Speech Coding and Phoneme Classification Using MATLAB and NeuralWorks

Brett A. St. George, Ellen C. Wooten, and Louiza Sellami
Department of Electrical Engineering
U.S. Naval Academy
Annapolis, MD 21402

Abstract - Applications involving speech coding and phonetic classification are introduced as educational tools for reinforcing signal processing concepts learned in senior level communication classes at the U.S. Naval Academy. These applications utilize the software packages MATLAB* and NeuralWorks* and are used here to explore the concepts of impulse sampling, Fourier transforms, data windowing, and homomorphic filtering.

Introduction

In 1963 the Naval Academy instituted the Trident Scholar undergraduate study program to provide a select number of exceptionally capable students the opportunity to perform independent research during their senior year. As a Trident Scholar, I used MATLAB and NeuralWorks software to create a method for identifying different voice patterns within my own speech signal. Many of the MATLAB m-files that were written could be instituted as labs that could teach students the properties of the Fast Fourier Transform (FFT) while at the same time presenting different windowing and filtering techniques. Structuring a course around such a sequence of labs would allow students to develop a system for performing speech coding and phoneme classification while reinforcing many digital signal processing concepts learned in the classroom. Because speech coding involves a process of extracting information from an analog, time-based signal, and improving the efficiency of speech recognition, many of the techniques and algorithms employed in the process inherently involve digital filtering, data windowing, and spectral analysis.

In this paper, several experiments with an analog speech signal are presented that could be instituted as labs to reinforce communication theory that students are receiving in the classroom. Specifically, these experiments allow students to extract the pitch frequency of their voice as well as those frequencies that add constructively within the oral cavity. These frequencies are known as formants and can be used to classify voice data using a phonetic alphabet comprised of 40 distinct sounds called phonemes. In these labs many of the important properties intrinsic to the FFT and discrete sampling are introduced. For example, the Hermitian symmetry of the amplitude spectrum would become evident in experiments with various padding lengths and data windows. Further properties of the FFT as well as various filtering routines could be instituted as labs to supplement digital signal processing theory.

Speech Coding

The flow chart depicted in Fig. 1 maps the method of our speech coding process and classification algorithm. To understand the speech coding process, it is necessary to begin with a description of the physical nature of speech. Sound is produced when air is forced from the lungs and becomes filtered by variations in the vocal tract shape to produce a speech signal [1, p. 53]. These variations in shape determine the characteristics of the filtering function that shape the frequency spectrum of the final speech signal. If this filtering function can be directly extracted from a sampled speech signal, it can be used to identify which phonetic character is being pronounced.

To begin the speech coding process, suppose that the vocal tract acts as a linear time-invariant system within a

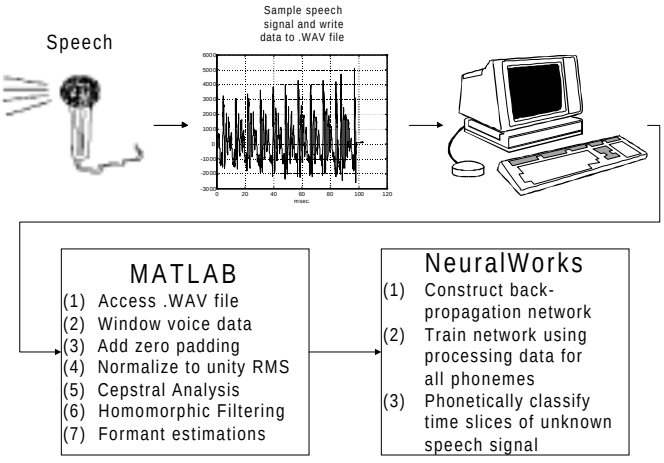

Fig. 1 Flowchart of speech coding and classification algorithm

*MATLAB is a trademark of The MathWorks, Inc.
NeuralWorks is a trademark of The NeuralWare, Inc.

sufficiently short time slice. This is a valid assumption for a short segment of speech where the vocal tract shape is unchanging and remains fixed in shape. Hence, the frequency representation of a speech segment, F(jω), can be represented as the product of an excitation source, E(jω), and a transfer function, H(jω):

$$F(j\omega) = E(j\omega)\cdot H(j\omega) \quad (1)$$

To separate these two functions, the complex logarithm of Eq. 1 is used to create a real and imaginary part that reflect the magnitude and phase angle respectively.

$$Ln(F(j\omega)) = Ln|F(j\omega)| + j\theta_{F(j\omega)} \quad (2)$$

Because phase angle carries only information about the time origin of the signal, the imaginary part of Eq. 2 can be ignored [2, p. 376]. The real values, however, represent a magnitude function where the excitation source and transfer function are additive in the frequency domain [3, p. 266].

$$Ln|F(j\omega)| = Ln|E(j\omega)| + Ln|H(j\omega)| \quad (3)$$

Taking the Inverse Fast Fourier Transform (IFFT) of the real portion of the complex logarithm, Eq. 3, produces a time domain signal in which the logarithm of the excitation source and impulse response are separable. This filtering process is known as homomorphic deconvolution and involves a technique referred to as "cepstral deconvolution." To employ this technique, the combined signals need to have their main components of energy concentrated at different frequencies [3, p. 266]. This condition is true of speech when only the transfer function and excitation source are considered. The transfer function describing the vocal tract, H(jω), has a band-limiting characteristic that confines the energy of the speech signal within a 5 kHz range. Conversely, the excitation source, E(jω), can be modeled as a white noise source which contains an equal distribution of energy across a frequency range far in excess of 5 kHz. Because the primary energy components of both functions do not overlap, cepstral analysis applies. This technique allows students to readily understand how the properties of logarithms can be used to remove unwanted noise from a frequency banded signal.

These algorithmic processes are fundamental to speech coding and are easily implemented in MATLAB as demonstrated in the next section. The process culminates when the formant frequencies derived from a segment of speech are coded into a training set vector in NeuralWorks that correlates the data to a particular phoneme. A simulation using the phoneme "e" as in "bet" is now introduced.

MATLAB and NeuralWorks Simulation

By analyzing the periodicity of time slices of a speech signal, segments can be classified as voiced or unvoiced. During voiced speech the vocal chords vibrate at a constant frequency known as the fundamental or pitch frequency. Alternatively, in unvoiced speech the vocal chords do not move and air is forced past the glottis, tongue, teeth and lips [1, pp. 36]. These characteristics become apparent when students view their speech as a MATLAB plot.

In Fig. 2, the periodicity of the speech signal can be observed for the voiced phoneme “e”. This signal was sampled using a 16-bit analog-to-digital converter and was then stored as a binary WAV file. The sampling rate (22.05 kHz) and storage of data was controlled by a soundcard driver. The speech signal in Fig. 2 represents the sampled

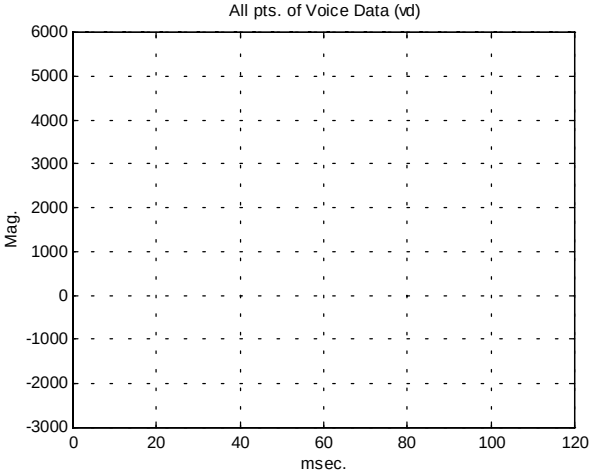

Fig. 2 Voice data for the phoneme "e" as in "bet"

points that are passed as a data vector into MATLAB.

Within MATLAB, the voice data is segmented into 30 msec time slices/records and fitted to various data windows

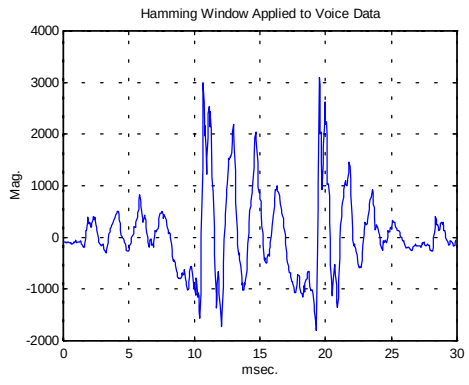

Fig. 3 Time slice of the phoneme "e" with Hamming window applied

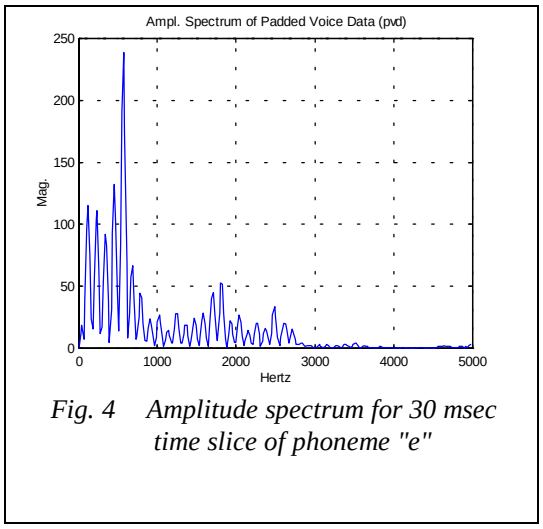

Fig. 4 Amplitude spectrum for 30 msec time slice of phoneme "e"

(Fig. 3). The windowed data is then padded with zeros to a power of 2, normalized to unity RMS, and transformed using the radix-2 FFT (Fig. 4). This process accounts for any variations in the intensity of the voice and adds resolution to the frequency domain output. These aspects provide students an opportunity to experiment with the properties of the FFT while observing the results created by applying various windowing functions to the padded voice data. Throughout this experimental process, students are forced to consider any masking effects that these signal processing techniques might impose on the output. To insure that these techniques are being implemented correctly, various checks are available. The first of these is the computation of the pitch frequency and involves a technique called "cepstral" analysis. In this process, a second Fourier analysis is applied to the logarithm of the frequency spectrum (Fig. 4). The resultant function is called the "cepstrum" and contains a spike at each harmonic of the pitch period [2, pp. 362]. The time domain function in Fig.

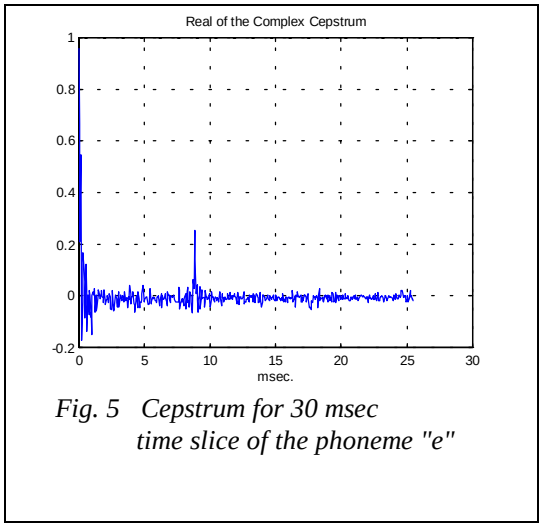

Fig. 5 Cepstrum for 30 msec time slice of the phoneme "e"

5 shows an initial spike at 8 msec. This spike corresponds to a pitch of 125 Hz. Students can compare the periodicity of the waveform presented in figure 2 with the pitch period computed using cepstral analysis as shown in figure 5 and verify the validity of this analysis technique.

Once in the cepstral domain, a time-window applied to the real part separates the impulse response from the excitation source, thus revealing the formant frequencies (Fig. 6) that cause the vocal tract to resonate. As pictured in figure 6, these formant frequencies appear as peaks and

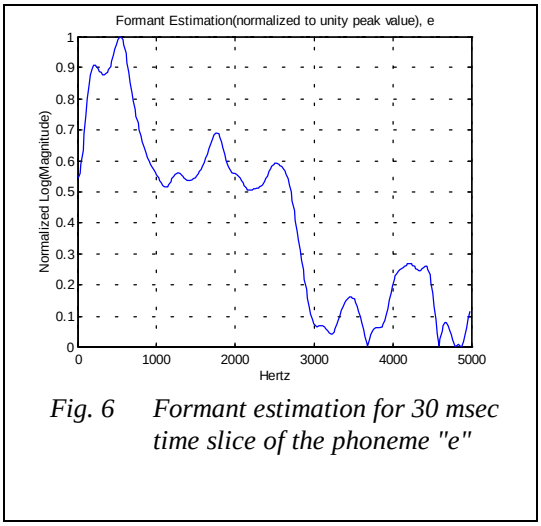

Fig. 6 Formant estimation for 30 msec time slice of the phoneme "e"

can be masked if the side-band leakage is too great or the resolution is poor. From such plots students can judge the effectiveness of their choice of padding length and data window. With these frequency characteristics resolved, a neural network can be used to identify different phonetic pronunciations [1, pp. 48].

The formant frequencies derived by this technique are used to create a set of vectors that is used to train a back-propagation neural network to recognize different phonemes. Because the vocal tract is very defined for the pronunciation of different phonemes but changes slowly with time, the formant frequencies reveal more information than is presented by the time domain analysis of the original speech waveform. In NeuralWorks students are able to observe the global error function converging to zero. This function is proportional to the square of the Euclidean distance between the desired output and the actual output of the network for a particular input pattern [4, pp. 69]. The error function is displayed as a strip chart in Fig. 7. In addition, a graphical matrix that quantifies the classification rate of the network is also displayed. This rate describes the ability of the network to correctly classify the training and test data. By observing these parameters and their convergence, students are able to judge the effectiveness of the data windowing and homomorphic filtering processes.

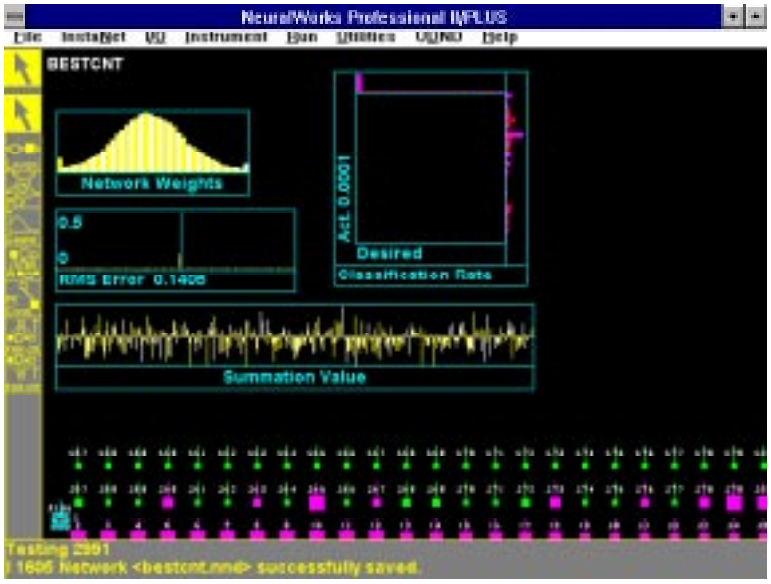

Fig. 7 NeuralWorks display of a back-propagation neural network trained to phonetically classify time slices of a speech signal. The network consist of 255 inputs that represent the frequency response of the vocal tract. The hidden layer contains 400 processing elements and is fully connected to an output layer of 40 elements. These 40 elements each correspond to one phoneme.

Each of the routines described in the preceding discussion is implemented as either a programming code in MATLAB or as a design file in NeuralWorks. Through this process, students gain better insight into signal processing theory by applying speech coding techniques to sampled voice data.

Conclusions

In this paper, speech coding and classification techniques have been explored via MATLAB and NeuralWorks. These software tools allowed us to sample an analog speech signal, find the pitch and formant frequencies, and phonetically classify voice data. The speech coding algorithms used here involve digital filtering, data windowing, and spectral analysis. This particular application provided the means of some of the aspects of diverse signal processing theory in a graphical and procedural manner. Each of the techniques introduced in this paper can be implemented as a lab to support signal processing theory learned in the classroom. Through such a sequence of labs, students learn the properties of the FFT while discovering the trade-offs of various data windowing techniques and filtering routines.

References

1. Pelton, Gordon, *Voice Processing*, McGraw-Hill, 1993.
2. Rabiner, Lawrence and Schafer, Ronald, *Digital Processing of Speech Signals*, Prentice-Hall, 1978.
3. John Proakis and Dimitris Manolakis, *Digital Signal Processing 3rd Edition*, 1996.
4. NeuralWare, Inc., *Neural Computing*, NeuralWare, 1995.