ECE2312 – Discrete-Time Signal and System Analysis

Project 1

Due: 5.00 pm on February 2, 2023

1. Project Goals

This first project in our course focuses on mastering sophisticated software tools, e.g., MATLAB or Python, to collect, store, and analyze human speech signals. The samples in this project use MATLAB but feel free to use Python. Through these three projects in this course, we will explore speech signals. Upon completing this project, you will possess the necessary skill set to handle speech signals in the following two-course projects.

2. Understanding Human Speech Production

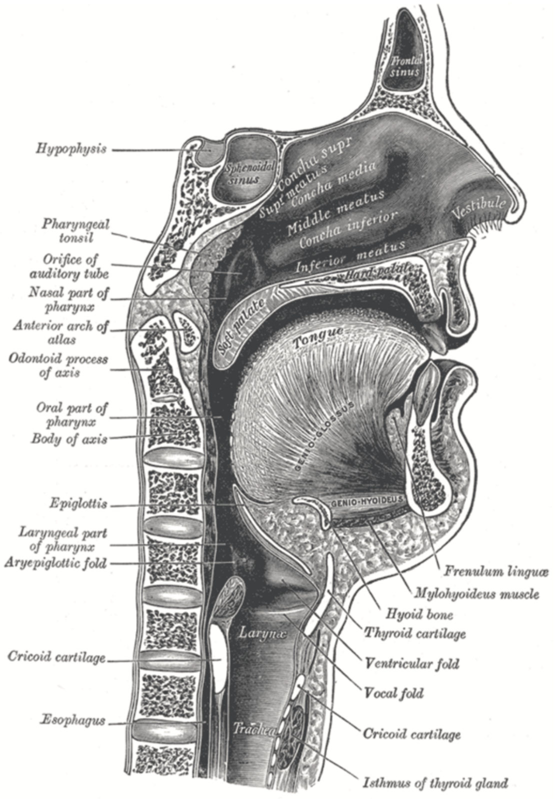

Figure 1. Sagittal section of nose, mouth, pharynx, and larynx. From Gray's Anatomy 1918 edition

In a nutshell, the production of human speech signals is the result of taking acoustic vibrations emanating from a pair of vocal cords and manipulating them into a wide range of different phonemes using the throat, nasal cavity, tongue, and other parts of the human vocal tract, as shown in Figure 1. Based on the physical characteristics of the vocal tract at any given moment, different phonemes can be produced, which can be generated sequentially to result in human speech communications, i.e., a string of phonemes concatenated together to produce words and sentences.

Interestingly, no two human beings are identical when it comes to possessing the same physical characteristics of a vocal tract, including biological twins. Consequently, for the same vibrations of the vocal cords, no two individuals will produce the same phonemes for a given configuration of a vocal tract. Factors such as the diameter and length of the throat, the volume of the nasal cavity, the shape and position of the teeth, the size and relative position of the tongue, and many other physical characteristics will all influence how each phoneme will sound. With modern digital signal processing, it is now possible to identify these differences clearly and quantitatively between two humans producing the same phoneme.

Given our knowledge of human speech production and digital signal processing, numerous unique frequency characteristics exist in a human speech signal that can be used as a speech "fingerprint" to distinguish between two human speakers. Some of the most readily observable differences between a phoneme produced by two human speakers can be seen in vowels, where the frequency locations of the first, second, and third harmonic frequencies, or formants, occur at sufficiently different values. As it turns out, several algorithms and techniques exist that can classify which human is speaking based on which kind of phoneme (e.g., vowel, fricative) is being detected and who is potentially saying it.

3. Record Your Speech

The first critical step in conducting this project is recording an analog speech signal from an actual human being into MATLAB. Although this might initially sound trivial, this process is quite a bit more complicated than it seems. As was mentioned on the first day of the course, the bridge between the analog world and the digital world is the analog-to-digital converter (ADC) and the digital-to-analog converter (DAC). Using these two components, we can quickly handle signals in both domains. However, there is more than meets the eye concerning this conversion process between the two domains.

For instance, we will need to decide upon an appropriate sampling rate (F_s) that will take an analog signal and transform it into a collection of discrete-time samples per unit of time. If we do not collect enough samples, we can potentially lose information about the analog signal of interest. On the other hand, too many samples per unit of time could be challenging to work with and make our discrete-time signal processing implementation inefficient. Common sampling frequencies for speech and audio applications include 8000, 11025, 22050, 44100, 48000, and 96000 Hz.

Another consideration when it comes to bridging the analog and digital worlds is the quantization needed to map a sample amplitude to minimize any quantization error accurately. In other words, we will need to map the infinite continuum of possible amplitude values of the recorded speech signal to a finite and discrete set of amplitude values, where each value possesses a unique binary word representation. The number of possible amplitude values is equal to 2^b, where b is the length of the binary word representing each unique amplitude value.

In MATLAB, you will need to use the following functions in the order listed such that real-time human speech signals can be recorded: audiodevinfo, audiorecorder, record, and getaudiodata. To understand how each function works, MATLAB possesses a comprehensive online help manual describing the input parameters, output values, and various options available for each function. Regarding implementation, you will need to use audiodevinfo first to obtain the appropriate ID for your computer's audio device. Please note that your computer should possess an audio card and a microphone. Although USB devices such as webcams might work, they are much more challenging to set up than a built-in audio device. Using the output structure produced by audiodevinfo, this information can then be used by audiorecorder along with an approximate choice of sampling frequency and sample resolution in bits to produce a handle that is used by the record function to obtain data from your audio device.

One important warning: When using a record, refrain from using the results from the recording until the process has been completed. The reason is that MATLAB would execute functions immediately one after another regardless of if the function just executed was completed. For example, suppose you are recording a speech signal for 5 seconds; then, all the functions following that record will be executed immediately after the function call rather than when the 5 seconds of recording has been completed. This becomes an issue when a subsequently called function depends on the recorded speech data, such as play, which immediately plays the recorded speech in the MATLAB environment. If the record is not done recording, then play will produce an error message saying that the speech memory buffer is empty since the record will not save the recorded speech to memory until the end of the recording. Once the recording is complete, you can test your speech recording script using the play function. If you are doing this in a public space, please use headphones.

Q

Using the getaudiodata and plot functions, produce three separate plots of the time domain representation of the following three phrases:

- "The quick brown fox jumps over the lazy dog"
- "We promptly judged antique ivory buckles for the next prize"
- "Crazy Fredrick bought many very exquisite opal jewels"

Please make sure that the x-axis and y-axis are property labeled, and the x-axis is scaled properly in terms of seconds.

4. Visualization via Spectrogram

From the previous section, we observed how speech signals appear in the time domain. Although useful, it is often important to also observe the behavior of these speech signals simultaneously in the frequency (i.e., spectral) domain as well since there are numerous features that we can leverage in order analysis and treatment of speech signals. One powerful tool that can be leveraged is the spectrogram, which maps out signal energies across frequency and time to study the time-varying behavior of a speech signal (or any sort of signal, for that matter). A spectrogram works by taking a small interval of time domain samples of the speech signal using a windowing function (e.g., hamming) and performing a short-time Fourier transform (STFT) that extracts out the frequency information about those time domain samples based on Fourier analysis. Since that STFT is just for one time instant, the window then progressively slides across time, repeating this frequency extraction repeatedly for each instant until it reaches the end of the time domain signal. The result is a three-dimensional plot, with one axis representing the time domain, another axis representing the frequency domain, and the last axis representing the signal's energy at a specific frequency and time instant. An example of a spectrogram on speech recording of “do-re-mi-fa-so-la-ti-do” is shown in Figure 2.

Q

Based on the spectrogram shown in Figure 2, approximately determine which time segments belong to the sounds “do”, “re”, “mi”, “fa”, “so”, “la”, “ti”, and “do”. One way of accomplishing this task is by looking at the frequency behavior at each time instant of the speech signal. Vowel phonemes will have well-pronounced energy bands in frequency called *formants*, while sounds like “efff” and “shhh” (i.e., fricatives) will have almost all their energy located above 4000 Hz.

In order to produce your own spectrogram in MATLAB, one can use the spectrogram function in addition to the hamming windowing function and surf plotting function. An example script to produce a spectrogram is shown in Figure 3.

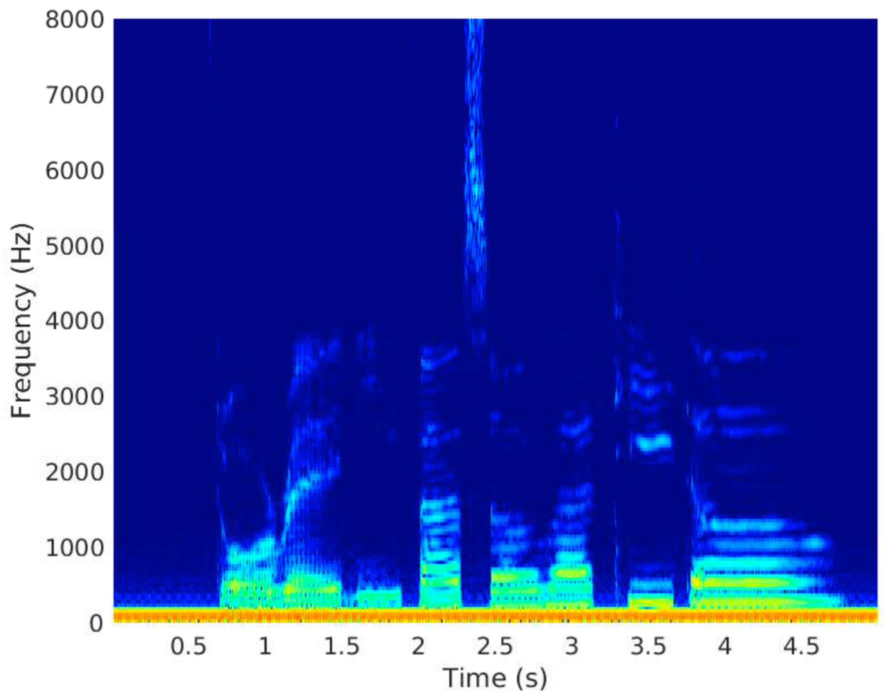

Figure 2. Spectrogram of someone saying “do-re-mi-fa-so-la-ti-do” during a 5 second time interval. Notice how the energy across the frequency domain changes for each of these sounds. This speech recording was performed using a sampling rate of Fs = 44000Hz and an 8-bit resolution

```
window = hamming(512);
N_overlap = 256;
N_fft = 1024;
[S,F,T,P] = spectrogram(speech_data,window,N_overlap,N_fft,F_s,'yaxis');
figure;
surf(T,F,10*log10(P),'edgecolor','none');
axis tight;
view(0,90);
colormap(jet);
set(gca,'clim',[-80,-20]);
ylim([0 8000]);
xlabel('Time (s)');ylabel('Frequency (Hz)');
```

Figure 3. Sample MATLAB script for producing a spectrogram of a human speech signal.

Q

Using the recorded speech signals from the previous section, namely:

- "The quick brown fox jumps over the lazy dog"
- "We promptly judged antique ivory buckles for the next prize"
- "Crazy Fredrick bought many very exquisite opal jewels"

Generate their corresponding spectrograms throughout their entire time duration and only across the first 8000 Hz of frequency. Furthermore, annotate these spectrograms with respect to highlighting any sounds, phonemes, and/or words that can be readily identifiable based on their spectral characteristics.

5. Saving and Loading WAV Files

Once the speech signals have been recorded in the MATLAB workspace environment, we often would like to save these signals to a file for processing/analysis later. Additionally, we sometimes would like to listen to and analyze a speech signal produced by another individual. Consequently, MATLAB possesses two functions, audioread and audiowrite, that allow you to save to and read from WAV files stored on your computer. A WAV file is a raw audio file format that is often used to store uncompressed lossless audio information, unlike other audio file formats such as the popular MP3 file format.

Q

Using audiowrite, store the recoeded speech signals from the previous section to three separate WAV files, namely:

- "The quick brown fox jumps over the lazy dog"
- "We promptly judged antique ivory buckles for the next prize"
- "Crazy Fredrick bought many very exquisite opal jewels"

Generate the corresponding spectrograms of these speech signals. Compare these side-by-side with your three speech signals, and highlight any similarities between the two sets of speech signals.

6. Fun with Stereo Speech Files

So far in this project, we have been dealing with mono speech files, where the left ear and right ear are receiving the exact same speech signal at the same time. However, it is also possible to record audio signals using a stereo format (e.g., configuring the audiorecorder to record in stereo). When comparing the data formats of the mono and stereo speech signals in the MATLAB workspace (generated using getaudiodata), the main difference between the two signals is that the mono speech signal consists of a single column vector containing amplitude values while the stereo speech signal consists of two equal-length column vectors containing amplitude values for the left and right channels.

Q

Using one of your previously recorded speech signals, add a second entire column vector consisting of zeros, and save this signal to a WAV file. Label this file with the convention "team[[yourteamnumber]]-stereosoundfile.wav". Also, listen to this speech file using headphones. Describe how this sounds by each member of the team.

7. Project Submission

Each student team should submit the following via the ECE2312 CANVAS website:

- A project report in PDF format that answers all the questions indicated in this project handout. Additionally, the following elements should be included:
  - Coverpage with course number and title listed, names of all student members of submitting team, date of submission, project number.
  - Descriptive captions for all figures contained within report submission.
  - Sufficiently detailed responses to all questions, providing insights about the answers pro- vided. Responses should be written in complete sentences/paragraphs, should be gram- matically correct, and written using professional vocabulary.
  - Proper pagination and formatting (e.g., 1-inch margins, single-spaced lines, 11-point serif fonts).
  - Proper pagination and formatting (e.g., 1-inch margins, single-spaced lines, 11-point serif fonts).
- Proper pagination and formatting (e.g., 1-inch margins, single-spaced lines, 11-point serif fonts).
- Link to the Github/Gitlab containing all source code generated by the student team. This code should be in a condition that it can be executed by the teaching assistant to verify its functionality. Moreover, you will run and show this code on **February 2**.