

MATLAB Functionality for Digital Speech Processing

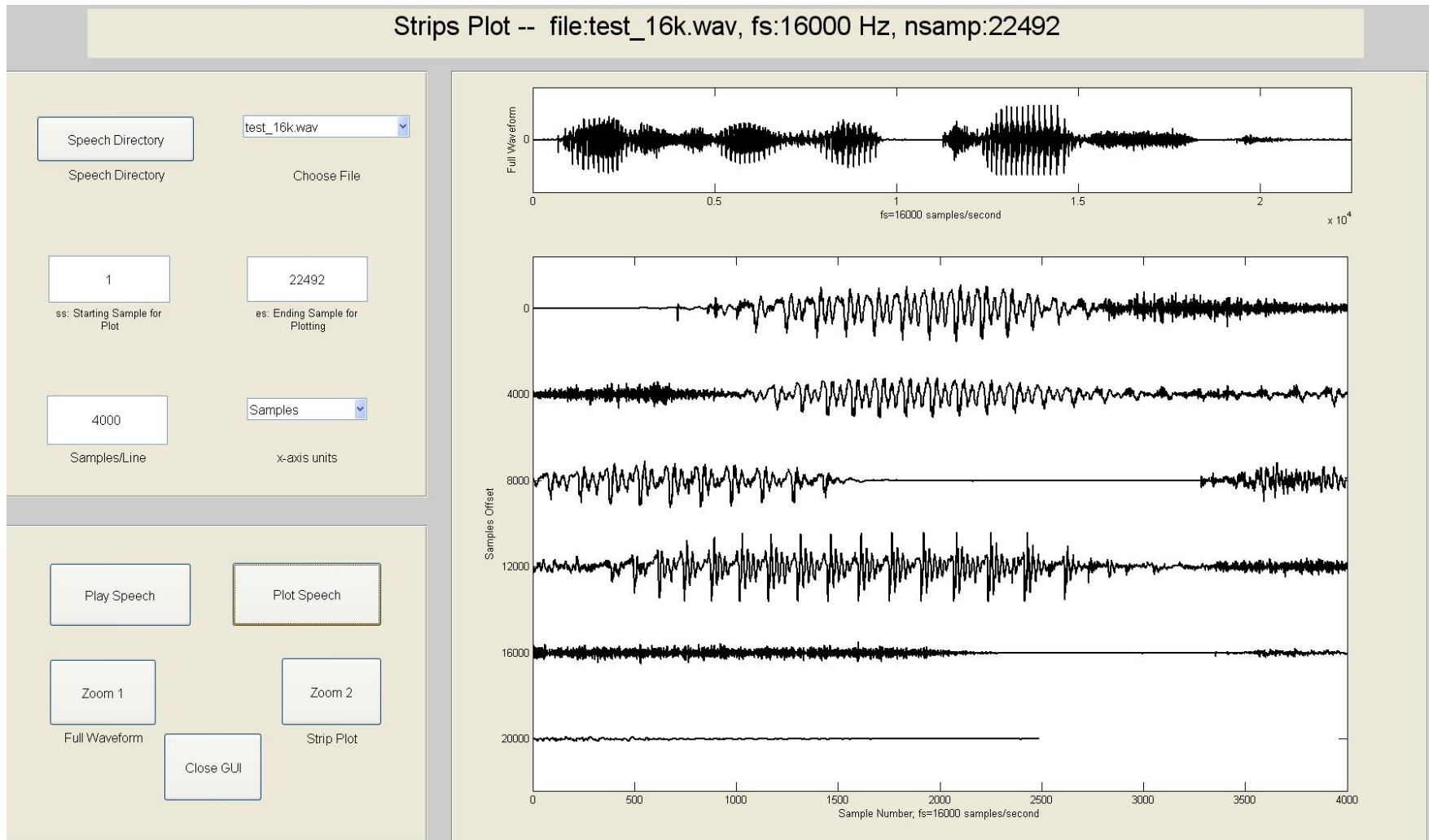
- MATLAB Speech Processing Code
- MATLAB GUI Implementations

Graphical User Interface

GUI Lite 2.6

Waveform Strips Plot

Basics



How do we rapidly and efficiently create a GUI for problems like the one shown above?

Graphical User Interface Components

- GUI Lite created by students at Rutgers University to simplify the process of creating viable GUIs for a wide range of speech and image processing exercises
- GUI Lite Elements
 - basic design tool and editor (GUI Lite 2.5/2.6)
 - panels; used to block group of buttons/graphical panels/etc., into one or more coherent blocks
 - graphics panels; used to display one or more graphical outputs (figures)
 - text block; used to display global information about the specific speech processing exercise
 - buttons; used to get and set (vary) exercise parameters; used to display a list of exercise options; used to initiate actions within the code
 - editable buttons – get and/or set parameter value
 - text buttons – display variable values
 - slider buttons – display variable range
 - popupmenu buttons – display list of variable options (e.g., list of speech files)
 - pushbuttons – initiate actions within the code

Zoom Waveform Strips Plot

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

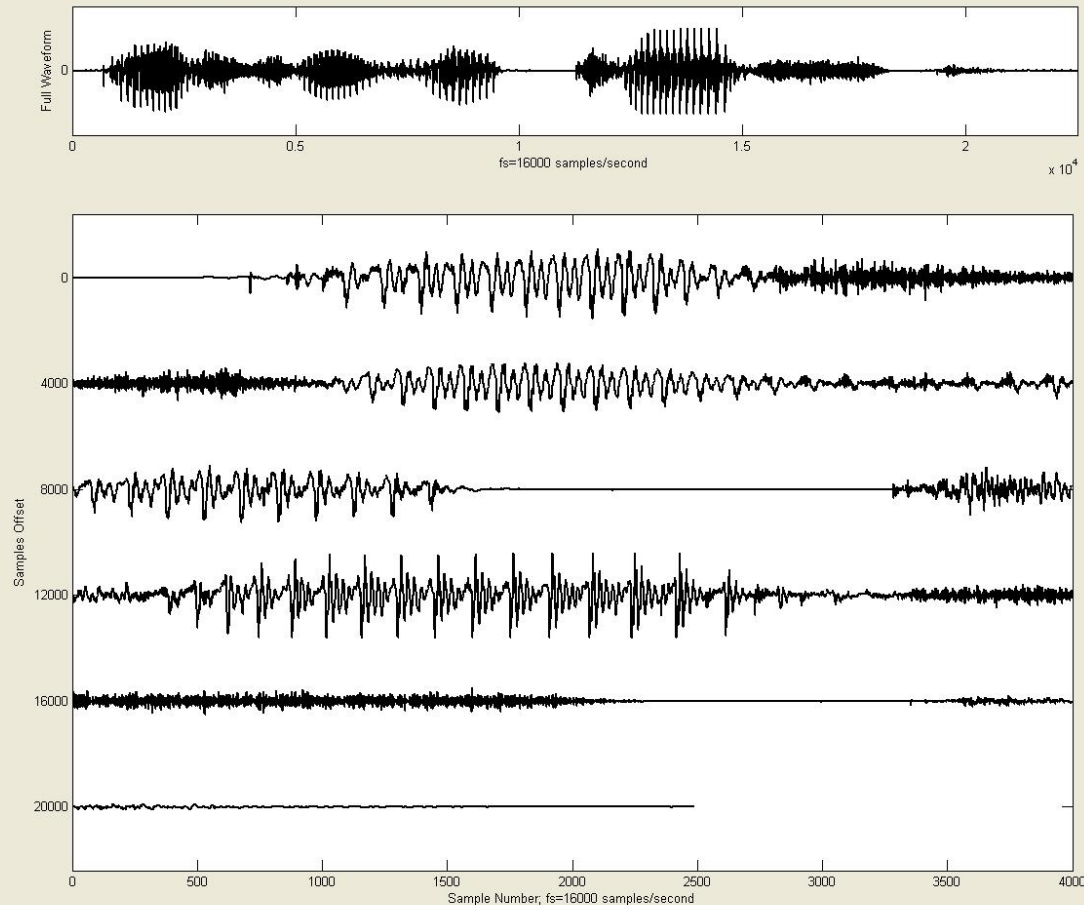
Text Box 1

Graphics
Panel 1

Graphics
Panel 2

Buttons 1-11

Panel 2



Panel 1

Panel 3

GUI LITE 2.6 Setup Process

- Create a directory for loading all necessary file folders for creating GUI 26 exercises; i.e., call this directory:
`'matlab_central_speech'`
- Define full path to the chosen directory; i.e.,
`path_to_speech='C:\data\matlab_central_speech'`
- Download (from MATLAB Central), and place in the chosen directory, the following code and data folders:
 - functions_lrr; speech_files; highpass_filter_signal; VQ;
 - cepstral coefficients; isolated_digit_files; GUI_LITE_26;
- Download (from MATLAB Central) the folder:
`'pathnew_matlab_central.m'`
- Run pathnew script to set up path links to GUI LITE 26 and to the directories above

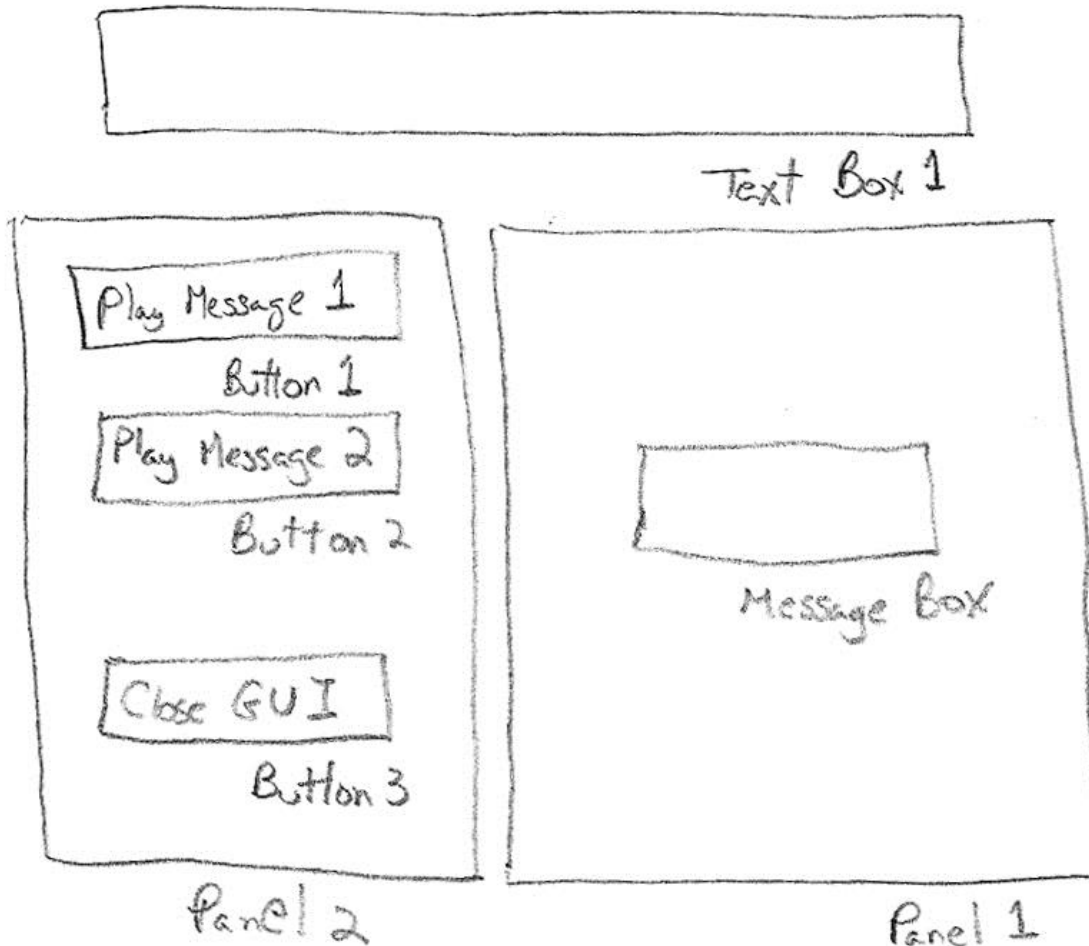
GUI LITE 2.6 Setup Process

- Search for MATLAB Central using web search tool with search input 'MATLAB Central' and go to MATLAB Central
- Click on File Exchange
- In the File Exchange website type 'speech processing exercises' in the search box
- Find and download each of the folders from the previous vu-graph;
 - find the folder of interest
 - click on download
 - from the displayed zip file, choose extract and search for the matlab directory and download all files

GUI LITE 2.6 Design Process

- begin with a rough sketch of the GUI 2.6 output, segmented into button panels, graphics panels, text boxes, and buttons
- create exercise folder; e.g., 'hello_goodbye_world'
- run program 'runGUI.m' to create GUI elements and save as a GUI file (e.g., filename.mat);
- the software also creates an apps program named filename_GUI26.m and a Callbacks program (just the GUI structure) named Callbacks_filename_GUI26.m
- edit the program Callbacks_filename_GUI26.m by adding the appropriate callback code for each of the graphics panels, text boxes and button boxes
- run the resulting exercise and loop on GUI design and Callbacks implementation

Hello/Goodbye World Plan



Design Specs:

- 2 Panels (for linking inputs and outputs)
- 1 Text Box (for describing the Exercise GUI)
- 3 Buttons (all pushbuttons) (for embedding Callback code to play two messages and to close up the GUI)

GUI26 Initial Screen

GUI Lite v2.5

Select Workplace Directory

Current Workplace Directory: C:\data\matlab_gui_current\hello_goodbye_world_gui25

New

Create New GUI

Run 1

Run with runGUI.m File

Run 2

Run w/ .mat & callBack.m Files

Mod

Modify Existing GUI

close

GUI26 Creation for 'hello_goodbye_world'

- run the program 'runGUI.m' and click on the 'New' button
- enter values for the number of panels (2), number of graphics panels (0), number of text boxes (1), and number of buttons (3)
- enter the name for the GUI .mat file ('hello_goodbye_world')
- automatically create the set of GUI objects specified above, using the mouse cursor to define the range and properties of each object (this step is guided by the program 'runGUI.m')
- the resulting specifications for the GUI are automatically saved in the designated .mat file, 'hello_goodbye_world.mat'
- edit the Callbacks routine, 'Callbacks_hello_goodbye_world_GUI26.m', by entering the MATLAB code for the graphics panels, text boxes and button boxes

GUI26 Callback Code – Button 1

% Callback for button 1 – print out message Hello World

```
function button1Callback(h,eventdata);
```

```
% title box
```

```
    stitle1=strcat('Hello World Using GUI2.5');
```

```
    set(titleBox1, 'String', stitle1);
```

```
    set(titleBox1, 'FontSize', 25);
```

```
% uiwait(msgbox('Hello World!', 'Message1', 'modal'));
```

```
waitfor(errordlg('Hello World','Message1'));
```

```
return;
```

```
end
```

GUI26 Callback Code – Button 2

% Callback for button 2 – print out message Hello World

```
function button1Callback(h,eventdata);
```

```
% title box
```

```
    stitle1=strcat('Goodbye World Using GUI2.5');
```

```
    set(titleBox1, 'String', stitle1);
```

```
    set(titleBox1, 'FontSize', 25);
```

```
    % uiwait(msgbox('Goodbye World!', 'Message1', 'modal'));
```

```
    waitfor(errordlg('Hello World','Message1'));
```

```
    return;
```

```
end
```

GUI26 Callback Code – Button 3

% Callback for button3 -- Close GUI

```
function button3Callback(h,eventdata);
```

```
% title box
```

```
    stitle1=strcat('Close GUI');
```

```
    set(titleBox1,'String',stitle1);
```

```
    set(titleBox1,'FontSize',25);
```

```
    uiwait(msgbox('Closing GUI','Message2','modal'));
```

```
    % waitfor(errordlg('Closing GUI','Message2'));
```

```
    % return;
```

```
    display Goodbye
```

```
    close(gcf);
```

```
end
```

[hello_goodbye_world](#)

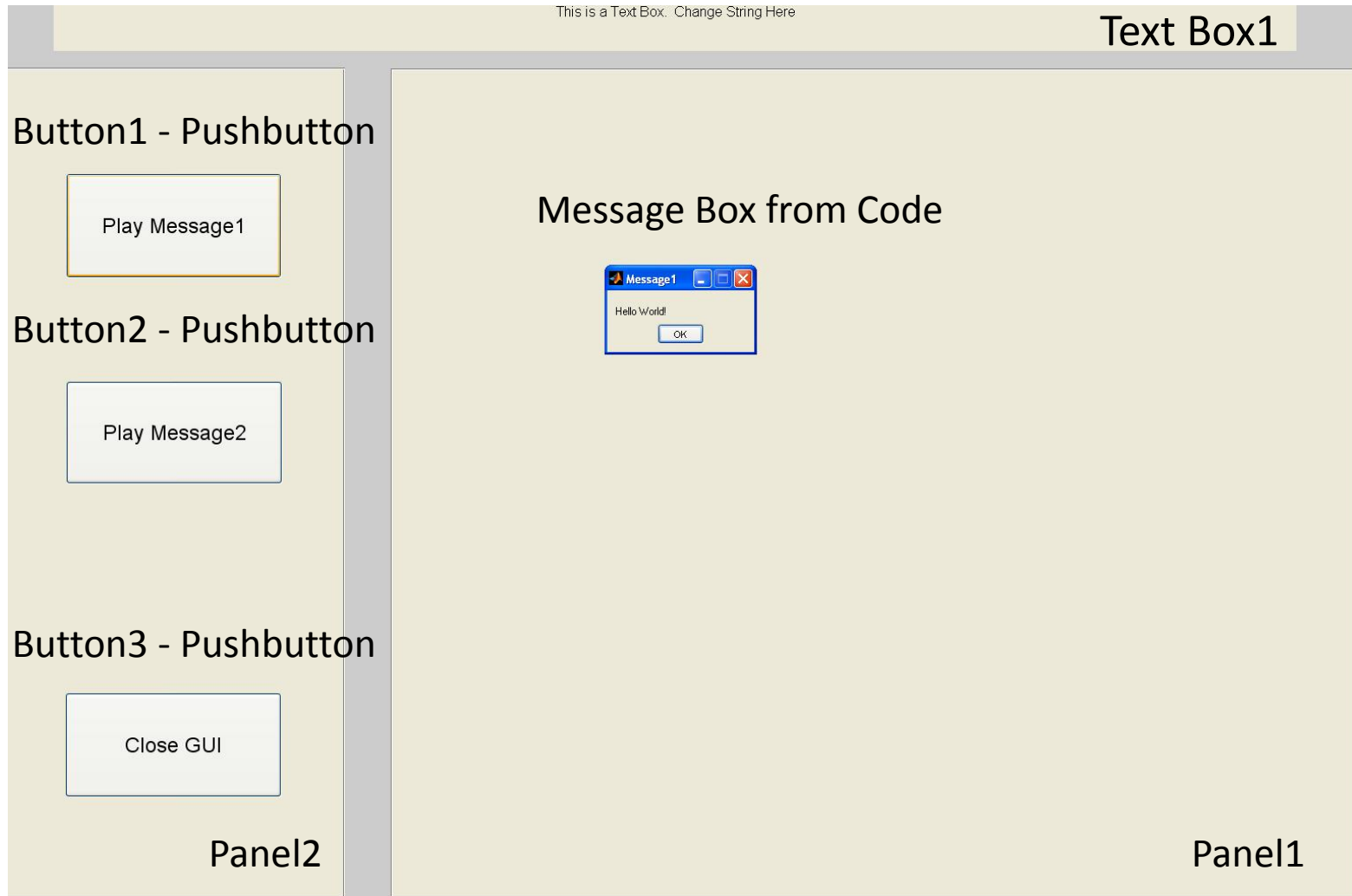
run:

hello_goodbye_world_GUI26.m

directory:

hello_goodbye_world

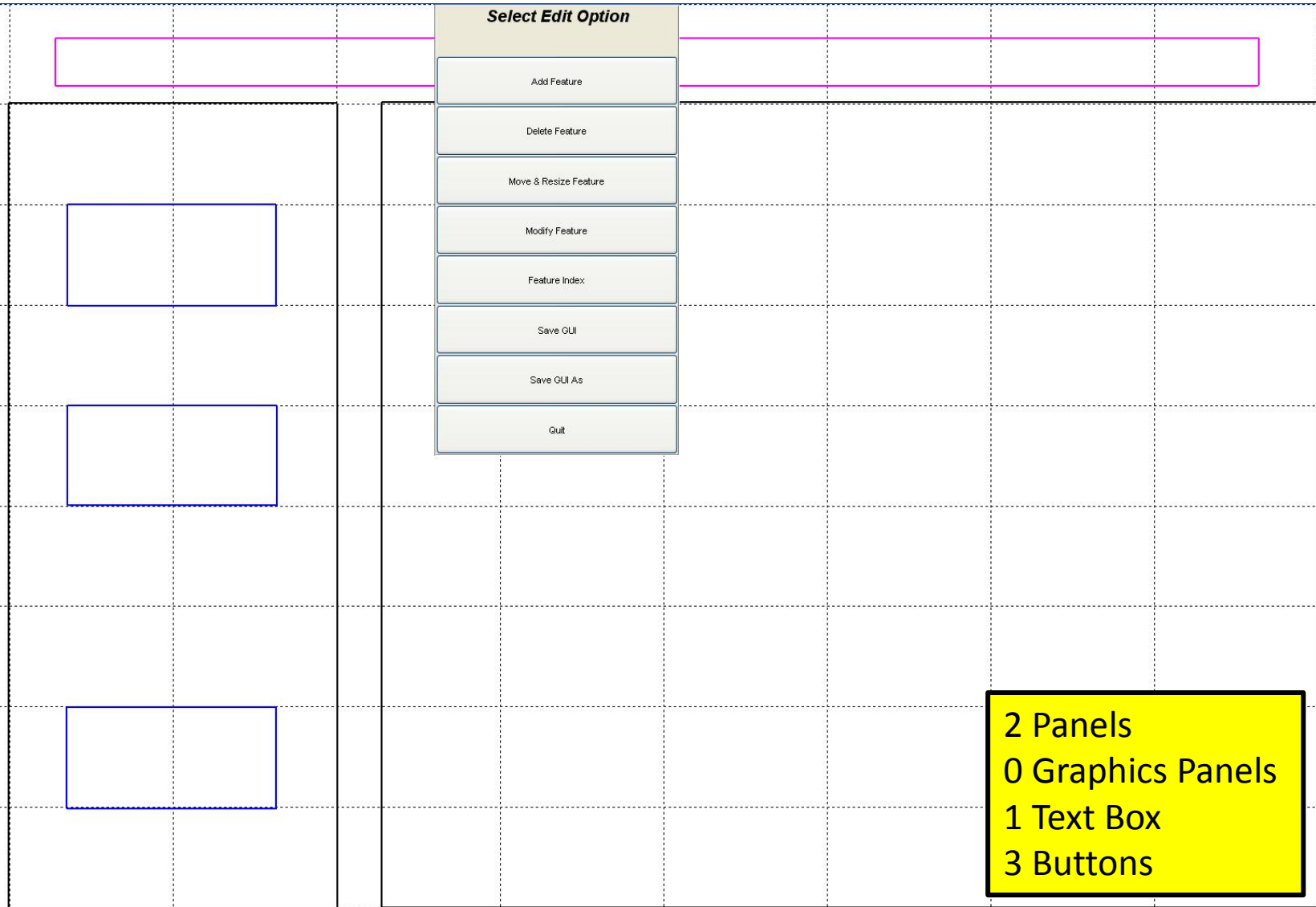
hello_goodbye_world



hello_goodbye_world – GUI Modifications

- Run program 'runGUI.m' to bring up GUI Lite 2.6 editor
- Choose Mod (modify) and select GUI file 'hello_goodbye_world.mat' for editing
- Choose 'Move & Resize Feature' option
- Choose 'Button' option
- Left click inside button to be modified
- Choose new button coordinates by using graphics cursor to identify lower left and upper right corners of modified button
- Click 'Save GUI' button
- Iterate on other buttons
- Click 'Quit' option to terminate GUI Lite 2.6 editor

GUI Lite 26 Edit Screen



GUI LITE 2.6 Edit Screen

Select Edit Option
Add Feature
Delete Feature
Move & Resize Feature
Modify Feature
Feature Index
Save GUI
Save GUI As
Quit

Add Feature

Delete Feature

Move & Resize Feature

Modify Feature

Feature Index

Save GUI

Save GUI As

Quit

[runGUI.m](#)

[hello_goodbye_world_GUI26.m](#)₉

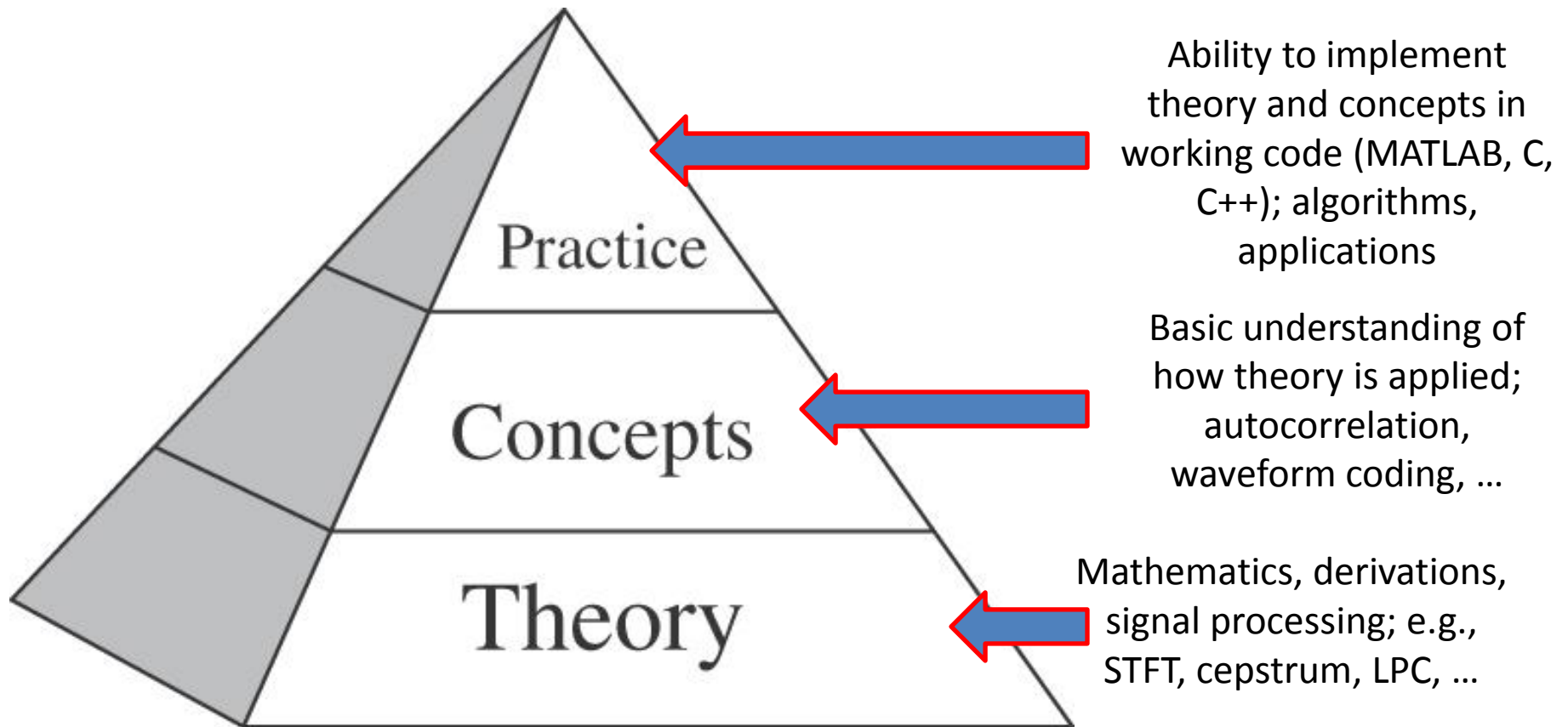
GUI Lite 2.6 Features

- separates GUI design from Callbacks for each GUI element
- provides a versatile editor for modifying GUI elements without impacting the Callback actions
- provides a GUI element indexing feature that enables the user to identify GUI elements with the appropriate Callback elements

Missing GUIDE Features

- radio button
- check box
- listbox
- toggle button
- table
- axes
- button group
- active X control
- are the missing features of value?
- do we need these features?
- can we create the desired set of speech processing exercises without these features?
- can we add these features to the GUI LITE editor?

Pyramid of Learning



Need to understand speech processing at all three levels

The Speech Stack

Speech Applications — coding, synthesis, recognition, understanding, verification, language translation, speed-up/slow-down

Speech Algorithms — speech-silence (background), voiced-unvoiced, pitch detection, formant estimation

Speech Representations — temporal, spectral, homomorphic, Linear Prediction Coding

Fundamentals — acoustics, linguistics, pragmatics, speech production/perception

Basics – read/write speech/audio files; display speech files; play files

MATLAB Exercise Categories

- Basic MATLAB Functions for handling speech and audio files
- Advanced MATLAB Functions for Speech Processing

MATLAB Exercise Categories

- The speech processing exercises are grouped into 5 areas, namely:
 - **Basics** of speech processing using MATLAB (5)
 - **Fundamentals** of speech processing (6)
 - **Representations** of speech in time, frequency, cepstrum and linear prediction domains (22)
 - **Algorithms** for speech processing (7)
 - **Applications** of speech processing (17)

Basic Functionality

- **read a speech file** (i.e., open a .wav speech file and read the speech sample into a MATLAB array)
- **write a speech file** (i.e., write a MATLAB array of speech samples into a .wav speech file)
- **play a MATLAB array** of speech samples as an audio file
- * **play a sequence of MATLAB arrays of speech samples** as a sequence of audio files
- **record a speech file** into a MATLAB array
- **plot a speech file** (MATLAB array) as a waveform using a strips plot format
- * **plot a speech file** (MATLAB array) as one or more 4-line plot(s)
- **convert the sampling rate** associated with a speech file (MATLAB array) to a different (lower/higher) sampling rate
- **lowpass/highpass/bandpass filter** a speech file (MATLAB array) to eliminate DC offset, hum and low/high frequency noise
- **plot a frame of speech** and its associated spectral log magnitude
- **plot a spectrogram** of a speech file (MATLAB array)
- * **plot multiple spectrograms** of one or more speech files (MATLAB arrays)

* indicates exercise not yet done

Read a Speech File into a MATLAB Array

- `[xin, fs, nbits] = wavread(filename);`
- `[xin, fs] = loadwav(filename);`
 - filename is ascii text for a .wav-encoded file which contains a speech signal encoded using a 16-bit integer format
 - xin is the MATLAB array in which the speech samples are stored (in double precision format)
 - fs is the sampling rate of the input speech signal
 - nbits is the number of bits in which each speech sample is encoded (16 in most cases)
 - program wavread scales the speech array, xin, to range $-1 \leq xin \leq 1$, whereas loadwav preserves sample values of the speech file and hence array xin is scaled to range $-32768 \leq xin \leq 32767$
- `[xin1, fs, nbits] = wavread('s5.wav');`
- `[xin2, fs] = loadwav('s5.wav');`

Read a Speech File into a MATLAB Array

- `% test_wavread.m`
- `% test waveread function`
- `%`
- `% read speech samples from file 'test_16k.wav' into array x1 using wavread`
- `% routine`
- `filein='test_16k.wav';`
- `[x1,fs1,nbits]=wavread(filein);`
-
- `% print out values of fs1, nbits, wavmin1, wavmax1`
- `wavmin1=min(x1);`
- `wavmax1=max(x1);`
- `fprintf('file: %s, wavmin/wavmax: %6.2f %6.2f, fs1: %d, nbits: %d \n',...`
- `filein,wavmin1,wavmax1,fs1,nbits);`
-
- `% read speech samples from same file into array x2 using loadwav routine`
- `[x2,fs2]=loadwav(filein);`
-
- `% print out values of fs2, nbits, wavmin2, wavmax2`
- `wavmin2=min(x2);`
- `wavmax2=max(x2);`
- `fprintf('file: %s, wavmin/wavmax: %d %d, fs2: %d \n',...`
- `filein,wavmin2,wavmax2,fs2);`

Terminal Display:

file: test_16k.wav, wavmin/wavmax: -1.00 1.00, fs1: 16000, nbits: 16

file: test_16k.wav, wavmin/wavmax: -32768 32767, fs2: 16000

Write a Speech Array into a Speech File

- `wavwrite(xout, fs, nbits, filename);`
- `savewav(xout, filename, fs);`
 - `xout` is the MATLAB array in which the speech samples are stored
 - `fs` is the sampling rate of the output speech signal
 - `nbits` is the number of bits in which each speech sample is encoded
 - `filename` is the ascii text for the .wav-encoded file in which the MATLAB signal array is to be stored
 - for `wavwrite` the MATLAB array `xout` needs to be scaled to the range $-1 \leq x_{in} \leq 1$ whereas for `savewav` the MATLAB array `xout` needs to be scaled to the range $-32768 \leq x_{out} \leq 32767$
- `wavwrite(xin1, fs, 's5out.1.wav');`
- `savewav(xin2, 's5out.2.wav', fs);`

Play a Speech File

- `sound(x, fs);`
- `soundsc(x, fs);`
 - for `sound` the speech array, `x`, must be scaled to the range $-1 \leq x \leq 1$
 - for `soundsc` any scaling of the speech array can be used
 - `fs` is the sampling rate of the speech signal
- `[xin, fs] = loadwav('s5.wav');` % load speech from s5.wav;
- `xinn = xin/abs(max(xin));` % normalize to range of - 1 to 1;
- `sound(xinn, fs);` % play out normalized speech file;
- `soundsc(xin, fs);` % play out unnormalized speech file;

Play Multiple Speech Files

- `*play_multiple_files.m;`
 - sequence of filenames read in via filelist, keyboard or file search
- Example of usage to play out 3 speech files in sequence:
 - `kbe=filename` entry via `filelist(2)`, `keyboard(1)`, or `file search(0):1;` % keyboard chosen
 - `N=number of files to be played in a group:3;` % play out 3 files
 - `i=1; filename: s1.wav;`
 - `i=2; filename: s2.wav;`
 - `i=3; filename: s3.wav`

Play Multiple Speech Files

- test_play_files.m
 - play the following sequence of files:

s2.wav

s3.wav

s4.wav

s5.wav

s6.wav



[play_multiple_files_GUI26.m](#)

Record Speech into MATLAB Array

- `record_speech.m` (calls MATLAB function `audiorecorder.m`, formally `wavrecord.m`)
- `function y=record_speech(fs, nsec);`
 - `fs`: sampling frequency
 - `nsec`: number of seconds of recording
 - `y`: speech samples array normalized to peak of 32767

Display Speech Waveform

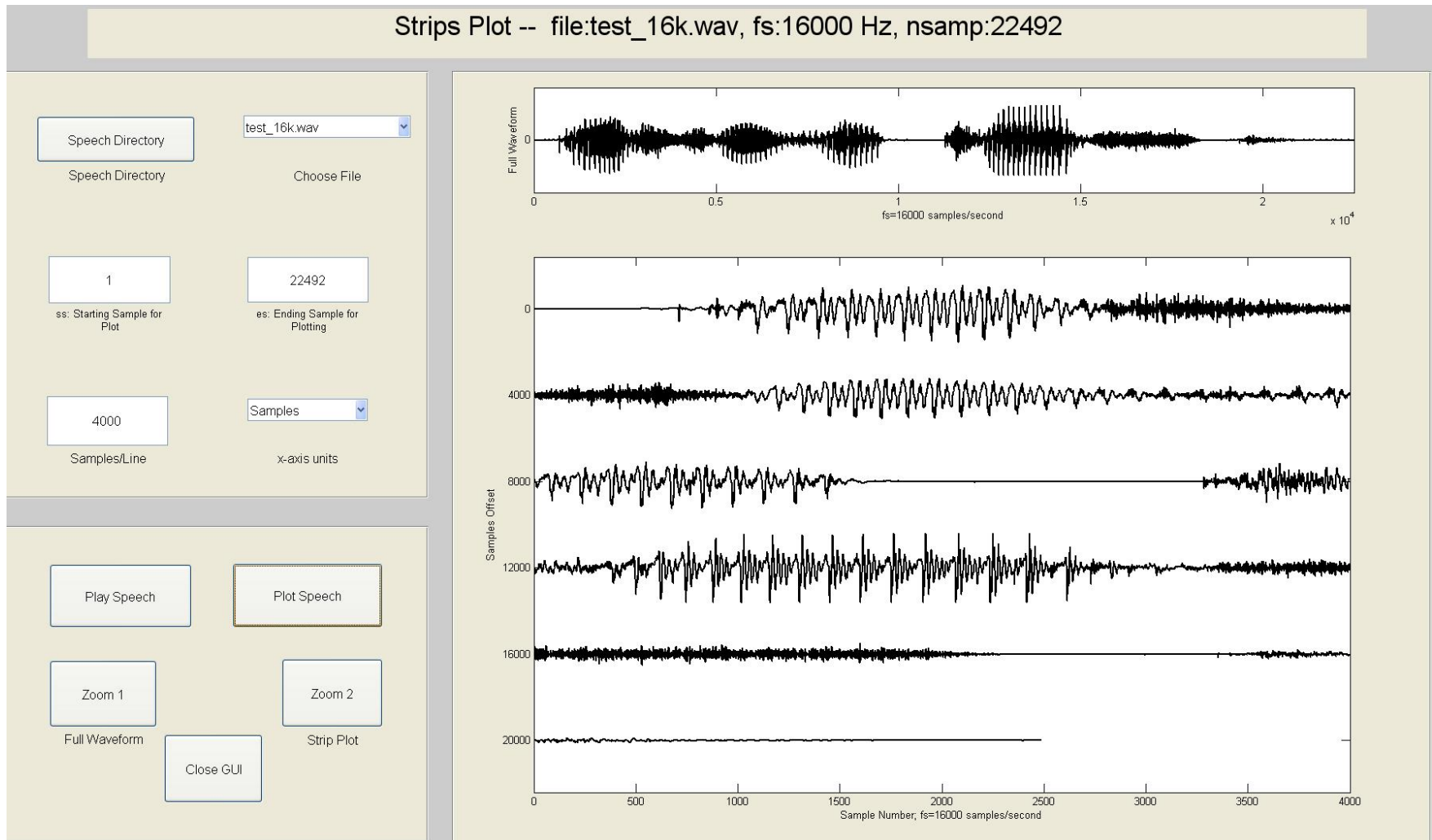
Strips Plot

4-Line Plots

Zoom Waveform Strips Plot

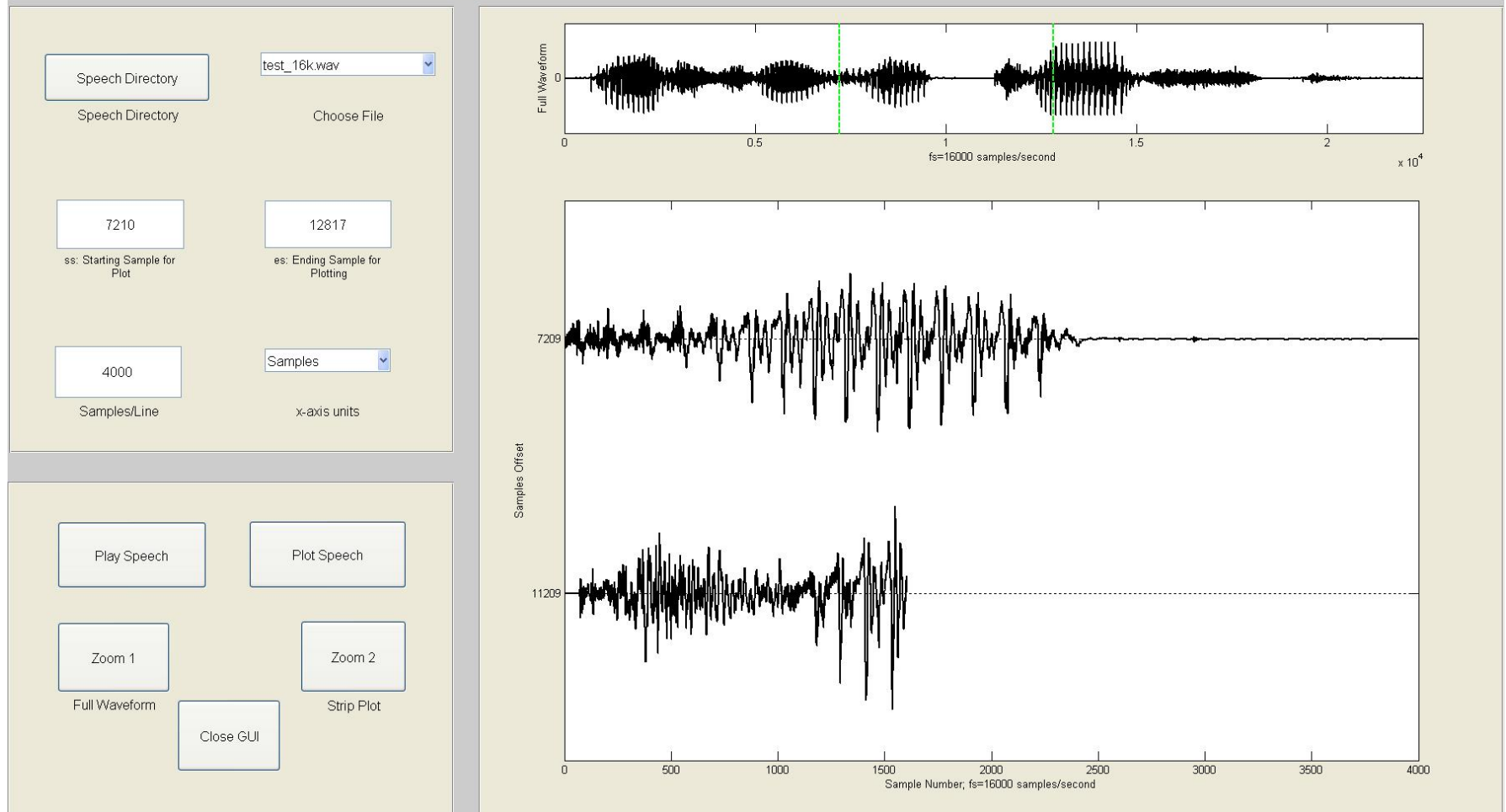
- Plotting and examining speech/audio waveforms is one of the most useful ways of understanding the properties of speech and audio signals.
- This MATLAB Exercise displays a speech/audio waveform as a single running plot of samples (called a Strips Plot).
- Exercise plots from designated starting sample to designated ending sample, with a user-specified number of samples/line.
- Zoom feature to select region of signal for display.
- Plots use either samples or seconds, as specified by the user.

Waveform Strips Plot



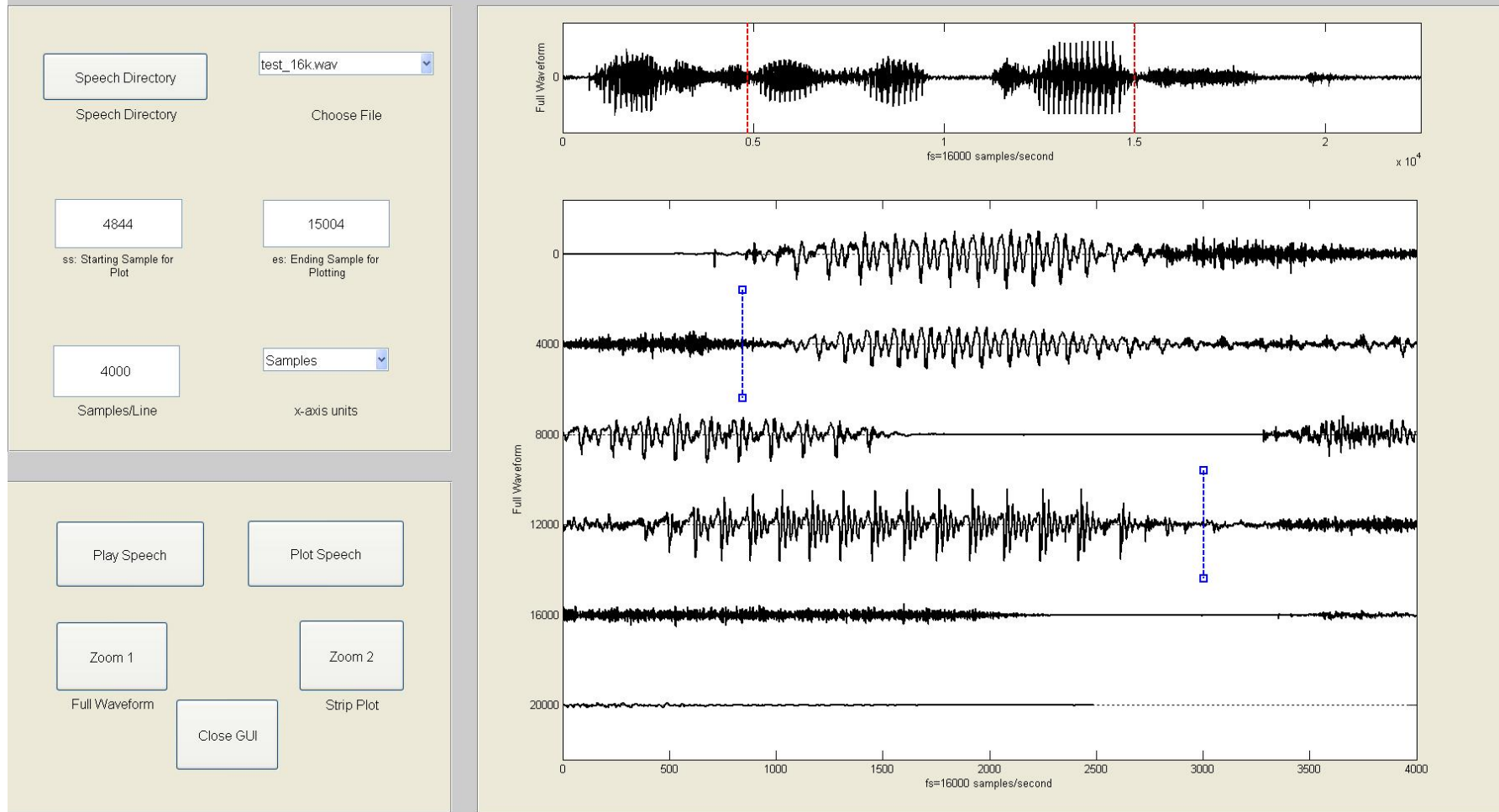
Waveform Strips Plot – Zoom 1

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



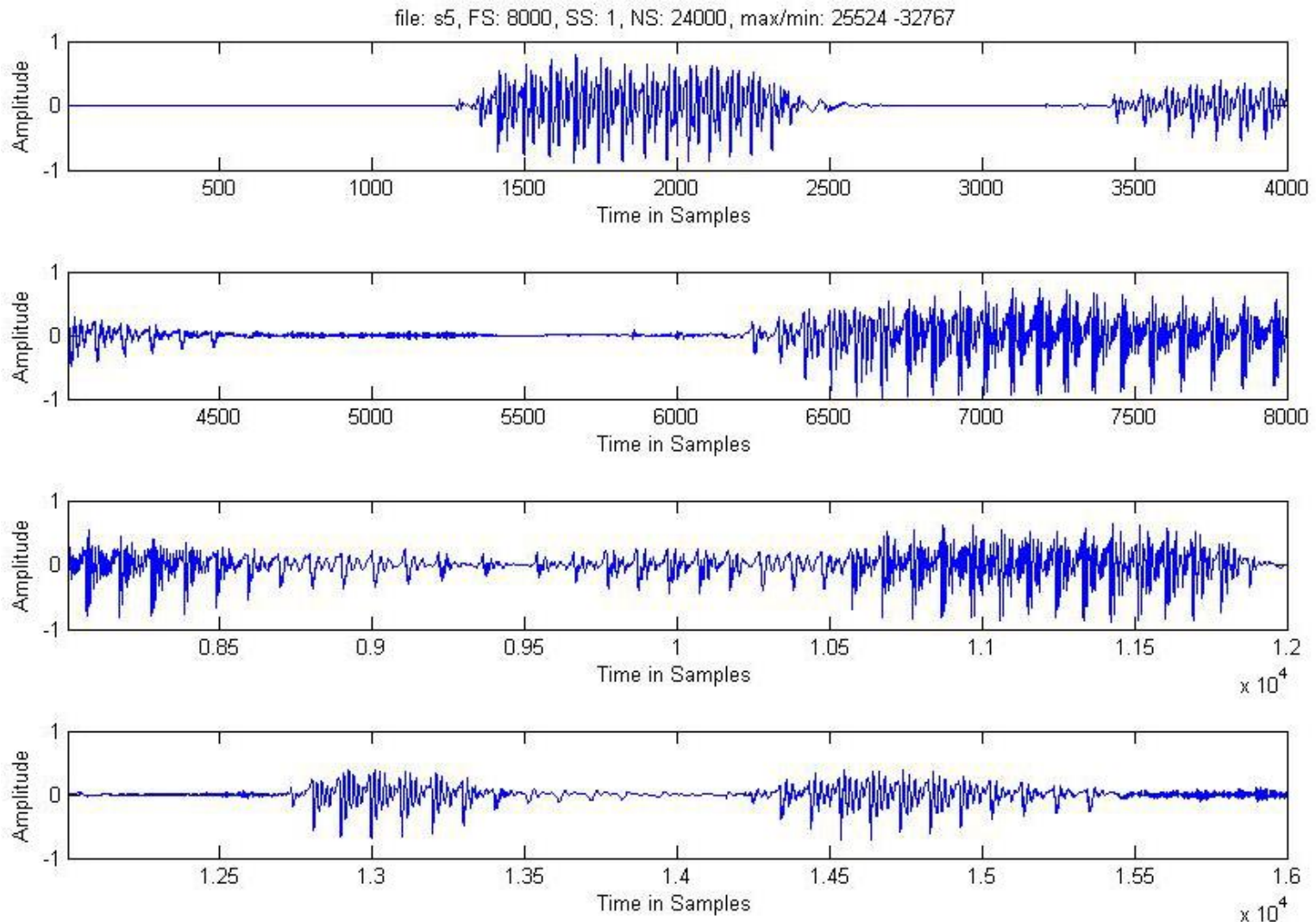
Waveform Strips Plot – Zoom 2

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



[zoom_strips_plot_GUI25.m](#)

*Plot Speech Using 4-Line Plot



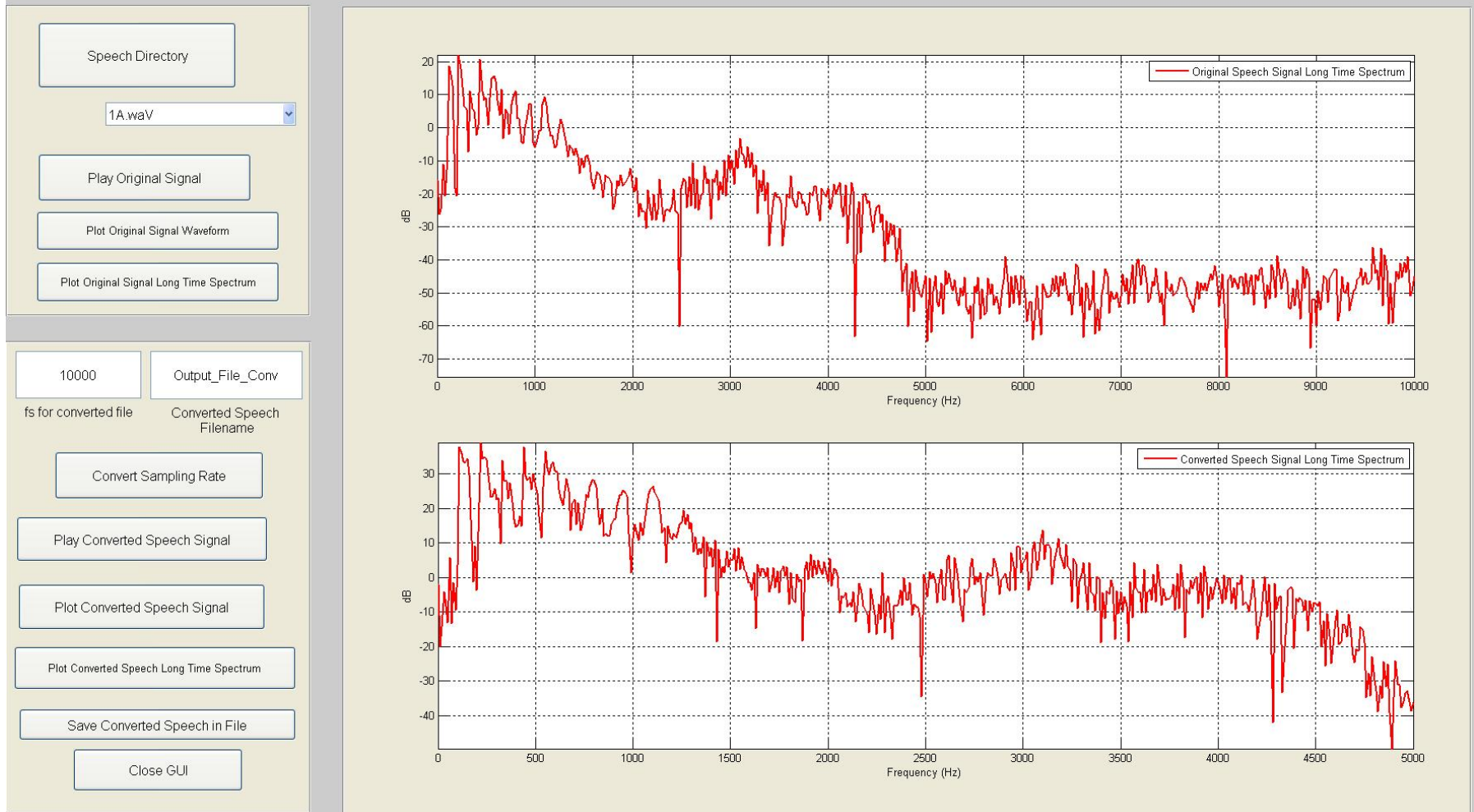
Sampling Rate Conversion

- `y = srconv(x, fsin, fsout);`
 - `x`: input speech array;
 - `fsin`: input speech sampling rate;
 - `fsout`: desired speech sampling rate;
- Example:
 - `[xin, fsin] = loadwav('s5.wav');` % `fsin=8000`;
 - `fsout = 10000;` % desired sampling rate;
 - `y = srconv(xin, fsin, fsout);`

Sampling Rate Conversion

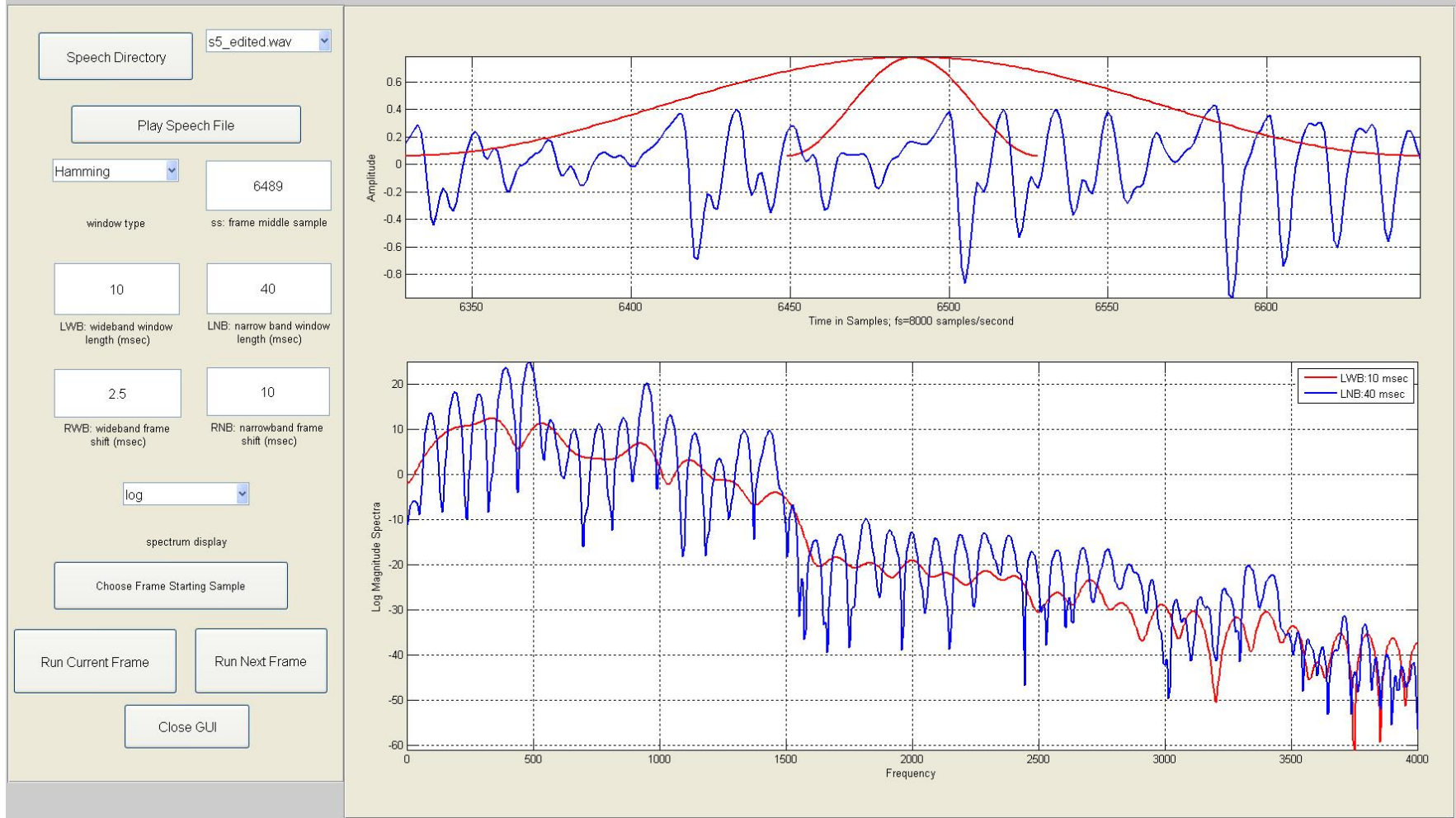
Basics

Sampling Rate Conversion -- file:1A.waV, fs:20000 Hz, nsamp:15872



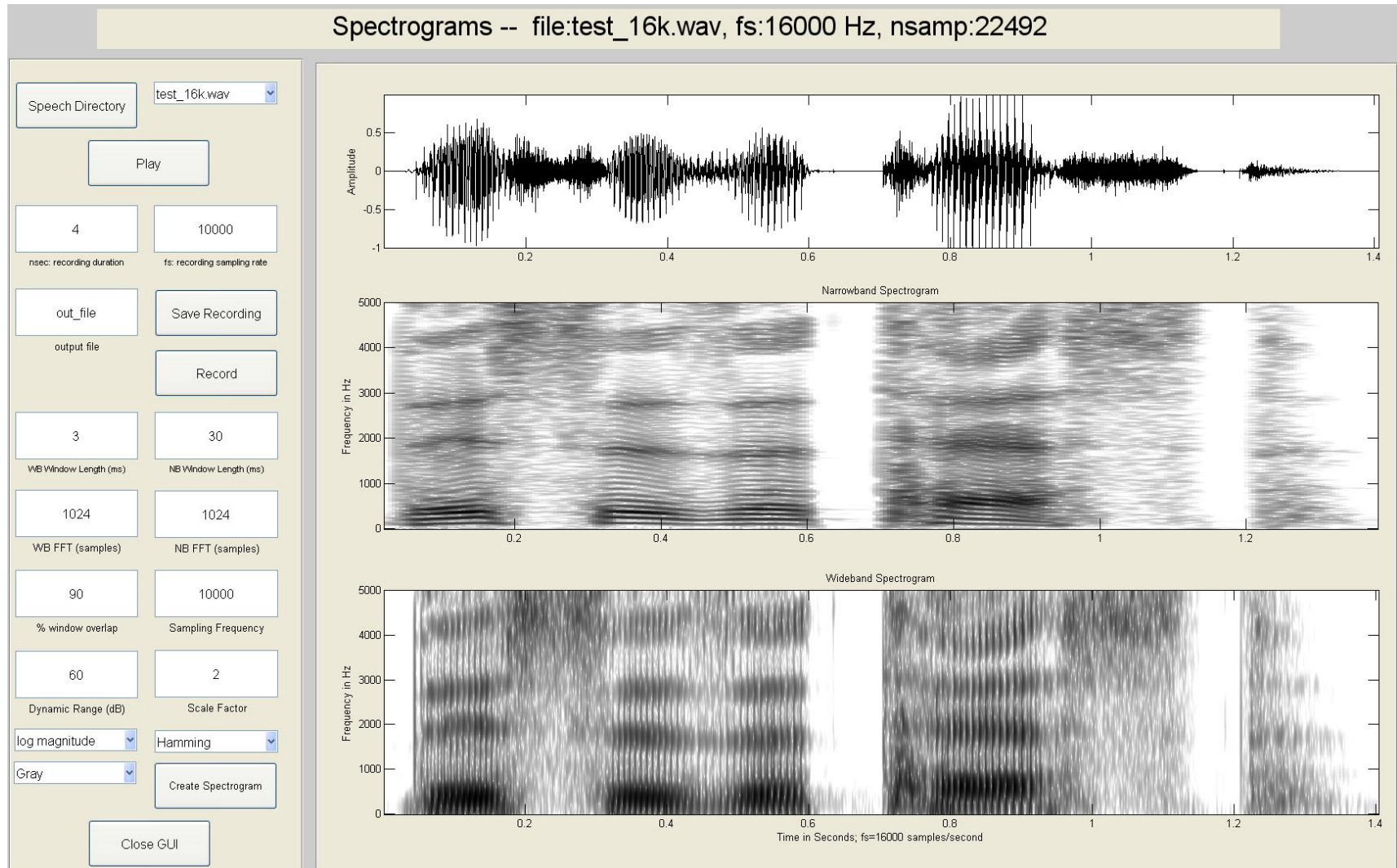
Frame-Based Spectra

HW magnitude/log magnitude spectra -- file: s5_edited.wav, middle sample: 6489, window lengths (msec): 10 40

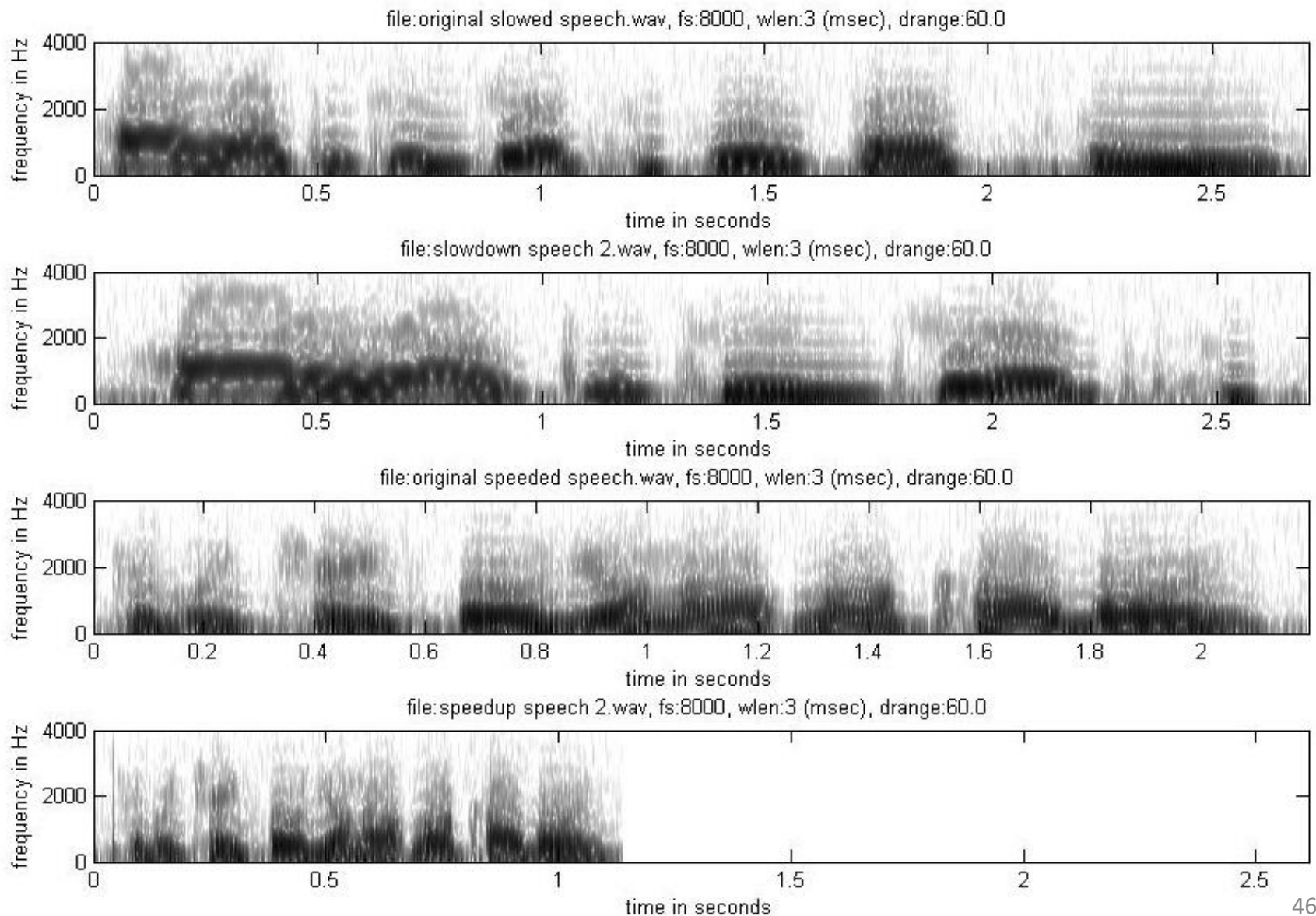


[NB WB spectra GUI25.m](#)

Wideband/Narrowband Spectrogram



*Plot Multiple Spectrograms



Fundamentals

- 2-tube vocal tract model
- 3-tube vocal tract model
- p-tube vocal tract model
- glottal pulse model and spectrum
- composite vocal tract model and spectrum
- ideal vocal tract model and spectrum

Representations

- **time domain exercises**
 - windows; features; autocorrelation estimates; amdf
- **frequency domain exercises**
 - phase/magnitude; overlap-add windows; WSOLA
- **cepstral domain exercises**
 - analytical cepstrum; single pole cepstrum; FIR sequence cepstrums; cepstrum aliasing; cepstrum liftering; cepstral waterfall
- **linear prediction exercises**
 - LPC frames; LPC error; LPC varying p ; LPC varying L ; LSP roots; plot roots

Algorithms

- endpoint detector
- Voiced-Unvoiced-Background estimation method
- autocorrelation pitch detector
- log harmonic spectral waterfall plots
- cepstral pitch detector
- SIFT pitch detector
- formant estimation method

Applications – Part 1

- Speech waveform coding;
 - statistical properties of speech; quantization characteristics of a B-bit uniform or mu-law compressed and quantized speech file; uniform quantization; mu-law compression; mu-law quantization; Signal-to-Noise Ratio (SNR) of uniform and mu-law quantizers
- Automatic Gain Control (AGC)
- Adaptive Differential Pulse Code Modulation (ADPCM) waveform speech coder
- Vector Quantizer (VQ); VQ Cells
- Synthetic vowel synthesizer

Applications – Part 2

- LPC error synthesis
- LPC vocoder
- Play pitch period contour
- Two-Band subband coder
- Phase Vocoder
- Isolated, speaker-trained, digit recognizer

Speech Processing Exercises

- Search for MATLAB Central using local browser
- Click on file exchange; search for speech processing exercises
- Create a directory in which the various speech processing apps and data folders will be placed (e.g., speech_apps); use the full path in the pathnew routine (see below); e.g., C:\data\speech_apps
- If loading one or more speech processing exercises, download the following:
 - Read_Me which explains the downloading process
 - pathnew_matlab_central which links a set of functions and several data sets to the current path including:
 - runGUI – the set of files for creation of GUIs for the various speech apps
 - speech_files – set of files used to demonstrate matlab apps capabilities
 - functions_lrr – set of matlab functions used by the various matlab apps
 - high pass filter signal – filtering function for speech signals
 - isolated digit files – training and testing files for isolated digit recognition
 - cepstral coefficients – used for vector quantization matlab app
- Be sure to put each of the downloaded folders in the chosen directory so that the path links will be consistent

Summary

- Set of about 60 MATLAB speech processing exercises
- Exercises aligned with distinct sections in the textbook TADSP by Rabiner/Schafer
- Each exercise has an associated Graphical User Interface created using a GUI LITE program and created expressly for these speech processing exercises
- GUI LITE design and implementation Callbacks are in totally separate code packages