EE438 - Laboratory 9: Speech Processing

June 11, 2004

1 Introduction

Speech is an acoustic waveform that conveys information from a speaker to a listener. Given the importance of this form of communication, it is no surprise that many applications of signal processing have been developed to manipulate speech signals. Almost all speech processing applications fall into three broad categories: speech recognition, speech synthesis, and speech coding.

Speech recognition may be concerned with the identification of certain words, or with the identification of the speaker. Automatic speech recognition systems attempt to recognize a continuous sequence of word utterances, possibly to convert into text within a word processor. Anybody who has made a collect phone call in the past few years has used a system that recognizes vocal commands to determine its next action. Speaker identification is useful in security applications, as a person's voice is much like a "fingerprint".

The objective in speech synthesis is to convert a string of text, or a sequence of words, into natural-sounding speech. This is used in speech production systems that allow people who cannot speak to better communicate. Another application is a system that reads text for the blind. Speech synthesis has also been used to aid scientists in learning about the mechanisms of human speech production, and thereby in the treatment of speech-related disorders.

Speech coding is mainly concerned with exploiting the redundancy of certain vocal sounds, allowing the speech to be represented in a digitally compressed form. Research in speech compression and transmission has been motivated by the need to conserve bandwidth in communication systems. For example, speech coding is used to reduce the bit rate in digital cellular systems.

Applications of speech processing rely on a detailed understanding of the properties of the many different vocal sounds. The objective of this lab is to identify some of these properties, and introduce some elementary aspects of speech processing.

Questions or comments concerning this laboratory should be directed to Prof. Charles A. Bouman, School of Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907; (765) 494-0340; bouman@ecn.purdue.edu

2 Time Domain Analysis of Speech Signals

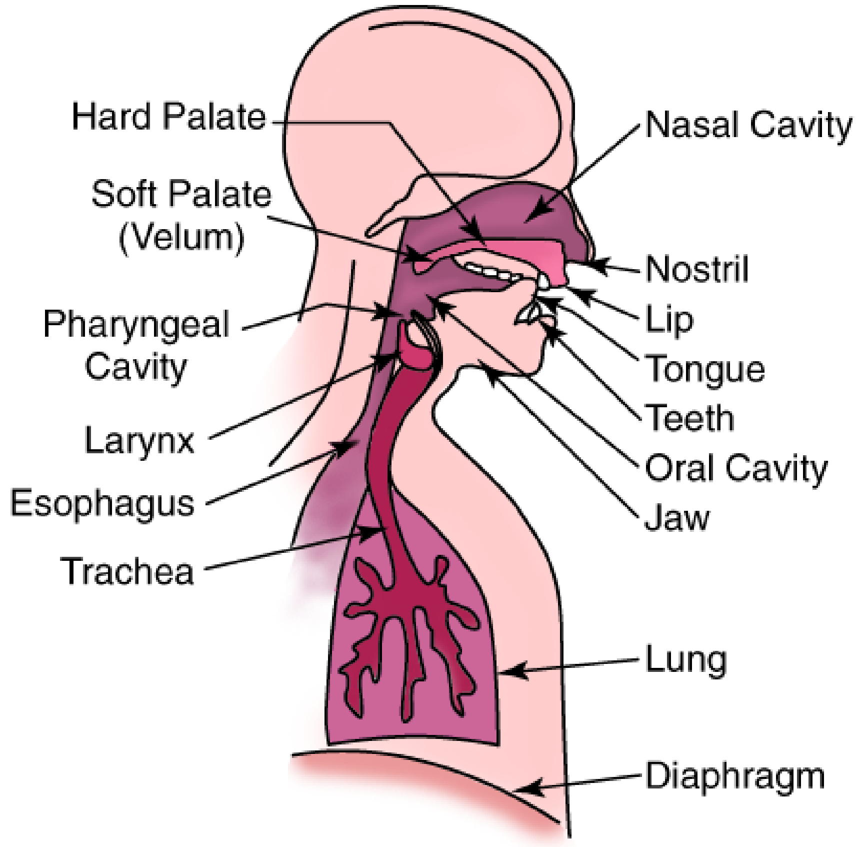


Figure 1: The Human Speech Production System

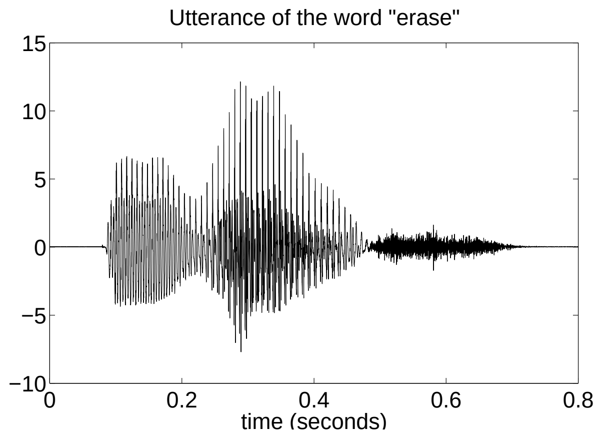
2.1 Speech Production

Speech consists of acoustic pressure waves created by the voluntary movements of anatomical structures in the human speech production system, shown in Figure 1. As the diaphragm forces air through the system, these structures are able to generate and shape a wide variety of waveforms. These waveforms can be broadly categorized into *voiced* and *unvoiced speech*.

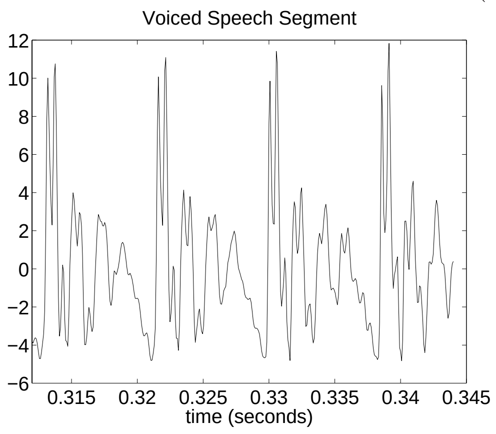
Voiced sounds, vowels for example, are produced by forcing air through the larynx, with the tension of the vocal cords adjusted so that they vibrate in a relaxed oscillation. This produces quasi-periodic pulses of air which are acoustically filtered as they propagate through the vocal tract and possibly the nasal cavity. The shape of the cavities that comprise the vocal tract, known as the *area function* of the vocal tract, determines natural frequencies, or *formants*, that are emphasized in the speech waveform. The period of the excitation, known as the *pitch period*, is generally small with respect to the rate that the vocal tract changes shape. Therefore, a segment of voiced speech covering several pitch periods will appear somewhat *periodic*. Typical values for the pitch period are 8 milliseconds (ms) for male speakers, and 4 ms for female speakers.

In contrast, unvoiced speech has more of a noise-like quality. It is usually smaller in amplitude, and oscillates much faster than voiced speech. These sounds are generally produced by turbulence, as air is forced through a constriction at some point in the vocal tract. Consequently, there are a number of different types of unvoiced sounds that can be generated.

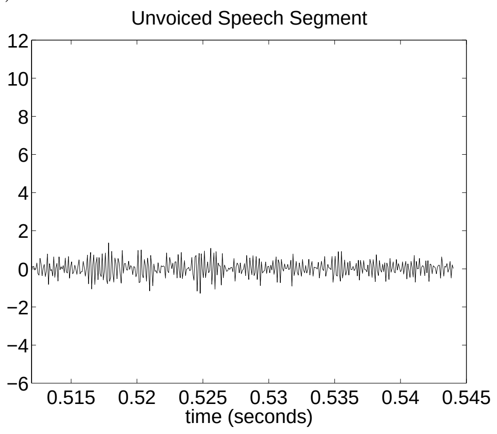
An illustrative example of voiced and unvoiced sounds contained in the word "erase" are shown in Figure 2. The original utterance is shown in (a). The voiced segment in (b) is a time magnification of the "a" portion of the word. Notice the highly periodic nature of this segment. The fundamental period of this waveform, which is about 8.5 ms here, is what we call the *pitch period*. The unvoiced segment in (c) comes from the "s" sound at the end of the word. This waveform is much "noisier" than the voiced segment, and is much smaller in magnitude.



(a)



(b)



(c)

Figure 2: (a) Utterance of the word "erase". (b) Voiced segment. (c) Unvoiced segment.

2.2 Classification of Voiced or Unvoiced Speech

Down load **start.au**
How to **load and play audio signals**

For many methods of speech recognition, a very important step is to determine the type of sound that is being uttered in a given time frame. In this section, we will introduce two simple methods for discriminating between voiced and unvoiced speech.

Down load the utterance **start.au** , and use the `auread()` function to load it into the Matlab workspace.

Do the following:

- Plot (not stem) the speech signal. Identify two segments of the signal: one segment that is voiced and a second segment that is unvoiced. The Matlab command `zoom xon` is useful for this. Circle the regions of the plot corresponding to these two segments and label them as voiced or unvoiced.
- Save 300 samples from the voiced segment of the speech into a Matlab vector called *VoicedSig*.
- Save 300 samples from the unvoiced segment of the speech into a Matlab vector called *UnvoicedSig*.
- Use the `subplot()` command to plot the two signals, *VoicedSig* and *UnvoicedSig* on a single figure.

INLAB REPORT:
Hand in your labeled plots. Explain how you selected your voiced and unvoiced regions.

Estimate the pitch period for the voiced segment. Keep in mind that these speech signals are sampled at 8 KHz, which means that the time between samples is 0.125 milliseconds (ms). Typical values for the pitch period are 8 ms for male speakers, and 4 ms for female speakers. Based on this, would you predict that the speaker is male, or female?

One way segments may be categorized in an algorithm is by computing the average power of the signal within a frame. Remember that this is defined by the following:

$$P_{AV} = \frac{1}{L}\sum_{n=1}^{L} x^2(n) \qquad (1)$$

where L is the length of the frame $x(n)$. Compute the average power of the voiced and unvoiced segments that you plotted above. For which segment is the average power greater?

Another method for discriminating between voiced and unvoiced segments is to determine the rate at which the waveform oscillates by counting number of zero-crossings that occur within a frame. Write a function that will compute the number of zero-crossings that occur within a vector, and apply this to the two vectors *VoicedSig* and *UnvoicedSig*. Which segment has more zero-crossings?

INLAB REPORT:
Give your estimate of the pitch period for the voiced segment, and your prediction of the gender of the speaker. For each of the two vectors, *VoicedSig* and *UnvoicedSig*, list the average power and number of zero-crossings. Which segment has a greater average power? Which segment has a greater zero-crossing rate?

2.3 Phonemes

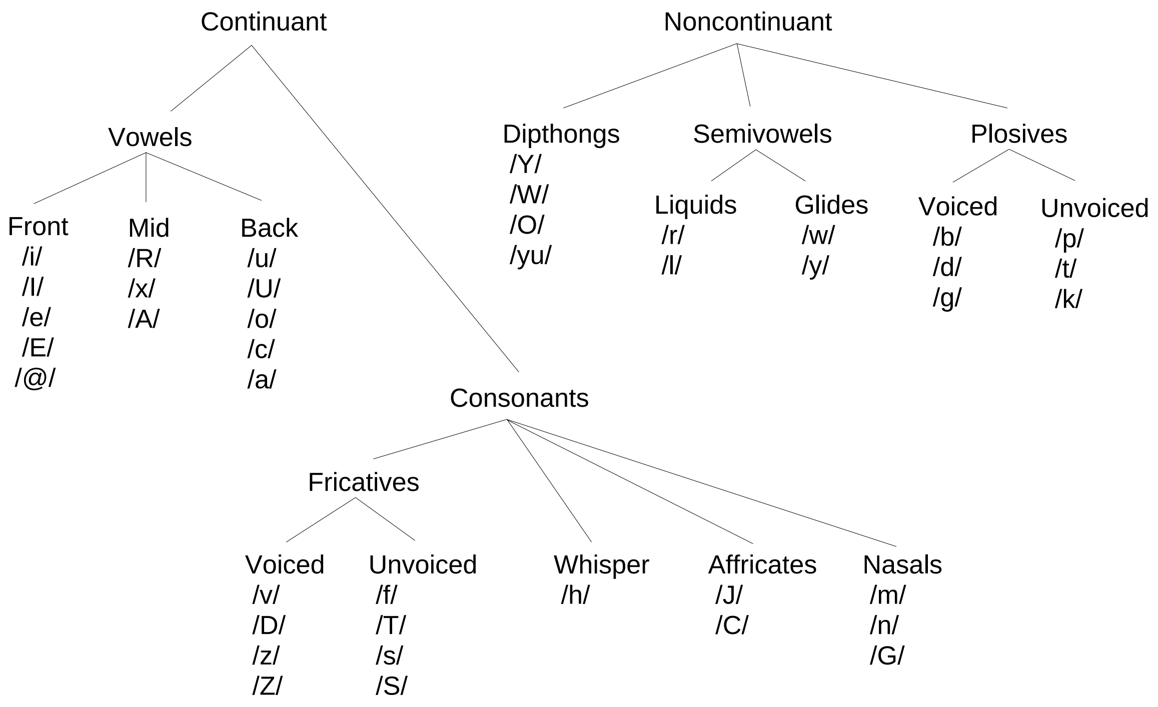


Figure 3: Phonemes in American English. See [1] for more details.

American English can be described in terms of a set of about 42 distinctive sounds called *phonemes*, illustrated in Figure 3. They can be classified in many ways according to their distinguishing properties. *Vowels* are formed by exciting a fixed vocal tract with quasi-periodic pulses of air. *Fricatives* are produced by forcing air through a constriction (usually towards the mouth end of the vocal tract), causing turbulence. These may be voiced or unvoiced. *Plosive* sounds are created by making a complete closure, typically at the frontal vocal tract, building up pressure behind the closure and abruptly releasing it. A *diphthong* is a gliding monosyllabic sound that starts at or near the articulatory position for one vowel, and moves toward the position of another. Try reciting several of the phonemes shown in Figure 3, and make a note of the movements you are making to create them.

2.4 Simple Speech Model

Down load **coeff.mat**

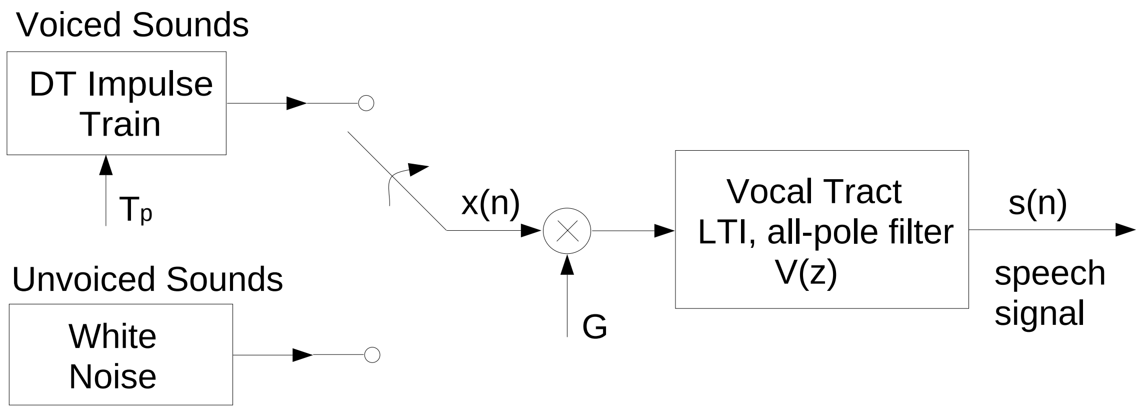


Figure 4: Discrete-Time Speech Production Model

From a signal processing standpoint, it is very useful to think of speech production in terms of a model, as in Figure 4. The model shown is the simplest of its kind, but it contains the major components that are involved. The impulse train is a discrete-time representation for periodic pulses of air, which act as the excitation for voiced speech. The spacing between each impulse is the pitch period, T_p. The excitation for unvoiced sounds can be thought of as a white noise generator. The speech signal, $s[n]$, is generated by running the excitation, $e[n]$, through an all-pole filter with the transfer function $G(z)$. Keep in mind that as speech is produced, the pitch period and filter parameters may change continuously, but speech segments of an appropriate length can be put in terms of a stationary system model.

It would seem that the easiest way to create a system that generates speech would be to store the words and call them as needed. However, for a significant vocabulary this quickly becomes unfeasible because of the memory limitations. An alternative to this is to use a model like Figure 4. The model parameters for the various speech sounds can be stored, and words can then be constructed piece-by-piece. We will demonstrate this shortly by synthesizing vowel sounds.

The transfer function of an all-pole filter can be written as

$$H(z) = \frac{1}{1 - \sum_{k=1}^{P} a_k z^{-k}} \tag{2}$$

where P is the order of the filter. This is an IIR filter that can easily be implemented with a recursive difference equation, as long as the a_k parameters are known.

Download the file **coeff.mat** and load it into the Matlab workspace by typing `load coeff`. This will load three sets of coefficients, *A1* through *A3*, for the transfer function in (2). Each set is for a filter of order 15.

To produce the sounds, we need an excitation. Create a discrete-time periodic impulse train with a pitch period of 8 ms, and a duration of one second. This will be a vector of "1"s, each separated by several zeros. Remember that the sampling frequency of our hardware is 8 KHz, which means that each sample of an audio signal corresponds to 0.125 ms.

Now filter the excitation with each set of parameters. Use the Matlab command `filter(1,A,e)` where A is the vector of coefficients, and e is your excitation signal. Try playing them using `soundsc()` or `auplay()` (if `auplay()` is used, you may need to scale the signal down to prevent clipping). For each signal, identify which vowel is being synthesized.

For each vowel signal, plot 5 pitch periods starting from the 500th sample. Use `subplot()` and `orient tall` to plot them in the same figure.

Next compute the frequency response of each filter you just implemented. This can easily be obtained using the Matlab command `[H,W]=freqz(1,A,512)`, where A is the vector of coefficients. Plot the magnitude of each frequency response versus frequency in Hertz. Use `subplot()` and `orient tall` to plot them in the same figure.

The location of the peaks in the spectrum correspond to the formant frequencies. For each vowel signal, estimate the center frequency of the first three formants.

INLAB REPORT:
Hand in the following:

- A figure containing plots of the three vowel signals. Label each subplot with the vowel that you identified for the signal.
- A plot of the frequency response for the three filters. Plot the spectrum on a linear scale and label the frequency axis in units of Hertz.
- For each of the three filters, list the approximate center frequency of the first three formant peaks.

3 Short-Term Frequency Analysis

As we have seen from previous sections, the properties of speech signals are continuously changing, but may be considered to be stationary within an appropriate time frame. If analysis is performed on a "segment-by-segment" basis, useful information about the construction of an utterance may be obtained. The average power and zero-crossing rate, as previously discussed, are examples of short-term feature extraction in the time-domain. In this section, we will learn how to obtain short-term frequency information from generally non-stationary signals.

3.1 stDTFT

Down load **go.au**

A useful tool for analyzing the spectral characteristics of a non-stationary signal is the *short-term discrete-time Fourier Transform*, or *stDTFT*, which we will define by the following:

$$X_m(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)w(n-m)e^{-j\omega n} \tag{3}$$

Here, $x[n]$ is our speech signal, and $w[n]$ is a window of length L. Notice that if we fix m, the stDTFT is simply the DTFT of x[n] multiplied by a shifted window. Therefore, $X_m(e^{j\omega})$ is a collection of DTFTs of windowed segments of x[n].

As we examined in **Lab 5** , windowing in the time domain will cause an undesirable ringing in the frequency domain. This effect can be reduced by using some form of a raised cosine for the window $w[n]$.

Write a function `X = DFTwin(x,L,m,N)` that will compute the DFT of a length L segment of the vector x.

- You should use a Hamming window of length L to window x.
- Your window should start at at the index m.
- Your DFTs should be of length N.
- You may use Matlab's `fft()` algorithm to compute the DFTs.

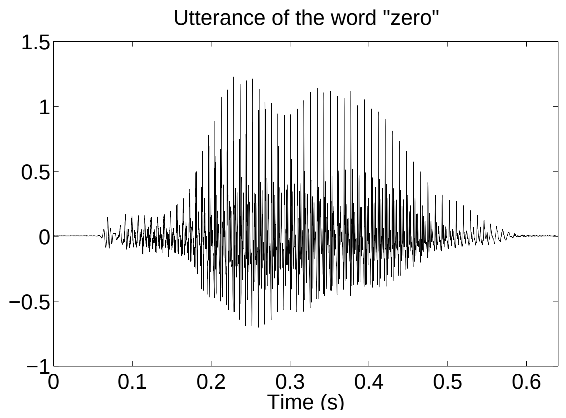
Now we will test your `DFTwin()` function. Down load **go.au** , and load it into Matlab. Plot the signal and select a voiced region. Use your function to compute a 512-point DFT of a window that will cover six pitch periods of this region. `Subplot` your chosen segment and the DFT magnitude (for ω from 0 to π) in the same figure. Label the frequency axis in Hz, assuming a sampling frequency of 8 KHz. Remember from the sampling theorem that a radial frequency of π corresponds to half the sampling frequency.

INLAB REPORT:
Hand in the code for your DFTwin() function, and your plot. Describe the general shape of the spectrum, and estimate the formant frequencies for the region of voiced speech.

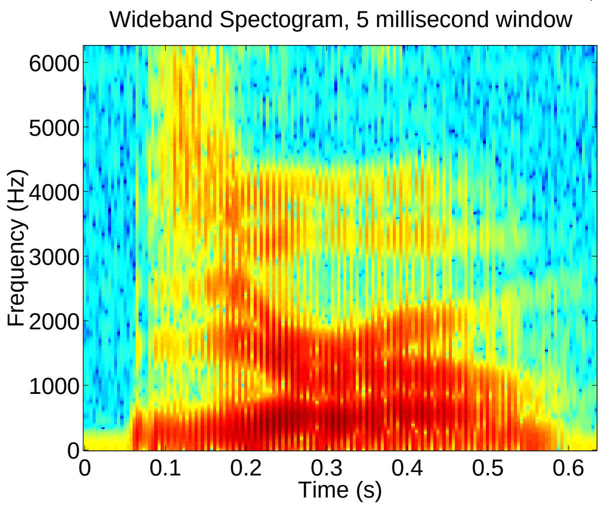
3.2 The Spectogram

Down load **signal.mat**

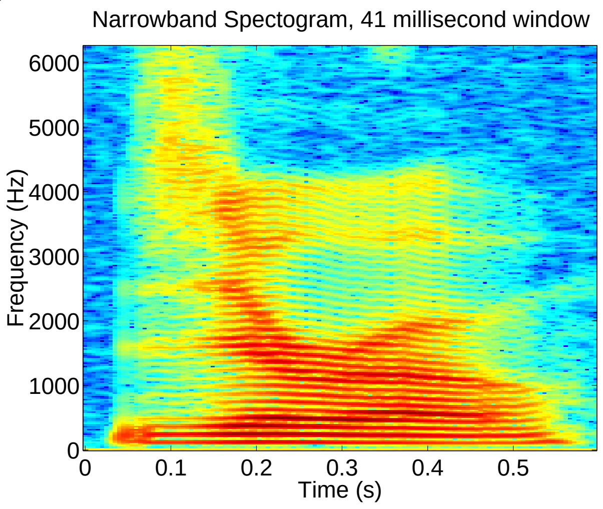
As previously stated, the short-term DTFT is a collection of DTFTs that differ by the position of the truncating window. These functions may be oriented in an image, called a *spectogram*, to give insight on how the spectral characteristics of the signal evolve with time. The spectogram is created by placing the DTFTs vertically in the image for different time segments, such that time increases from left to right, and frequency increases from bottom to top. The magnitude of the DTFT at each point is proportional to the intensity of that point in the image, allowing one to see the spectrum of segments of a signal at each time instant. A spectogram may also use a "pseudo-color" mapping, which uses a spectrum of colors to indicate the magnitude of the frequency content, as shown in Figure 5.



(a)



(b)



(c)

Figure 5: (a) Utterance of the word "zero". (b) Wideband Spectogram. (c) Narrowband Spectogram.

For quasi-periodic signals like speech, spectograms are placed into two categories according to the length of the truncating window. *Wideband* spectograms use a window with a length comparable to a single period. This yields high resolution in the time domain but low resolution in the frequency domain. These are usually characterized by vertical striations, which correspond to high and low energy regions within a single period of the waveform. In *narrowband* spectograms, the window is made long enough to capture several periods of the waveform. Here, the resolution in time is sacrificed to give a higher resolution of the spectral content. Harmonics of the fundamental frequency of the signal are resolved, and can be seen as horizontal striations. Care should be taken to keep the window short enough, such that the signal properties stay relatively constant within the window.

When computing spectograms, not every possible window position is used from the stDTFT, as this would result in mostly redundant information. Successive windows will generally start many samples apart, usually thought of in terms of the overlap between the windows. Criteria in deciding the amount of overlap includes the length of the window, the desired resolution in time, and the rate at which the signal characteristics are changing with time.

Given this background, we would now like you to create a spectogram using your `DFTwin()` function from the previous section. You will do this by creating a matrix of windowed DFTs, oriented as described above. Your function should be of the form `A = Specgm(x,L,overlap,N)`, where x is your input signal, L is the window length, *overlap* is the number of points common to successive windows, and N is the number of points you compute in each DFT. Within your function, you should plot the magnitude (in dB) of your spectogram matrix using the command `imagesc()`, and label the time and frequency axes.

Important Hints:

- Remember that frequency in a spectogram increases along the positive y-axis, which means that the first few elements of each column of the matrix will correspond to the highest frequencies.
- Your `DFTwin()` function returns the DT spectrum for frequencies between 0 and 2π. Therefore, you will only need to use the first or second half of these DFTs.
- The statement `B(:,n)` references the entire n^{th} column of the matrix B.
- In labeling the axes of the image, assume a sampling frequency of 8 KHz. Then the frequency will range from 0 to 4000 Hz.
- The `axis xy` command will be needed in order to place the origin of your plot in the lower left corner.
- You can get a pseudo-color image by using the command `colormap(jet)`.

Down load **signal.mat** , and load it into Matlab. This is a raised square wave that is modulated by a sinusoid. What would the spectrum of this signal look like? Create a both a wideband and narrowband spectogram using your `Specgm()` function for the signal.

- For the wideband spectogram, use a window length of 40 samples and an overlap of 20 samples.
- For the narrowband spectogram, use a window length of 320 samples, and an overlap of 60 samples.

Subplot the wideband and narrowband spectograms, and the original signal in the same figure.

INLAB REPORT:
Hand in your code for Specgm() and your plots. Do you see vertical striations in the wideband spectogram? Similarly, do you see horizontal striations in the narrowband spectogram? In each case, what causes these lines, and what does the spacing between them represent?

3.3 Formant Analysis

Down load **vowels.mat**

The shape of an acoustic excitation for voiced speech is very similar to a triangle wave. Therefore it has many harmonics at multiples of its fundamental frequency, $1/T_p$. As the excitation propagates through the vocal tract, acoustic resonances, or standing waves, cause certain harmonics to be significantly amplified. The specific wavelengths, hence the frequencies, of the resonances are determined by the shape of the cavities that comprise the vocal tract. Different vowel sounds are distinguished by unique sets of these resonances, or $formantfrequencies$. The first three average formants for several vowels are given in Figure 6.

A possible technique for speech recognition would be the determination of a vowel utterance based its unique set of formant frequencies. If we construct a graph that plots the second formant versus the first, we find that a particular vowel sound tends to lie within a certain region of the plane. Therefore, if we determine the first two formants, we can construct decision regions to estimate which vowel was spoken. The first two average formants for some common vowels are plotted in Figure 7. This diagram is known as the *vowel triangle* due to the general orientation of the average points.

Keep in mind that there is a continuous range of vowel sounds that can be produced by a speaker. When vowels are used in speech, their formants almost always slide from one position to another.

Download the file **vowels.mat** , and load it into Matlab. This file contains the vowel utterances *a*,*e*,*i*,*o*, and *u* from a female speaker. Load this into Matlab, and plot a narrowband spectogram of each of the utterances. Notice how the formant frequencies change with time.

For the vowels *a* and *u*, estimate the first two formant frequencies using the functions you created in the previous sections. Make your estimates at the beginning and end of each

Formant Frequencies for the Vowels				
Typewritten Symbol for the Vowel	Typical Word	F1 (Hz)	F2 (Hz)	F3 (Hz)
IY	(beet)	270	2290	3010
I	(bit)	390	1990	2550
E	(bet)	530	1840	2480
AE	(bat)	660	1720	2410
UH	(but)	520	1190	2390
A	(hot)	730	1090	2440
OW	(bought)	570	840	2410
U	(foot)	440	1020	2240
OO	(boot)	300	870	2240
ER	(bird)	490	1350	1690

Figure 6: Average Formant Frequencies for the Vowels

utterance, and plot them in the vowel triangle provided in Figure 7. You may want to use both the *Specgm* and *DFTwin* functions to determine the formants. For each vowel, draw a line connecting the two points, and draw an arrow indicating the direction the formants are changing.

INLAB REPORT:
Hand in your formant estimates on the vowel triangle.

References

[1] J. R. Deller, Jr., J. G. Proakis, J. H. Hansen, *Discrete-Time Processing of Speech Signals,* Macmillan, New York, 1993.

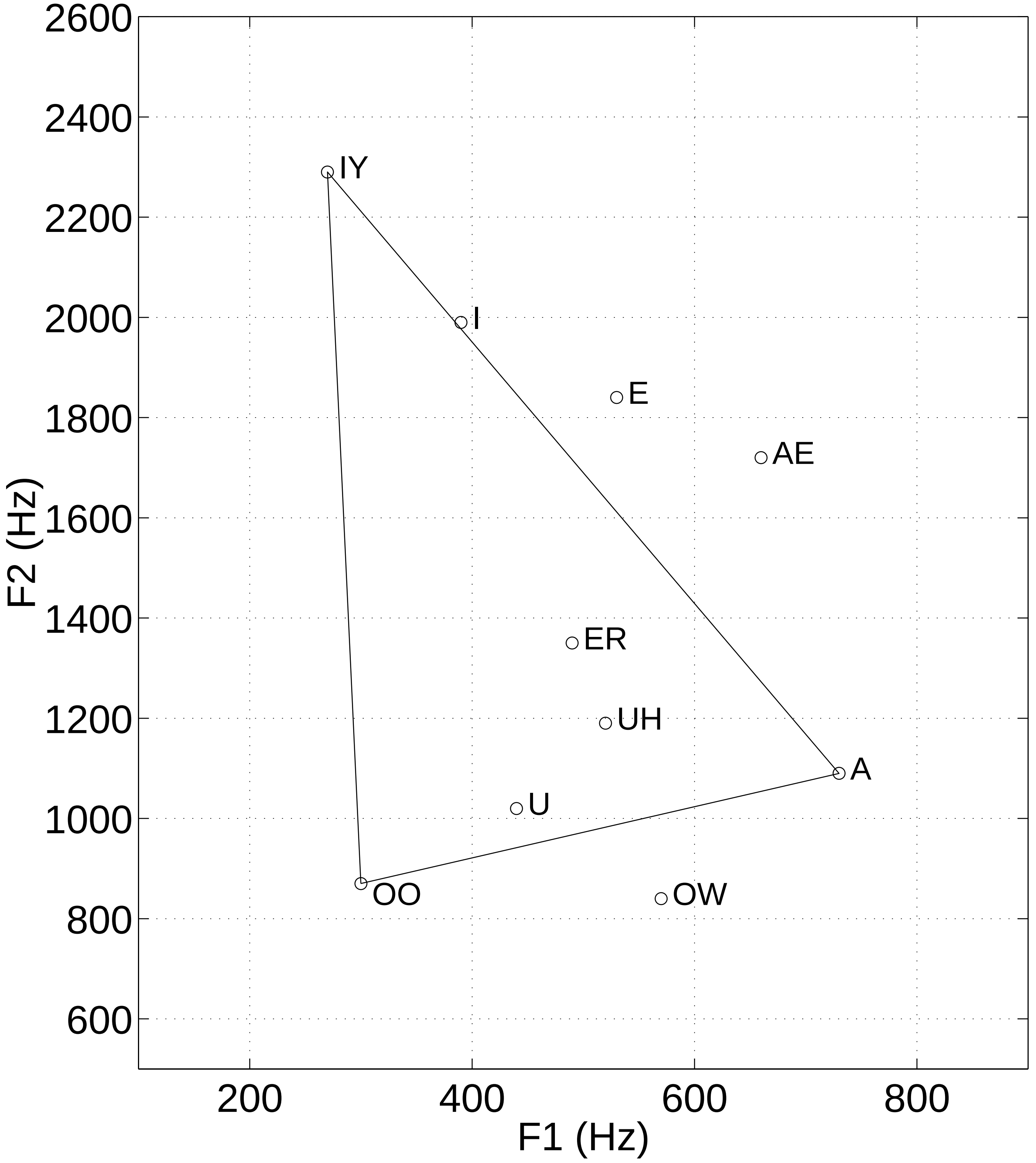


Figure 7: The Vowel Triangle