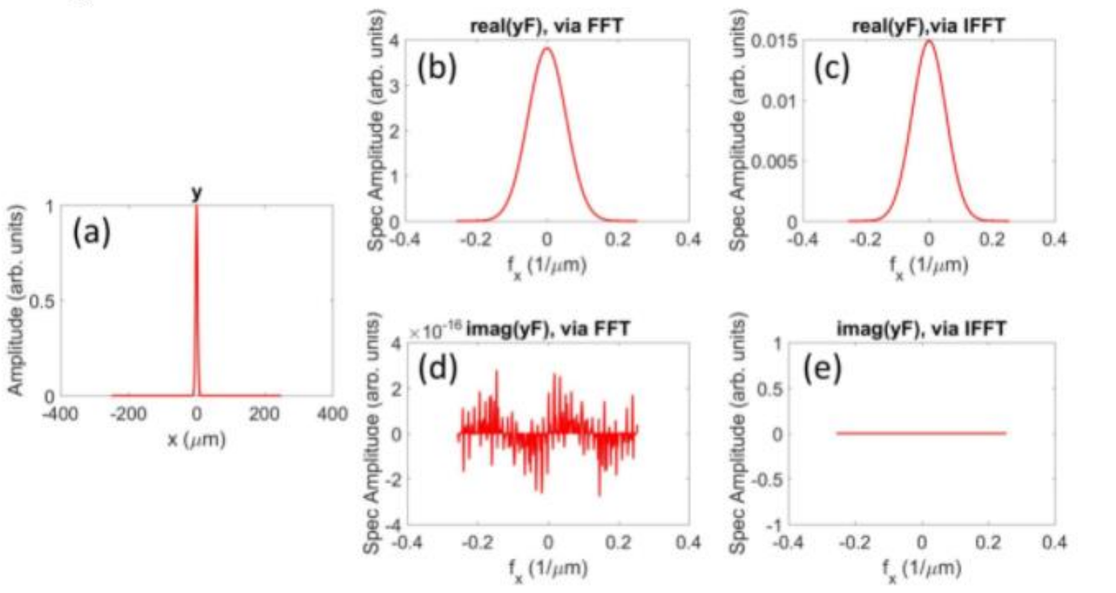
In Part One, which is the Simple Speech Recognition Part. I have designed to use "One" and "Two" as the 2 words that I need to distinguish because these two words have obviously different frequency spectrums, which means that it is very easy to distinguish identify between these 2 words.

For the process that I distinguish these two words. First, Since the wav files are stereo channel which carries 2-channel. I change the waveform to mono-channel so that I can just look for the one-sided channel. After that, I use max function to find out the bigger amplitude and then use it to divide the entire waveform so as to resolve the effect of amplitude of different wav files.

The most important process is that I use Fast Fourier Transform (FFT), converting a digital signal with length from the time domain into a signal in the frequency domain. Since every word has its unique set of frequencies (frequency domain). After getting the frequency domain. Since Fourier transform is to get the time domain convolution, which is similar as the correlation operation of convolution, I use the convolution to find out the similarity of the 2 samples wav files with the input file. The higher the similarity, the more similar as the sample word. Therefore, I can distinguish the input word between the two samples. Besides, I take the FFT function with the length of the wave multiplied by 2 to gain the accuracy since 2 to the power of length is too large.

Plotting the results:



The IFFT does a perfect job: yIFFT is a purely real Gaussian. However, FFT yields a complex number: a very small imaginary part exists. This is fine, since an error should be expected in the fourier transform algorithms, and it's negligible anyway. What confuses me is ***why there is no error at all*** in IFFT? Are

After acquiring the knowledge about FFT and IFFT function in MATLAB, which is a kind of Discrete Fourier Transform method and it is a method to make the Fourier transfer becomes observable. Hence, I designed to use the real part of the result of the demodulated signal since the imaginary part will become “zero” after applying Inverse Fast Fourier Transform.

For the accuracy of this speech recognition program, I would say that it has a high accuracy when comparing with 2 words having substantial difference in their frequency domain. So, when we test about 2 words that have a similar frequency domain, the accuracy will decrease rapidly. For example, if I take 2 samples which are both in high pitch, their frequency domains will be very similar that is hard to figure out the input by the method of convolution.

To reiterate, the principle of this simple speech recognition is convolution, getting the maximum of the cross-correlation of the two signals, and output the sample with a higher cross-correlation index.

In Task2, I need to design a transmitter and a receiver, and I have used some basic technique to design both. The methods are changing the waveform to bit sequence, adding sequence to find the delay, offset and attenuation, sampling up the bit sequence by sample per bit, comparing with the received waveform to the threshold to acquire a bit sequence, taking sub-sampling to get the original bit sequence and converting the bit sequence into character.

The problem that I had found is that I can only predict the value of delay, but I could not actually figure out it. Therefore, If I took a sub-sampling which is far away from the real value of delay, I would get bit error easily.

Regarding the transmitter part, I firstly convert the text message into bit sequence, changing each character into ASCII code in binary format. For example, character ‘I’ is transferred as ‘01101001’. Therefore, I design to make a start bit (‘1’) before the bit sequence and add a stop bit (‘0’) after the bit sequence so that I can realize the beginning of my data. Also, the length of the transmitted samples Is limited by 200,000 samples. Therefore, I convert the bit sequence into waveform by sampling up the bit sequence with the sample per bit, and I choose 49 as my sample per bit is because I want to use the maximum of the sample length and give more time for a bit to get a more accurate value that is lower than or higher than the threshold during the receiver part. Finally, I have added a training sequence before transmitting the waveform since it is useful for finding the value of attenuation(k), delay(d) and, offset(c).

After completing the transmitter part, it comes to the receiver part. First, I use the training sequence to get the c, d, k value. By knowing the offset and attenuation value, I can predict the threshold easily and convert the waveform into bit sequence. Also, by shifting the waveform by value d, I can predict the beginning of the start bit, and this information can fully help me reverse the waveform into transmitted text. By using the sub-sampling method, I can extract each message bit by taking a sample in 49 samples. Although there is a knowledge that the beginning of start bit adds with 2*SPB-1 is the end of the first message bit, the value of delay sample may not be as accurate as the real delay. As a result, after I looped the whole bit sequence of the waveform which sampled up by sample per bit, I can get back the original text message by converting it into text format and then convert it as character.