EECE 2150 - Circuits and Signals: Biomedical Applications

Lab 12 Frequency Content of Signals

Introduction

In this lab you will use Matlab to measure the frequency content of signals, you will try to connect what you hear in certain signals with an ECE understanding of frequency, and you will see if you can remove (or filter out) unwanted noise/interference from a signal by removing particular unwanted frequencies from the signal. Note that the filtering method you will use is not used much in practice. There are much better ways to build discrete time filters that have more precise control over how they affect input signals. However, the method you will use in this lab is simple to implement and understand and can give interesting results.

Part I – Frequency Content of Recorded Sounds

1.1 **Record yourself speaking on your computer** for about 3 seconds. Use the recording application Audacity to record. Plug in the mic, you will be asked for the device plugged in, choose mic in. Press the red record button to record your audio signal. To stop recording, click on the Stop button. Then Export the file (File >> Export as WAV format). Choose a filename and location.

1.2 Read in the sound using the audioread function in Matlab. If you do not know how to use the audioread function type **"help audioread"** or **"doc audioread"** for instructions. Note from the help that audioread returns the sampling rate as well as the audio signal. You will need to use the sampling rate so be sure to specify an output variable to save it. Helpful hints: Note that the WAV file must be in the Matlab path. Also note that the file name must be in single quotes when you use the audioread function (…audioread('myfile.wav')).

1.3 Now, analyze the frequency spectrum of your file using the provided plot_frequency_content Matlab function (This is something from the class, not a Matlab function!). To be sure how to use the plot_frequency_content function type "help plot_frequency_content." After you obtain the plot of the signal frequency content versus frequency, use the Edit menu to modify the scale of the x-axis so that you can see the lower frequencies better. Below is a sample example of reading a saved audio file and using the plot_frequency_content function to plot the frequency content of the signal.

```
 %'Sound_wav.wav' is the name of the sound wave file used in this example
% reading the audio signal and returning both the signal and the sampling rate Fs
 [signal, Fs]=audioread('Sound_wav.wav');
 plot_frequency_content(signal,Fs)
 xlim([ 0 7000]);   % Limiting the frequency axis to 7000 Hz
```

Q1: Describe the frequency content of your spoken audio signal. For example, what frequency has the largest amplitude? Where is most of the signal located in the frequency spectrum, qualitatively?

1.4 Next, use the stft_sound_plot function provided to examine the frequency content of shorter intervals of your sound, to find out how the frequencies shift as you go from one part of the sound sample to another. (This is also from the class!) You may need to play around with the window size parameter to get the best results. Note that this parameter is in number of samples, so look at the total number of samples and adjust the window for an appropriate subset. To start, you may want to divide your signal into about 10 parts, so make the number of samples in the window about 1/10 of the number of total samples. Note that stft_sound_plot expects a 1-D array, whereas the read functions generate a 2-D array. You need to supply only 1 column of the array. You can do this in Matlab using **x(:, 1)** (if **x** is the 2-D array). Below is an example for using stft_sound_plot.

```
%'Sound_wav.wav' is the name of the sound wave file used  in this example
[signal, Fs]=audioread('Sound_wav.wav');
signal_Left=signal(:,1);  % Selecting one of the channels
Window_Size = ceil(length(signal_left)/10); % divide signal into 10 parts
stft_sound_plot(signal_Left, Window_Size, Fs);
```

Q2: How much does the frequency content change from one part of your sample to another. Record plots of at least two very different sub-intervals.

1.5 Try this with a number of different sounds:

i) Try generating a vowel or a constant pitch tone. See if you can achieve a tall spike in your frequency spectrum by keeping the pitch constant, and see if the harmonics change depending on the vowel! Hint: You will almost certainly need to expand the scale of the x-axis to see the fundamental frequency and harmonics of the constant pitch sound. **Q3: Did you get a tall spike in frequency? Were there other harmonically related spikes (as we might expect for a periodic signal?**

ii) Try generating a "noise-like" sound like "sh" or "ffff". **Q4: How is the spectrum of this sound different from the constant pitch sound? Does this sound have more high frequencies, low frequencies, all frequencies, or spikes, or something else?**

iii) Compare similar sounds made by lab partners, such as two vowel sounds. **Q5: Can you see differences between the frequency spectra?**

Part II – Frequency Content of an ECG Signal

2.1 Load the example ECG data "**ecg_waveforms.mat**" into Matlab. This contains two ECG traces: i) **ECG1** - a noisy ecg file and ii) **ECG2** - a longer ECG trace. **Plot these in Matlab** to be sure they have loaded correctly.

2.2 Use the plot_frequency_content function to examine the frequency spectrum as shown by the Discrete Fourier Transform (DFT) of the signals. (plot_frequency_content uses the Matlab fft function to calculate the DFT.)

Note that the sampling rate was **Fs = 1000 Hz** for both signals. Again, after seeing the big picture (frequencies up to 500 Hz), it is important to expand the scale of the x-axis so that you can see the details of the low frequencies.

```
Fs=1000;
load ecg_waveforms.mat; % load ecg data file;
                        % two ecg signals are included
figure(1)
plot_frequency_content(ecg1,Fs);
figure(2)
plot_frequency_content(ecg2,Fs);
```

i) For both signals: **Q6: What is the overall frequency range that has the dominant frequency content of the signal?**

ii) For ECG1: **Q7: What is the frequency of the major noise content in the signal?**

iii) For ECG2: **Q8: What is the person's heart rate in beats per minute?** If you are not sure what to expect, check your own pulse to see what kind of frequency range you should expect, and then use that to guide your effort to answer this question. Again – you will have to expand the low-frequency part of the signal spectrum!

iii) **Q9: Do you see any other noticeable noise in both signals?** In particular look at frequencies that are integer multiples of the noise you found in in ii).

2.3 **Q10: If you were building an amplifier for ECG, what frequencies would you want to amplify? What frequencies would you want to reject?**

2.4 **Q11: What would be the minimum sampling rate you would want to record (digitize) someone's ECG?**

2.5 **If you have time:** Try to use the result above (the frequency of the most significant noise content) to filter out the noise by modifying the FFT coefficients. There are many ways you can do this, both in terms of locating which coefficients to change and deciding what value to set them to. One method to locate them would be to call the Matlab fft function, plot the magnitude (using the abs function, since the DFT is generally complex-valued) and using the data cursor to find the location of the values you want to change. You can set the coefficients to

zero, or try to set it to approximate the “neighboring” coefficients at nearby frequencies. Experiment with both methods. After you change the DFT coefficients, use the inverse fft function using ifft in Matlab to get the filtered signal. How well does this work? Do you get close to what you would expect?

*One very important detail: you **must** change coefficients in appropriately spaced **pairs**, one towards the first half of the DFT and one in a matching position in the last half. When you plot the magnitude of the fft you will see that there is this symmetry. It is important to preserve it. (Using* plot_frequency_content *function only plots the first half of the DFT, to make it easier to interpret ,so you do not see this symmetric part with that function.) If you do not preserve this symmetry you will not get a real-valued signal back. Ask if you are having trouble with this or what to know more about why this is the case.)*

Department of Electrical Engineering, Northeastern University.
Last updated: 11/5/2021, by Iman Salama 3/15/16 by N. McGruer