make use of the magnitude of their spectra. After this, I will have three different spectra, one for “bar”, one for “door” and one for the spoken/loaded word. Technically, now if we plot the three results of `abs(fft(signal))`, the horizontal axes corresponds to the frequency components, and the vertical axes corresponds to the corresponding amplitude of each frequency components. By visual inspection of the plots, we can already tell the difference between them in terms of where the maximum amplitude occurs. To simplify the comparison, I will not focus on the exact frequency where the maximum amplitude occurs. In other words, the exact value of the dominant frequency component of the signal is not important in terms of determining which word was spoken/loaded. Instead, as essentially the data of the three signals of stores in vectors/arrays in MATLAB, I will mainly focus on the index where the max amplitude occurs in the array. The index corresponding to the dominant frequency component of “bar”, “car” and the spoken/loaded word is stored in `ind_bar`, `ind_door` and `ind` respectively by using

```
[max_mag_of_bar, ind_bar] = max(FT_of_bar);
[max_mag_of_door, ind_door] = max(FT_of_door);
[max_mag, ind] = max(FT_of_y);
```

Then the decision of the program will be made based on the numerical difference between `ind` and `ind_door` and the numerical difference between `ind` and `ind_bar`. The two numerical differences are stored in the variables `diff_between_bar` and `diff_between_door` respectively. Whichever returns a smaller value suggests that the spoken/recorded word is “more similar” to that prerecorded word. If the user chose to load one of the existing files, then one of the index difference will simply be 0.

Testing and Accuracy

Since if the user’s decision was to load one of the existing files, the index difference between the loaded file with itself must be zero. There is no point testing this scenario as it’s just impossible to have any mistakes in terms of determining which word was loaded. The testing focus was on making decisions according to the recording on the scene. I used my own voice to randomly say one of the two words several times and check both the eventual output and the value of `diff_between_bar` and `diff_between_door` in the MATLAB workspace. The bigger the difference between `diff_between_bar` and `diff_between_door`, the better the detection performance will be. In terms of eventual output, no error was detected during my testing. But the difference between `diff_between_bar` and `diff_between_door` is quite unstable. Sometimes their difference is quite significant, which suggests that the recorded sound

is extremely similar to one of the words and differs a lot from the other word. Yet there are occasions where the difference is not large, indicating that the recorded wave lies approximately in the middle between the two known data waves. This issue should be caused by the fact that even the same person repeatedly pronounces the same word for several times, each time the spectrum of his or her sound may not be the same. Sometimes there's an overall shift to the high frequency direction, sometimes the opposite. Limited to a binary decision, if the frequency shift due to human voice each time isn't large enough to make the gap reverse to the opposite direction, we can still obtain the correct output. But this indicates that the recognition accuracy may depend on the speaker's tone, which means a high accuracy when the speaker is same as the person who recorded the data waves may not guarantee a high accuracy when the speaker is different.

PART II

Description

The second part of the project is about designing a transmitter and receiver to minimize the bit error rate when transmitting message across an unknown channel. Ideally, the transmitter and receiver, working with an agreed protocol, should be able to transmit any message with no bit error.

Problems/issues found

The first thing I did before starting to design any part of the transmitter and receiver was to set a fixed bit sequence at random (e.g.: 001100111100) and just directly send it through the channel with out doing anything. I plot the graph of the transmitted bit sequence, transmitted waveform, received waveform and received bit sequence and observe the channel effect by particularly comparing the transmitted waveform and the received waveform. After repeatedly doing raw transmission multiple times it can be observed that the channel can possibly cause the following issues:

1. Square waveforms will result in gradual changing waves similar to exponential growth/decay. Due to the settling time, the inter symbol interference is quite severe.
2. The whole wave can have both a vertical shift and a horizontal shift (i.e.: having an offset and time delay), making this an asynchronous communication system.

3. The system produces a random noise along the way, causing high frequency components to occur in the received waveform. Sometimes the noise can be so big that the wave may cross the threshold.

The above issues are the main problems I will try to fix when designing my communication protocol and doing manipulation on the received waveform.

Methodology used and testing method

My methodologies are pertinently used to resolve the previously mentioned three major issues caused by the channel. To tackle the first issue, I used the same function and techniques in lab2 task3, where we modeled the channel as an exponential mathematical expression $\ y(n) = c + k(1-a^{(n-d)}) \ $, and used unit step as a training sequence to estimate the four parameters $\ c, k, a, d \ $. Here as we are unable to tune the parameters on the scene, and the channel may change over time, I used a simple algorithm to automatically estimate the four parameters. The estimation of $ c\ $ and $ k\ $ is quite cursory. I simply choose the minimum value of the received unit step to be the offset $c\ $ and add approximately $ 0.02\ $ to it because of the noise. $ k\ $ is estimated by finding the difference between the maximum and minimum value of `rx_wave` and minus $ 0.03\ $ to it due to noise. This is indeed a cursory estimation but it's hard to do significantly better without any other information. Before estimating $d\ $ and $a\ $, I first applied a low pass filter to filter the wave form to make my estimation more precise. The filter I used was a simple mathematical model known as "moving average". It's like a window that propagates through the whole wave and filter out the high frequency components to eliminate or reduce the effect of noise. Here the window length I chose was 10. If the window length was too small, the filtering effect may not be good enough, but if it was too large, it may conceal too many actual values and leading to a loss of accuracy. The basic idea is to let each sample be the mean value of its neighboring 10 values, and it's achieved by propagating the window using a for loop as follows.

```
filtered_rx_wave =[c,c,c,c,c,c,c,c,c,c];
window = filtered_rx_wave;
for i = 11:length(rx_wave)
    filtered_rx_wave = [filtered_rx_wave,mean(window)];
    window=[window(2:end),rx_wave(i)];
end
rx_wave = filtered_rx_wave;
```

Then d is estimated by finding the sample where there are 40 continuous increments. I had to make the condition such rigorous as the noise can't be completely eliminated and one single increment may be purely due to noise. The estimation of $d\ $ is just a simple "trial and error" idea. Using the previously estimated $c, d\ $ and $k\ $, $a\ $ is estimated by finding the best $a\ $ that gives the smallest MSE from $a\ $ ranging from $ 0.5\ $ to $ 1\ $ with a step length of $ 0.01\ $. After obtaining the optimal four parameter estimations, I can derive the equalizer and apply it to the received waveform. ISI is now significantly deduced. To further eliminate the effect of noise, I perform moving average again to filter out high frequency components, and then determine the threshold by finding the mean of `max(rx_wave)` and `min(rx_wave)`. This solves the third issue. To tackle the second issue, apart from the training sequence, I also added an "ending sequence" at the end of data bit sequence. So, on the receiver side, I can locate the actual data by identifying the training sequence and ending sequence in the decoded waveform. This solves the synchronization issue. To do testing, apart from using the given `txtext.mat`, which was successfully transmitted and received with no bit error, I also tried transmitting some other random text message with appropriate length. As `compute_BER` can only regard the given `txtext.mat` as the transmitted message, I used an online text comparer to compare the original message and my decoded result. No error was found, which suggests my protocol works well with the given channel.