Signal Processing for Machine Learning
Fall 2022

Project 1

Due: September 19, 2022

1. Consider the audio file TONE.wav.Your charge is to write a MATLAB program to remove 4 largest harmonics of the interfering tone and submit a report of your analysis that includes answers to the questions given below.Your deliverables are the report and a MATLAB program. Words written in **boldface** letters refer to MATLAB variables you should use.

   (a) Read the file TONE.wav into a vector called **speech**. Note its sample rate **sf**. Plot **speech** versus **time** and play it using *soundsc*. Note the length of **speech** as **lengthspeech**. Compute the **nfft**-point DFT of **speech** where **nfft** is 4 times the smallest power of 2 larger than **lengthspeech**. Save it in **dftspeech**.Plot the magnitude **dftspeech** versus normalized frequency ω between 0 and 2π, versus normalized frequency ω between $-\pi$ and π and the actual frequency $F = \frac{\Omega}{2\pi}$ between +/- half the sample rate. Comment about the file content based on listening to the file and observing the plots. Note the largest spectral magnitude and its actual and normalized frequency.That is the major interfering tone. Create a sinusoid of the same frequency and listen to it. Compare the tones. Follow the instructions below to design a filter to remove the interfering tone. The filter has the transfer function

$$H_0(z) = \frac{(1 - e^{j\omega_0} z^{-1})(1 - e^{-j\omega_0} z^{-1})}{(1 - re^{j\omega_0} z^{-1})(1 - re^{-j\omega_0} z^{-1})}.$$

   i. Find ω_0. Explain how you found it.

   ii. Choose r near but less than 1.$r = 0.98$ may be a good choice. Experiment with it.

   iii. Note the poles and zeros of $H(z)$ and plot them on the complex z-plane using MATLAB.

   iv. Write the transfer function in terms of the filter coefficients as

$$H_0(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}.$$

   Record the filter coefficients in **b** and **a** in your program.

v. Use MATLAB function *filter* to filter **speech**. Record the output in **clean1**

A. Plot **clean1** versus **time**.

B. Listen to **clean1**.

C. Compute the **nfft**-point DFT of **clean1** and save it in **dft-clean1**. Plot its magnitude.

D. Compare the outputs to that of the original speech. Comment on the results.

(b) Repeat the filtering as above using $H_1(z), H_2(z), H_3(z)$ to notch the next 3 interfering tones at $\omega_1, \omega_2, \omega_3$ in cascade. Use time and frequency domain plots and listening tests as necessary and write the results.

To find $\omega_1, \omega_2, \omega_3$ you may use "the human in the loop" system, i.e. find the frequencies manually. Alternatively, you may write code to find a local peak. You may contact me for discussion on how.

(c) Obtain the overall filter $H(z)$ in terms of $H_i(z), i = 0, 1, 2, 3$ and plot its poles and zeros on the z-plane. Plot also its spectral magnitude superposed over the original speech spectrum to show how the filter's zeros and the interfering tones align.

2. Speech is a non-stationary stochastic process. Its characteristics are not represented adequately with a DFT on a large segment as we did in Part 1. Also note that the speech segment in TONE.wav is only a segment of a longer speech which is possibly a stand-in for a real-time conversation. Processing the signal should be in real-time or pseudo real-time. Here we will do the latter. Real-time usually refers to changing the output sample by sample, which is the subject of adaptive signal processing. Pseudo real-time means we process small, contiguous segments, that is several samples at a time. How long should each segment be? The decision is based on a good representation of the speech signal, the interfering tone and the amount of delay introduced into a communication system while we process. If we analyze a 200 ms segment, and it takes 10 ms to process the segment, then the output will be delayed by 210 ms.

Use the MATLAB function *frames* provided with this assignment to divide **speech** into 200 ms segments called frames overlapping by 80%. Arrange the frames in the columns of a matrix **speech_frames**. Note the length and the number of frames.

(a) Analyze the segments first.

i. Compute the **fr_nfft**-point DFT of each frame by using fft on the matrix **speech_frames**. Choose the DFT length using the same criterion you used to choose **nfft** adjusted to the frame length.

ii. Make a movie out of plots each frame versus time and frequency.

iii. Generate a spectrogram of the frames using *imagesc*.

iv. Comment on your observations.

(b) Assume that the signal in TONE.wav is available to you in frames in pseudo real-time. Remove the dominant interfering tone one frame at a time. After you filter each frame, collate the output to generate an alternative "clean" signal that you can listen to. I will discuss this in class.