MATLAB EXERCISES IN SUPPORT OF TEACHING DIGITAL SPEECH PROCESSING

Lawrence R. Rabiner[*]

Rutgers University
Dept. of Electrical and Computer Engineering
New Brunswick, NJ 08854

Ronald W. Schafer

Stanford University
Dept. of Electrical Engineering
Stanford, CA 94306

ABSTRACT

This paper describes a set of 58 MATLAB®-based speech processing exercises designed to give students and instructors hands-on experience with digital speech processing basics, fundamentals, representations, algorithms and applications. This result is achieved by providing working MATLAB code using a LITE graphical user interface (GUI) for ease of use and understanding of the code. For each MATLAB exercise we provide a brief description of the code and GUI structure, an explanation of key technical aspects of the solution, a description of the organization of the Lite Graphical User Interface, a scripted run of the software with an extensive set of speech files for processing, some results figures generated by the provided code, and a set of technical issues for further experimentation with the code. Our goal with these exercises is to provide a set of speech processing tools for the instructor and an extended set of speech processing exercises (denoted as "Issues for Further Experimentation") that students can use to develop their understanding of the technical content of each speech processing exercise. This paper describes the resulting set of exercises and shows how they can be used to enhance the teaching and learning of digital speech processing.

Index Terms— MATLAB, speech processing, GUI

1. INTRODUCTION

Digital signal processing is inherently a mathematics-based subject and can be very abstract. Therefore, hands-on experience is exceedingly important in DSP education. Fortunately MATLAB and other tools are available to enable straightforward implementations of DSP concepts. It is especially important in courses focusing on specific application areas, such as digital processing of speech signals, to have access to such powerful learning tools.

Our goal in this project was to provide support for teaching speech processing using our text [1]. Over many years of teaching and writing about this subject, the authors have developed a variety of MATLAB programs for use with both teaching and research applications. We have collected these programs and created a standardized graphical user interface that make them easy to use, in order to illustrate very specific points about the design and implementation of speech processing systems. The complete package of 58 programs and ancillary supporting materials is still under development, but it has reached a stage of maturity where we are comfortable in describing the contents of the package and illustrating what has been and what can be accomplished using these exercises for teaching and learning. The complete package of programs is available through the MATLAB Central website and can be found under the search topic of *speech processing exercises*.

The set of speech processing exercises are aligned, as closely as possible, with the material in our speech processing textbook.[1] The material (i.e., the set of speech processing exercises and the MATLAB code, along with the User's Guide) is grouped into five chapters with the headings:

1. **Basics** – basic MATLAB functionality required for digital speech processing;
2. **Fundamentals** – programs that implement some basic functionality of speech processing;
3. **Speech Representations** – programs that implement the four types of speech representation;
4. **Algorithms** – programs that implement standard speech processing algorithms;
5. **Applications** – programs that implement speech processing applications.

This paper is organized along the same lines. Since there are a total of 58 programs, we can only selectively illustrate the contents and utility of the package by focusing on one example in each of the five exercise areas.

2. OVERVIEW OF THE SET OF PROGRAMS

A question that might naturally arise is, "why are these programs needed when there are many powerful signal analysis programs available?" It is manifestly true that many powerful open source and proprietary programs are available (e.g., Audacity, SPdemo, CoolEdit, wavesurfer, colea, etc.) for displaying waveforms, capturing speech, computing and displaying spectrograms, etc. However, these programs require that the user know exactly what he or she wants to do with

[*]The authors thank MathWorks for generous support of this program.

Table 1. Complete list of MATLAB programs

Basics	record/plot, strips plot, sampling rate conversion, filter signal, echo/reverberate signal
Fundamentals	2-tube vocal tract, 3-tube vocal tract, p-tube vocal tract, glottal pulse, composite vocal tract, ideal vocal tract
Representations	windows, time-domain features, autocorrelation estimates AMDF, phase/magnitude DTFT, narrowband/wideband spectra, overlapped windows, spectrogram, WSOLA, cepstrum computation, single pole cepstrum, FIR sequence cepstrum, cepstrum aliasing, cepstrum liftering, cepstrum waterfall, LPC solutions, LPC error spectra, LPC varying p, LPC varying L, LSP roots, plot roots on spectrogram, spectral smoothing
Algorithms	endpoint detector, VUS training/analysis, autocorrelation pitch detector log harmonic product spectrum, cepstral pitch detector, SIFT pitch detector, formant estimation
Applications	speech coding (uniform, μ-law), AGC, ADPCM coder, VQ training, VQ cells, vowel synthesis, LPC error synthesis, LPC vocoder, play pitch contour, subband coder, phase vocoder, isolated digit recognition

them; e.g., filter the signal, display regions of signal, scale the signal, compute a frequency analysis, etc. Such tools are exceedingly powerful in the hands of an expert signal processing practitioner. However, students and novice users can be at a loss as to which button to push or which menu to select to accomplish a specific task. Our goal was to create a set of speech processing exercises where each exercise focuses on one or a few specific speech or signal processing concepts so that the user is guided in a step-by-step manner to an increased understanding of a specific speech or signal processing concept. (It should be noted that these programs are intended for educational purposes only. They are not meant to be tools for research in speech processing.)

A list of the current set of MATLAB speech processing exercises is given in Table 1, which is organized by the five speech/signal processing areas. We have also developed a user's guide that gives details about each exercise as well as a list of questions for further investigation by the user. These issues for further investigation are provided in order to guide the user toward better understanding of the important aspects of the concepts illustrated in each exercise.

2.1. Basics of Speech Processing

The MATLAB exercises in the "Basics" section guide the user in five speech signal processing directions, namely: how to record and plot the resulting recorded signal; how to display a recorded speech signal using an oscilloscope-like display (called a strips plot); how to change the sampling rate of a speech or audio signal from any of six standardized formats to any other of the six standardized formats; how to lowpass/bandpass/highpass filter a speech signal in order to remove unwanted signal components such as DC offset and hum; and how to echo or reverberate a speech or audio signal using either an FIR or an IIR filter. An illustrative exercise in this area is the zoom_strips_plot exercise.

Strips Plot: Plotting and examining speech waveforms is one of the most useful ways of understanding the properties of speech. This MATLAB exercise displays a speech waveform as both a single line display and as a multi-line plot of speech samples (often called a strips plot). An example of the output from this exercise is shown in the left side of Figure 1. A typical issue for experimentation for this exercise might be: How does the text of the speech file align with the waveform plot on a word-by-word basis, or on a phoneme-by-phoneme basis? Using the *zoom* buttons the user can isolate a section of speech and plot and play the associated speech. Each of these issues for experimentation requires the user to go beyond simply hitting buttons, and forces him or her to think about the phonetic/lexical content of the speech utterance and how it is embodied within the speech waveform.

2.2. Fundamentals of Speech Processing

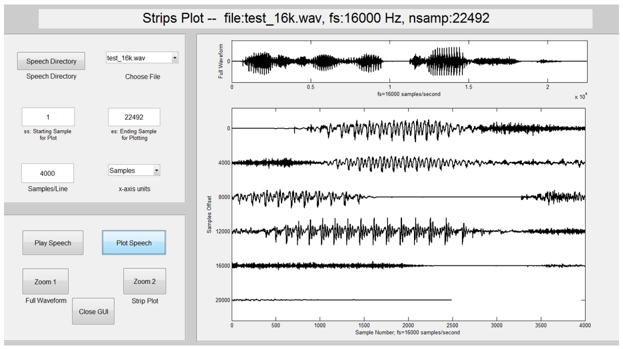
In this section we cover topics such as the basics of speech production and perception from both an acoustic and a linguistics point of view. The MATLAB exercises included in this section are primarily based on modeling the human vocal tract via 2, 3, or p tubes, of varying lengths and cross-sectional areas, along with a discussion of the characteristics of the vocal tract excitation, the vocal tract transfer function, and the radiation characteristic from the lips, as covered in Chapters 2-5 of our textbook.[1] One program that is illustrative of the MATLAB exercises in the area of Fundamentals is the Three-Tube Vocal Tract model.

Three-Tube Vocal Tract Model: The human vocal tract can be modeled as a concatenation of tubes of variable length and area, excited at the glottis by either noise or periodic puffs of air, and terminated at the lips. A three-tube vocal tract model, and the resulting log magnitude frequency response is shown in the right side of Figure 1.

An issue for further experimentation with this MATLAB exercise would be to determine a set of areas and lengths of 3-tube approximations to the major vowel sounds.

2.3. Representations of the Speech Signal

In this section, the speech processing exercises cover the four domains of speech representations, namely time domain, fre-


Strips Plot -- file:test_16k.wav, fs:16000 Hz, nsamp:22492

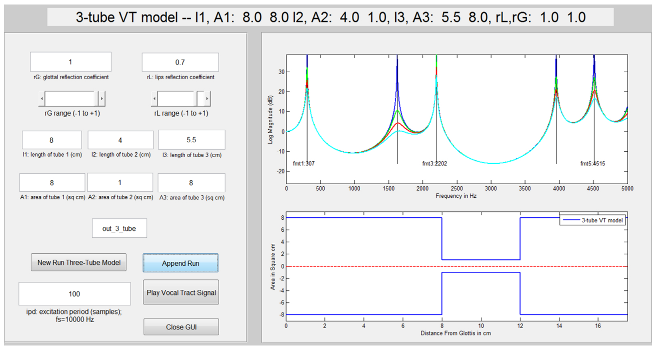
Speech Directory: test_16k.wav (Choose File)

ss: Starting Sample for Plot: 1

es: Ending Sample for Plotting: 22492

Samples/Line: 4000

x-axis units: Samples


3-tube VT model -- l1, A1: 8.0 8.0 l2, A2: 4.0 1.0, l3, A3: 5.5 8.0, rL,rG: 1.0 1.0

rG: glottal reflection coefficient: 1

rL: lips reflection coefficient: 0.7

l1: length of tube 1 (cm): 8

l2: length of tube 2 (cm): 4

l3: length of tube 3 (cm): 5.5

A1: area of tube 1 (sq cm): 8

A2: area of tube 2 (sq cm): 1

A3: area of tube 3 (sq cm): 8

out_3_tube

ipd: excitation period (samples); fs=10000 Hz: 100

Fig. 1. Left: Strips Plot GUI output. Right: Output plot of 3-tube vocal tract model.

quency domain, cepstral domain, and linear predictive domain, as covered in Chapters 6-9 of the textbook.[1] This section has 21 exercises spread across the 4 domains of speech representations. We use the topic of a 'cepstral waterfall' plot to illustrate how the cepstrum evolves over a region of speech in a sequence of cepstral and spectral domain plots.

Cepstral Waterfall: This MATLAB exercise computes, on a frame-by-frame basis, the real cepstrum of a speech signal (or more specifically a section of a speech signal), and displays the resulting frame-by-frame sequence of real cepstrums in a 'waterfall' type of display. To achieve the desired waterfall effect each real cepstrum is normalized to a peak of 1.0 and offset by a constant from the previous frame thereby giving the perception of a sequence of flowing real cepstrums. Such waterfall plots are displayed for both the real cepstrum (on one graphics panel) and the STFT and cepstrally smoothed log magnitude spectrums (on another graphics panel).

An example of the graphical output obtained from this exercise is shown in the left hand side of Figure 2.

One issue for further experimentation would be to manually track the pitch period variation over the frames included within the waterfall plot.

2.4. Algorithms

In this section we include MATLAB exercises that are Algorithms which combine knowledge of fundamentals of speech signals with speech representations and enable the solution of specific problems that are part of one or more speech applications, e.g., an algorithm for detecting the regions of speech and non-speech signals, an algorithm for finding the endpoints of a speech signal embedded in a noisy (non-speech) background, etc., as covered in Chapter 10 of the textbook. We illustrate this type of MATLAB exercise with a simple 'endpoint detector' algorithm.

Endpoint Detector: This MATLAB exercise analyzes an audio file which contains a period of background signal, followed by an interval of spoken speech, and ending with an interval of background signal. The analysis is conducted on a frame-by-frame basis, and, based on a set of short-time log energy and short-time zero crossings rate parameters, the program determines the frame that is the best estimate of the beginning of the speech signal and the frame that is the best estimate of the end of the speech signal.

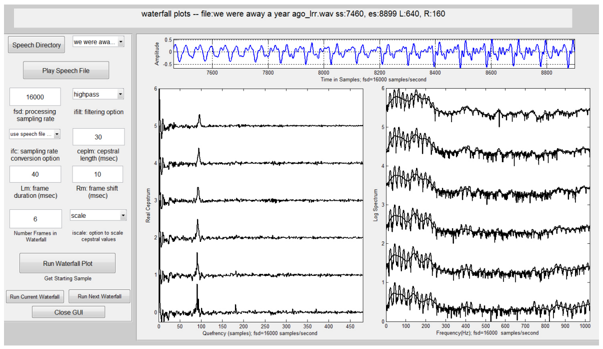
An example of the graphical output obtained from analysis of an utterance that consists of 4 isolated words spoken in sequence, for the sentence /This/-/Shirt/-/Is/-/Red/ is shown in the center of Figure 2. It can be seen that each of the four words was detected properly for this utterance.

2.5. Applications

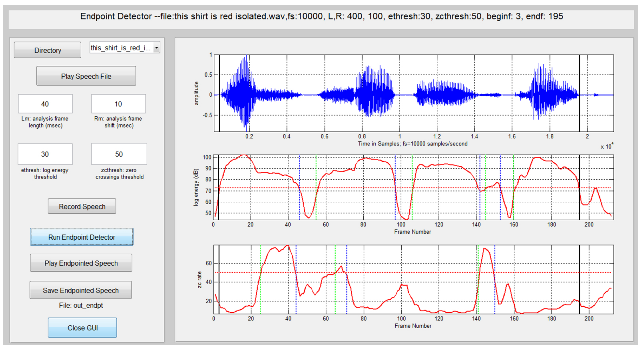
In this section we present MATLAB exercises that are combinations of algorithms based on fundamental representations that lead to solutions to speech processing applications such as waveform coders, LPC vocoder, subband coders, and isolated digit recognition, as covered in Chapters 11-14 of the textbook.[1] We use an LPC Vocoder to illustrate the exercises in this broad category.

LPC Vocoder: This MATLAB exercise builds a classical LPC vocoder, i.e., performs LPC analysis and synthesis on a speech file, resulting in a synthetic speech approximation to the original speech. The LPC analysis uses a standard autocorrelation analysis to determine the sets of LPC coefficients, on a frame-by-frame basis, along with the frame-based gain, G. An independent analysis method (a cepstral pitch period detector) classifies each frame of speech as being either voiced speech (with period determined by the location of the cepstral peak in a designated range of pitch periods) or unvoiced speech (simulated by a random noise frame) designated as a frame pitch period of 0 samples. The independent analysis provides a two-state excitation function for the LPC synthesis part of the processing, consisting of a series of pitch pulses (during voiced frames) and/or noise sequences (during unvoiced frames).

An example of the graphical output obtained from this exercise is shown in Figure 3. The graphics panels show the


waterfall plots -- file:we were away a year ago_lrr.wav ss:7460, es:8899 L:640, R:160

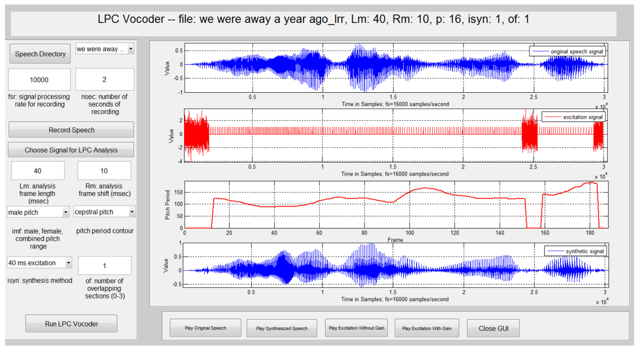
we were awa...
16000 fsd: processing sampling rate
highpass ifilt: filtering option
use speech file ... ifc: sampling rate conversion option
30 ceplm: cepstral length (msec)
40 Lm: frame duration (msec)
10 Rm: frame shift (msec)
6 Number Frames in Waterfall
scale iscale: option to scale cepstral values


Endpoint Detector --file:this shirt is red isolated.wav,fs:10000, L,R: 400, 100, ethresh:30, zcthresh:50, beginf: 3, endf: 195

this_shirt_is_red_i...
40 Lm: analysis frame length (msec)
10 Rm: analysis frame shift (msec)
30 ethresh: log energy threshold
50 zcthresh: zero crossings threshold
File: out_endpt

Fig. 2. Left: Waterfall plots of speech signal. Center: Graphical example of the performance of the speech endpoint detector for the utterance consisting of the spoken words: /This/-/Shirt/-/Is/-/Red/. Right: Graphical output from LPC Vocoder exercise.

speech waveform in the top graphics panel, the excitation signal in the second graphics panel, pitch period contour in the third graphics panel, and the the synthesized signal in the bottom graphics panel.


LPC Vocoder -- file: we were away a year ago_lrr, Lm: 40, Rm: 10, p: 16, isyn: 1, of: 1

we were away ...
10000 fsr: signal processing rate for recording
2 nsec: number of seconds of recording
40 Lm: analysis frame length (msec)
10 Rm: analysis frame shift (msec)
male pitch imf: male, female, combined pitch range
cepstral pitch pitch period contour
40 ms excitation isyn: synthesis method
1 of: number of overlapping sections (0-3)

Fig. 3. Graphical output from LPC Vocoder exercise.

2.6. Modifying the Code

Beneath all the GUI interfaces is MATLAB code that can be modified easily by both instructors and students. Indeed, valuable homework project experiences can be crafted by asking students to modify the programs to include additional or modified functionality. This is particularly true of the programs that illustrate algorithms and applications. For example, in the LPC vocoder, the synthesis procedure can be changed to use overlap-add methods, the pitch detector can be changed from the cepstral method to another one, and the LPC analysis parameters can be quantized or otherwise modified to observe the impact of such changes.

3. SUMMARY AND CONCLUSIONS

We (the authors) have taught digital speech processing for more than four decades and have found that MATLAB exercises increase the understanding of key speech processing concepts. With that in mind, we have created a set of 58 MATLAB speech processing exercises that are tied closely to concepts taught in our recent textbook.[1] Our early experience with these MATLAB speech processing exercises is that students learn more and understand concepts better when they have practical implementations that are so closely aligned with the material of the course. Although the exercises are keyed to the organization of our textbook, this should not preclude their use with any other textbook such as [2, 3] and others.

4. ACKNOWLEDGEMENT

The support of Mathworks for the development and distribution of this package of MATLAB programs is greatly appreciated. In particular, Ye Cheng of Mathworks has given invaluable assistance in working out the details of making the programs available through the MATLAB Central website. We acknowledge and thank our two students, Reemy Maria d'Souza and Dan Litvin who created and developed GUI Lite 2.5, the platform on which all the GUIs were designed. We also acknowledge the contributions of Siva Yedithi and Kirty Vedula for their work on making the exercises as robust as possible. Finally we acknowledge and thank Prof. Dan Ellis of Columbia for creating the phase vocoder software and making it available to us via the Columbia University website.

5. REFERENCES

[1] L. R. Rabiner and R. W. Schafer *Theory and Applications of Digital Speech Processing*, Prentice-Hall, 2011.

[2] T. F. Quatieri, *Principles of Discrete-Time Speech Processing*, Prentice-Hall, 2002.

[3] E. S. Gobi, *Digital Speech Processing using Matlab*, Springer India, 2014.