EE430 Lab 4 : Real-time DTMF decoding, and Demodulation / Demultiplexing

1) **Real-time DTMF Decoding**

In this problem we will study adapting the DTMF decoding function that you wrote previously to a simple real-time DTMF decoder. Real-time signal processing systems work by storing incoming signal data into one or more buffers, and then processing the previously stored data simultaneously with storing the next block of signal data.

Tasks :

i) Write a MATLAB function with signature `digit=decodeDTMFblock(x,fs)`

that takes an array of input audio data `x` at sampling frequency `fs`, and then returns either a valid DTMF symbol, or an empty string, if there is no DTMF signal detected in the given signal.

I would recommend proceeding by analyzing the FFT, finding the two largest peaks, and then comparing those identified peaks to the DTMF frequencies.

Determining whether there is DTMF signal present could be done by computing the ratio of the height of the identified peaks in the absolute value of the FFT to the average of the absolute value of the FFT over the entire frequency spectrum. If the two identified peaks were sufficiently close to the DTMF frequencies, and the ratio described previously exceeded some threshold, then one could say that DTMF signal was present.

ii) I have provided the MATLAB function `audio_loop_template.m` which provides a simple framework for near real-time audio analysis. Run the script and familiarize yourself with what it is doing. Then, modify the script so that it calls your `decodeDTMFblock` function in the processing section of the loop, and prints the identified DTMF symbol. If this is working properly, and the computer has a microphone attached, your code should print out the DTMF symbols if the microphone picks up a DTMF signal. Test this by playing DTMF tones, you may either use an actual touch-tone phone, or a touch-tone phone simulator such as

`http://onlinetonegenerator.com/dtmf.html`.

Describe how well your DTMF decoding system works, and how you made any choices that you made for parameters in your decodeDTMFblock function.

iii) Extra credit : Modify your code so that holding a single key for a long time (longer than the blocksize in the `audio_loop_template.m`) will not result in the same DTMF symbol being printed multiple times, by ensuring that a new symbol is only printed if it is different than the last symbol printed, or if there was an interleaved audio block with no DTMF symbol detected (so that, for instance, the symbols 33 could be printed by pressing 3, waiting some time, and then pressing 3 again).

2) **Demodulation and demultiplexing**

I have provided an audio file `lab4_modulated_audio.wav`. This signal was produced by modulating 4 different radio program segments (by lowpass filtering, then multiplying by a sinusoidal carrier $cos(2\pi f_c t)$), with 4 different modulation frequencies, and adding them together.

Your job is to demodulate and demultiplex each of these component signals, so that you can transcribe the audio speech. You can do this, as described in Chapter 12-2, by demodulating and lowpass filtering, 4 separate times, with the 4 different carrier frequencies.

I am not telling you the carrier frequencies f_c, however I will tell you that they are all exact multiples of 500 Hz. I am also not telling you the original bandlimit of each of the component signals. You can determine each of these by looking at the spectrum of the provided audio signal.

Tasks :

i) Plot the spectrum of the provided audio signal, determine the 4 carrier frequencies that were used, and the approximate bandlimit of the 4 original audio signals.

ii) Demodulate and lowpass filter using the carrier frequencies that you determined. You must design appropriate lowpass filters, you may use either fir or iir filters. Plot the spectrum of each signal after demodulating and before filtering, and also after filtering.

iii) Listen to and transcribe each of the 4 audio segments.