# ECE2312 – Discrete-Time Signal and System Analysis

Project 1

Due: 5.00 pm on Monday, February 12, 2024

## 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. 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

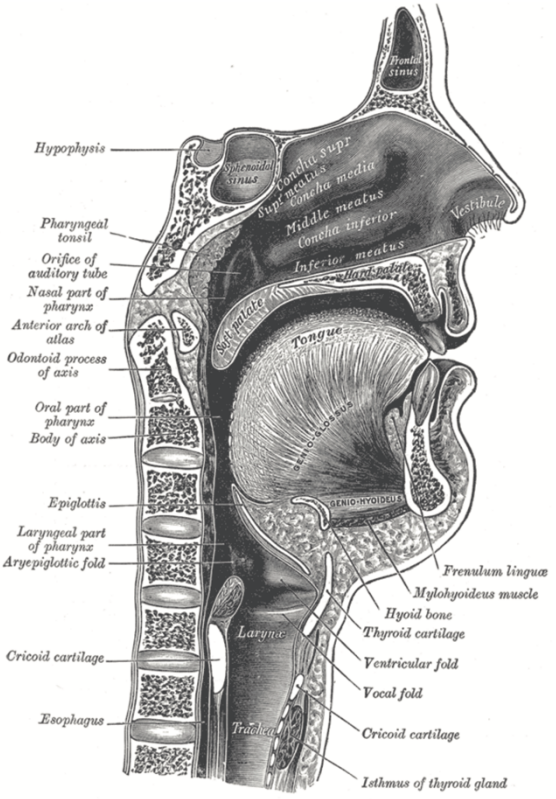


*Figure 1. Sagittal section of nose, mouth, pharynx, and larynx. From Gray's Anatomy 1918 edition*

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 using Python. 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.

You can use the sample code shown in class to record audio in Python.

If you are using 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`. 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 a universally compatible audio card and a microphone. Your operating system handles the communication between the audio hardware and user applications, which makes it easy to write programs which support any hardware. Although USB devices such as webcams might work, they are much more challenging to set up than a built-in audio device.

**Q**

Record your audio 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.

An example of the produced spectrogram is shown in Figure 2.

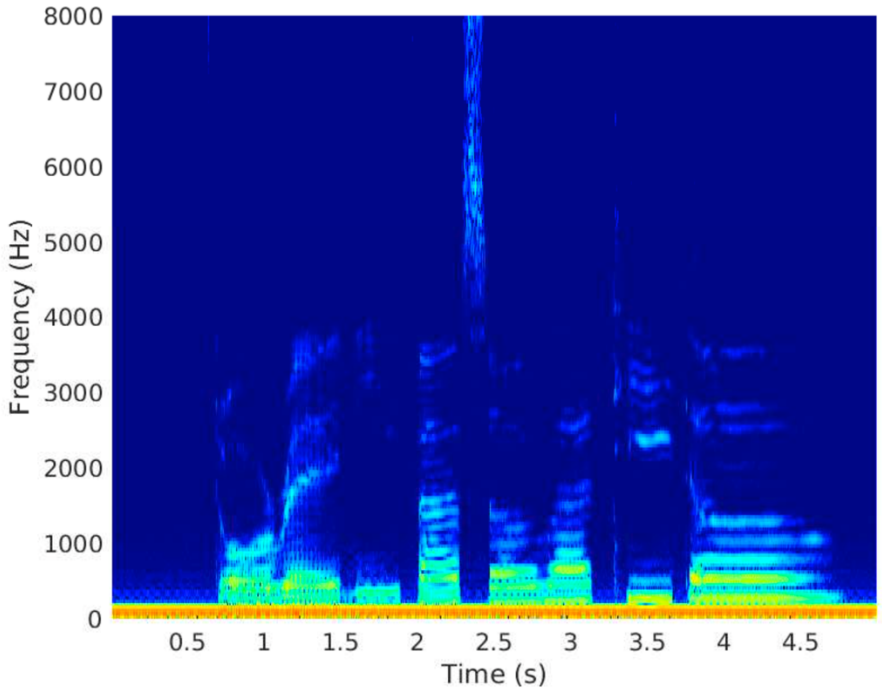


*Figure 2. Spectrogram of someone saying “do-re-mi-fa-so-la-ti-do” during a 5 second time interval. Notice how the energy across the frequency domain changes for each of these sounds. This speech recording was performed using a sampling rate of Fs = 44000Hz and an 8-bit resolution*

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 Python 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. You can look at the example code from lecture 1 to save and load audio files. 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 recorded 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 audio recorder to record in stereo). When comparing the data formats of the mono and stereo speech signals, 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.

When we hear sounds in real life, we can locate the source of a sound using the acoustic properties of our head. The most basic model of our ears is two microphones arranged along one axis and aimed away from each other. The microphones are separated by a spherical mass of a water-like density. Since sound waves propagate at a rate of about 343 m/s, there is a small but noticeable delay and attenuation from one ear to the other.

**Q** Using any measurement method, measure the absolute distance (in meters) from one ear canal to the other for each team member. Use the speed of sound to calculate the time delay (in milliseconds) between the ears. Finally, calculate the number of samples that would occur in that time delay (hint: answer will depend on the sample rate of your audio device). Calculate the team average number of samples.

We can recreate the illusion of sound sources in 3-dimensional space using vectorized operations. We will study both delay and basic attenuation to create the illusion of a sound positioned directly to the left of the listener.

Consider the following sample of a stereo audio recording (transposed as a row):

```
L [  4, -4,  2, -2,  2, -2,  4, -4,  0,  4, -4,  0,  0,  0]
R [  4, -4,  2, -2,  2, -2,  4, -4,  0,  4, -4,  0,  0,  0]
```

We can insert 3 zeros at the beginning of the right column, and remove 3 entries at the end of the right column to delay the right channel by 3 samples. This corresponds to a delay of 0.0625ms at a sample rate of 48kHz.

```
L [  4, -4,  2, -2,  2, -2,  4, -4,  0,  4, -4,  0,  0,  0]
R [  0,  0,  0,  4, -4,  2, -2,  2, -2,  4, -4,  0,  4, -4]
```

**Q** Using one of your previously recorded speech signals, copy the column to the right to create two identical columns, and save this signal to a WAV file. Label this file with the convention "team[[yourteam#]]-stereosoundfile-[[delay]].wav". Then, delay the second column by the average number of samples calculated previously, and save this signal. Finally, delay the second column by 1ms, 10ms, and 100ms, and save these. Have each team member listen to the files and compare the acoustic experience of each of these files (0ms, avghead, 1ms, 10ms, 100ms).

The second "trick" we can use is an attenuation of the right channel to model the human head. We can apply an attenuation of about -3dB by multiplying by 0.5.

```
L [  4, -4,  2, -2,  2, -2,  4, -4,  0,  4, -4,  0,  0,  0]
R [  2, -2,  1, -1,  1, -1,  2, -2,  0,  2, -2,  0,  0,  0]

L [  4, -4,  2, -2,  2, -2,  4, -4,  0,  4, -4,  0,  0,  0]
R [  0,  0,  0,  2, -2,  1, -1,  1, -1,  2, -2,  0,  2, -2]
```

**Q**

Attenuate the right channel of the 0ms delay audio by -1.5dB, -3dB, and -6dB, and save these signals to WAV files. Label these files with the convention "team[[yourteam#]]-stereosoundfile-[[delay]]-[[attenuation]].wav". Perform the same operation to the average head delay file. Have each team member listen to the files and compare the acoustic experience of each of these files (0ms, 0ms-1_5dB, 0ms-3dB, 0ms-6dB, avghead, avghead-1_5dB, …).

**Q**

Which delay and attenuation values produced the most convincing illusion that the sound source is to the left of the listener? How might you improve the basic model to improve the illusion or support sounds from other locations in 3D space?

## 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:
  - Cover page with course number and title listed, names of all student members of submitting team, date of submission, project number, and Github Link.
  - Descriptive captions for all figures contained within report submission. Figures should be labeled and referenced in text (e.g. "As shown in Figure 1")
  - Sufficiently detailed responses to all questions, providing insights about the answers provided. Responses should be written in complete sentences/paragraphs, should be grammatically correct, and written using professional vocabulary.
- 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 executable by the teaching assistant to verify its functionality.