Goals:

1. Learn how to use the linear predictive model for speech analysis and synthesis.
2. Implement a linear predictive model for speech coding.

Note: The symbol means to put your discussion, flowchart, block diagram, or plots in your report. The symbol indicates that you should put multimedia data in your report. Save speech in signed mono16 bits WAV format unless otherwise stated. The symbol means to put your source codes (Matlab or C/C++) in your report.

I. Analyzing Speech Signals

To analyze a speech signal, we should first understand the human vocal tract and build a model to describe it. In this part, we investigate the linear predictive model.

The figure above shows the mid-sagittal plane of human vocal apparatus. The vocal tract begins at the opening of the vocal cords and ends at the lips. The figure below represents a model of speech production.

1. [http://www.phon.ox.ac.uk/~jcoleman/phonation.htm](http://www.phon.ox.ac.uk/~jcoleman/phonation.htm)
This model consists of two parts. The first part is the excitation signal which switches between two states: an impulse train, and white noise. The impulse train generator produces impulse trains at specified pitches for voiced sounds and the white noise generator produces noise for unvoiced sounds. The impulse train generator is parameterized by a given pitch period (i.e., the fundamental frequency of the glottal oscillation). The second part is the vocal tract model (with gain $G$) that is usually modeled as the following $p^{th}$ order all-pole linear predictive model, $V(z)$.

$$V(z) = \frac{G}{\prod_{k=1}^{p} (1 - \alpha_k z^{-k})}$$

The frequency parts of $1 - \alpha_k z^{-k}$ are called formants, which are the resonant frequencies caused by airflow through the vocal tract.

We will use the Matlab COLEA toolbox to study the linear predictive model. Download the COLEA toolbox from the following location, and unzip the files into a local directory. ([http://www.utdallas.edu/~loizou/speech/colea.htm](http://www.utdallas.edu/~loizou/speech/colea.htm))

There are four major window interfaces in the COLEA toolbox.

**Time-Domain Waveform:** This window shows the raw speech signal. We can observe the signal, `bode.wav`, in the time domain by selecting Display $\Rightarrow$ Time Waveform.

![Time-Domain Waveform](image)

**Spectrogram:** The spectrogram (short-time Fourier transform) is a popular spectral representation which allows us to view special properties about a speech signal in the time-frequency domain. To display the spectrogram, select Display $\Rightarrow$ Spectrogram $\Rightarrow$ Color.

![Spectrogram](image)

**Pitch and Formant tracking:** We can use Display $\Rightarrow$ F0 contour $\Rightarrow$ Autocorrelation Approach and Display $\Rightarrow$ Formant Track to visualize the pitch and formant of a speech signal.
Pitch Contour

Formant tract

LPC spectra: We can also characterize the spectrum of a speech signal using the linear predictive model, $V(z)$. For example, we first open a speech signal file (e.g., bode.wav). Click on the waveform or spectrogram in the corresponding window. Two more windows will appear. One is the LPC Spectra and the other one is Controls, which can be used to set the parameters of the displayed LPC spectra.

To verify the correctness of LPC for speech modeling, we can calculate the short-time Fourier transform (STFT), overlap it with the LPC spectrum, and compare how close those two spectrums are. We can choose the Spectrum as FFT on the Controls window and select Overlay on the bottom of the window to compare them.
1. Open the *Recoding Tool* (as shown in the figure below, found in the menu bar under *Record*). Record your own voices with ten vowels listed in the table below. **

Then, as shown in the table below, using the COLEA toolbox, *analyze the pitches and their first three formants for each of the ten vowels.* **

<table>
<thead>
<tr>
<th>Word/Vowel</th>
<th>Pitch (Hz)</th>
<th>Formant 1 (Hz)</th>
<th>Formant 2 (Hz)</th>
<th>Formant 3 (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>bgt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>bat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>but</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>hot</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>bought</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>foot</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>boot</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>bird</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To locate the position of a specified vowel in a speech signal, you can search for it by listening to a small frame in this speech signal. To specify a small frame, you can use left and right mouse clicks on the waveform window to set the start and the end of a frame, respectively. To listen to this small frame, press the "set" button in the *Play* area.

**Plot the first formant (X-axis) and second formant (Y-axis) of each vowel for all the members in your group in a single figure.** (See the following figure.) **Discuss what you observe from this figure.**
Adjust the order and duration of linear predictive model. Describe what you have observed from the LPC spectral results and STFT for different orders and durations.

2. As discussed in lecture, the linear predictive model is widely used in digital speech processing due to its simplicity and effectiveness. In this part, we use the linear predictive model to implement gender identification. You should develop your own algorithms using Matlab to identify the gender of a speaker. Ten male speech signals and ten female speech signals are provided on the course web page. You can train your gender identifier with those signals. At the end of this lab, you will be asked to test your program with a new set of signals.

The figure below shows the LPC gender identification framework. There are three building blocks in this system: LPC Analysis, Feature Extraction for training set, and Gender Identification testing.

LPC Analysis: Download and extract the Auditory Toolbox for Matlab (http://rvl4.ecn.purdue.edu/~malcolm/interval/1998-010/). Using the toolbox function proclpc.m, we can obtain LPC coefficients and other information.

```matlab
% [aCoeff,resid,pitch,G,parcor,stream] = proclpc(data,sr,L,fr,fs,preemp)
% % LPC analysis is performed on a monaural sound vector (data) which has been
% % sampled at a sampling rate of "sr". The following optional parameters modify
% % the behaviour of this algorithm.
% % L - The order of the analysis. There are L+1 LPC coefficients in the output
% % array aCoeff for each frame of data. The default value is 13.
% % fr - Frame time increment, in ms. The LPC analysis is done starting every
% % fr ms in time. Defaults is 20ms (50 LPC vectors a second)
```
% fs - Frame size in ms.  The LPC analysis is done by windowing the speech
% data with a rectangular window that is fs ms long.  Defaults to 30ms
% preemp - This variable is the epsilon in a digital one-zero filter which
% serves to preemphasize the speech signal and compensate for the 6dB
% per octave rolloff in the radiation function.  Defaults to .9378.
% The output variables from this function are
% aCoeff - The LPC analysis results, a(i).  One column of L numbers for each
% frame of data
% resid - The LPC residual, e(n).  One column of sr*fs samples representing
% the excitation or residual of the LPC filter.
% pitch – A vector of frame-by-frame estimate of the pitch of the signal, calculated
% by finding the peak in the residual’s autocorrelation for each frame.
% G - The LPC gain for each frame.
% parcor - The PARCOR coefficients.  The PARCOR coefficients give the ratio
% between adjacent sections in a tubular model of the speech
% articulators.  There are L PARCOR coefficients for each frame of speech.
% stream – A vector representing the residual or excitation signal in the LPC analysis.
% Overlapping frames of the resid output combined into a new one-
% dimensional signal and post-filtered.

The following diagram illustrates how this M-file works.

Note: The output variable pitch indicates the pitch period, not the pitch frequency. The output variable parcor contains the PARCOR coefficients for each frame of the speech signal, calculated using the partial autocorrelation of the signal. These coefficients are the coefficients of the equivalent lattice realization of the predictor, and they are less sensitive to round-off noise and coefficient quantization. You may find that not all of the outputs from proclpc.m are necessary to complete the current task.

Feature Extraction for training sets: For each signal, we can obtain one set of coefficients. Develop your own algorithm as a Matlab function to distinguish gender using those coefficients. Briefly explain how your algorithm works.

Note 1: Use the Matlab function TIMITread to read these wave files. You may need to multiply the signals by $2^{-15}$ in order to scale the amplitude of these signals to fall in the interval [-1, 1].

Note 2: The unvoiced signals in the speech files may affect your identification performance.

Testing new voice files: Your algorithm will be tested with ten new signals and your scores for this part depends on the percentage of the correct identification by your gender identifier.
II. Speech Coding: Linear Predictive Vocoder

The linear predictive vocoder (vocoder) is a popular tool for speech coding. To efficiently encode a speech signal at low bit rate, we can employ an analysis-synthesis approach. In this part, we design a 2.4 kbps $10^{th}$-order linear predictive vocoder according to the linear predictive model we studied earlier.

We have already learned how to obtain LPC related parameters, such as the 10 LPC coefficients, \( \{a_k\}_{k=1}^{10} \) (aCoeff), as well as the gain (G) and pitch (pitch) for each frame using proclpc.m. We can represent those parameters as a vector \( \mathbf{A} = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, \text{Gain}, \text{Pitch}) \) and quantize \( \mathbf{A} \) to compress a speech signal. After reconstructing a speech signal from those quantized parameters, we can use synlpc.m, (also in the Auditory Toolbox) to synthesize this speech.

\[
\% \text{synWave} = \text{synlpc(aCoeff,source,sr,G,fr,fs,preemp)};
\%
\% \text{LPC synthesis to produce a monaural sound vector (synWave) using the following parameters}
\%
\% aCoeff - represents the LPC analysis results, \( a(i) \). Each column of \( L+1 \) numbers for each
\% frame of speech data. The number of columns is determined by the number of frames in the
\% speech signal
\%
\% source - represents the LPC residual, \( e(n) \). One column of \( sr*fs \) samples representing
\% the excitation or residual of the LPC filter.
\%
\% sr - sampling rate
\%
\% G - The LPC gain for each frame.
\%
\% fr - Frame time increment, in milli-seconds (ms). The LPC analysis is done for every
\% fr ms in time. Default is 20ms (i.e., 50 LPC vectors a second).
\%
\% fs - Frame size in ms. The LPC analysis is done by windowing the speech
\%
\% data with a rectangular window that is \( fs \) ms long. Default is 30ms (i.e., allow 10 ms’ overlap
\%
\% between frames).
\%
\% preemp - This variable is the epsilon in a digital single-zero filter that
\%
\% is used to preemphasize the speech signal and compensate for the 6dB
\%
\% per octave rolloff in the radiation function. Default is 0.9378.
\]

In proclpc.m, if pitch is equal to zero, it means this frame is unvoiced (UV). If it is nonzero, pitch indicates that this frame is voiced (V) with pitch period, T. To avoid confusion with the meaning of pitch, we denote the value of pitch as UV/V/T.

Line Spectrum Pair: If we directly quantize LPC coefficients \( (a_1 \sim a_{10}) \), some of the poles located just inside the unit circle before quantization may shift outside the unit circle after quantization, causing instability. One way to overcome this problem is to convert the LPCs to Line Spectrum Pair (LSP) parameters which are more amenable to quantization. LSPs can be calculated first by generating polynomials \( P(z) \) and \( Q(z) \):

\[
P(z) = 1 + (a_1 - a_{10})z^{-1} + (a_2 - a_9)z^{-2} + \ldots + (a_{10} - a_1)z^{-10} - z^{-11}
\]

\[
Q(z) = 1 + (a_1 + a_{10})z^{-1} + (a_2 + a_9)z^{-2} + \ldots + (a_{10} + a_1)z^{-10} + z^{-11}
\]

Then, rearrange \( P(z) \) and \( Q(z) \) to obtain parameters \( \{w_k\} \):

\[
P(z) = (1-z^{-1}) \prod_{k=2,4,...,10} (1-2\cos w_k z^{-1} + z^{-2})
\]

\[
Q(z) = (1+z^{-1}) \prod_{k=1,3,...,9} (1-2\cos w_k z^{-1} + z^{-2})
\]

where \( \{w_k\}_{k=1}^{10} \) are the LSP parameters with order of \( 0 < w_1 < w_2 < \ldots < w_{10} < \pi \).
We can use the Matlab function `lpcar2ls.m`, located in the speech processing toolbox “Voicebox” (http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html), to convert LPC (AR) parameters to LSP parameters and use `lpcls2ar.m` to convert LSP back to LPC (AR).

Quantization: To achieve a coding rate of 2.4 kbps with a frame size of 20 msec (i.e., 50 frames per second), each frame will be represented by 48 bits. The following table shows how to allocate bits for the above-mentioned parameters of \{w_k\}, Gain and UV/V,T.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Bits/frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_1</td>
<td>3</td>
</tr>
<tr>
<td>w_2</td>
<td>4</td>
</tr>
<tr>
<td>w_3</td>
<td>4</td>
</tr>
<tr>
<td>w_4</td>
<td>4</td>
</tr>
<tr>
<td>w_5</td>
<td>4</td>
</tr>
<tr>
<td>w_6</td>
<td>3</td>
</tr>
<tr>
<td>w_7</td>
<td>3</td>
</tr>
<tr>
<td>w_8</td>
<td>3</td>
</tr>
<tr>
<td>w_9</td>
<td>3</td>
</tr>
<tr>
<td>w_10</td>
<td>3</td>
</tr>
<tr>
<td>Gain</td>
<td>7</td>
</tr>
<tr>
<td>UV/V,T</td>
<td>7</td>
</tr>
</tbody>
</table>

We can employ non-uniform quantization to minimize the reconstruction error. The usefulness of non-uniform quantization relies on our knowledge of the probability distribution of the data. In other words, if we know that the value of a certain coefficient (e.g., w_1, w_2, etc.) falls within a certain interval with high probability, we can perform dense sampling within the interval and sparse sampling outside the interval.

We assign seven bits (range of value: 0~127) for the parameter UV/V,T. If this frame is unvoiced (UV), we encode it as (0)_10 or (0000000)_2. Otherwise for the voiced case (V), we encode the corresponding pitch period, T, according to the following table. For example, if T is equal to 22, we encode it as (3)_10 or (0000011)_2.

<table>
<thead>
<tr>
<th>UV/V</th>
<th>T</th>
<th>Encoded Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UV</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>V</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>V</td>
<td>21</td>
<td>2</td>
</tr>
<tr>
<td>V</td>
<td>22</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>V</td>
<td>146</td>
<td>127</td>
</tr>
</tbody>
</table>

1. Design your own 2.4 kbps vocoder. The figures below show the encoder and decoder, respectively. **Write Matlab functions to implement the encoder and decoder.** Briefly explain your design.
Note that the encoder and decoder should be implemented separately. That is, the encoder reads a wave file using `wavread.m`, generates a compressed bit stream, and saves it in the storage. The decoder reads this compressed bit stream from disk and decompresses it into a wave file using `wavwrite.m`. You can use `proclpc.m`, `synlpc.m`, `lpcar2ls.m` and `lpcls2ar.m` as the basic building blocks.

Then, design the quantization scheme for the LPC parameters $A'=(w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10}, \text{Gain}, \text{UV}/\text{V}, \text{T})$ and design the impulse train generator for voiced state and white noise for unvoiced speech. You are welcome to use a non-uniform quantization scheme, if you wish.

Compress the speech signal stored in `tapestry.wav`. Calculate the mean squared error for the reconstructed speech signal and the original speech signal.

2. Code Excited Linear Prediction (CELP) is a federal speech-coding standard (FS-1016) that also uses linear prediction. This standard offers good speech compression at intermediate bit rate (4.8-9.6 kbps). Unlike LPC coding, CELP coding does not distinguish explicitly between voiced and unvoiced segments. Furthermore, CELP is a hybrid coding scheme because the prediction residue is also encoded along with the prediction coefficients to improve voice quality. In this part, we will compare speech signals compressed using an LPC vocoder and a CELP vocoder.
Download the following three files from the course website: ulaw.au (original signal), lpc.au (LPC-encoded, 4.8 Kbps), and celp45.au (CELP-encoded, 4.5 Kbps). These files are actual voice samples that were compressed and decompressed with the codecs included with HawkVoice, a free speech compression library. (http://www.hawksoft.com/hawkvoice/codecs.shtml)

Load each of the three signals into the Matlab workspace using the function auread.m. Using the sound command, listen to all three signals. **Compare the sound quality of the LPC-encoded and CELP-encoded signals.**

Use the following commands to view an excerpt of each signal:

```matlab
gg figure; plot(orig(500:4400)); figure; plot(lpc(500:4400)); figure; plot(celp45(500:4400));
```

**Compare the nature of the LPC-encoded and CELP-encoded signals. What do you notice about the speech segments in each signal? What aspects of the two coding schemes cause these differences in the signals? How might this affect sound quality?**

### III. Speech Recognition

The Microsoft Speech Recognition Engine (SRE) is a recently released commercial speech recognition software. In this part, we use the SRE to get some experience with the state of the art in speech recognition.

1. **Training:** Connect your USB headset to the computer. Be sure the playback and recording settings are properly configured. To open the Microsoft SRE, open Microsoft Word, and select **Tools → Speech.** First, follow the steps presented to calibrate your microphone. Next, as a new user, you are required to train this software. Follow the instructions provided. Be sure that you are in a quiet environment during training.

2. **Once the SRE has been trained and activated, we can start operating Microsoft Word with our voice! (Of course, you are still able to use the keyboard and mouse whenever necessary.)** To enable access to the menu bar, say “voice command.” Try to perform common tasks such as opening a new file, closing a document, cutting and pasting, formatting text, selecting text, or any other task you can think of. **Briefly assess the capabilities and ease-of-use in using your voice to operate Microsoft Word.**

3. **Now, use the SRE to perform a short dictation on Microsoft Word.** To activate the dictation tool, simply say “dictation.” Visit the following link to the first chapter of *A Tale of Two Cities* by Charles Dickens: (http://www.litrix.com/twocitys/twoci001.htm#1)

Use the dictation tool to write the first paragraph of Chapter One in a new Microsoft Word document. In your report, **include the SRE’s interpretation of your dictation.** **Discuss the strength and weaknesses of this speech recognition system.** Perhaps you may wish to perform additional training, repeat the dictation, and compare the results. You are welcome to experiment with any other text passages, too.
IV. Speech Synthesis

Speech synthesis systems are generally classified into two categories: concept-to-speech systems and text-to-speech (TTS) systems. Concept-to-speech system is used by automatic dialog machine that has a limited vocabulary, e.g. 100 words, but with artificial intelligence to respond to inputs. Text-to-speech system aims at reading text (e.g. aids for the blind) and has the ability to record and store all the words of a specified language. In this part, we explore the TTS system and implement (a simple) speech synthesis system.

1. Vowel Synthesis: In practice, designing a text-to-speech system is not a simple task. This system should consist of several levels of processing: acoustic, phonetic, phonological, morphological, syntactic, semantic, and pragmatic levels. In this part, we emphasize on the phonetic level and synthesize the vowel by `MakeVowel.m` that is also provided with the Auditory Toolbox.

   \[
   \% y = \text{MakeVowel}(\text{len}, \text{pitch}, \text{sampleRate}, f_1, f_2, f_3)
   \]

   \% len: length of samples
   \% pitch: can be either a scalar indicating the actual pitch frequency, or an array of impulse locations. Using an array of impulses allows this routine to compute vowels with time-varying pitch.
   \% sampleRate: sampling rate
   \% f1, f2 & f3: formant frequencies

Synthesize those ten vowels that you have provided in Part I.1 by putting the values of pitch, f1, f2, and f3 in the input argument of `MakeVowel.m`. Use the Matlab function `sound(y, sampleRate)` to hear those synthetic vowels. Use `wavwrite.m` to write out the wave files.

Compare the vowels you recorded in Part I.1 with the synthesized results.

2. Microsoft Windows Narrator (for further exploration): The Narrator tool (available in Windows XP) recognizes text and uses synthesized speech to recite the content of a window. Such a tool may be useful for the visually impaired.

In the Start Menu, select Programs → Accessories → Accessibility → Narrator. You can change the voice properties of the narrator by selecting Voice in the narrator window. As you switch from window to window, the narrator will recite the content of each window. Visit the link in Part III.3 and assess the quality of the narrator as it recites the text, both in terms of voice quality and usability. For more information, visit the Microsoft website. (http://support.microsoft.com/kb/306902/)
V. Human-Computer Interface

One of the motivations to analyze and synthesize speech is to create a friendly and convenient interface between users and computers. Though such a concept-to-speech system, users can operate and communicate with machine through voice. In this part, we explore two advanced human computer interfaces based on speech recognition.

MIT Galaxy System: The MIT Spoken Language Systems Group has been working on several research projects in human-computer interfaces via telephone. The following applications are available here: (http://www.sls.lcs.mit.edu/sls/applications/). These are the available applications:

- JUPITER - A weather information system
- MERCURY - An airline flight scheduling and pricing system
- PEGASUS - An airline flight status system
- VOYAGER - A city guide and urban navigation system
- ORION - A personal agent for automated, off-line services

See the instructions on the web sites of JUPITER and PEGASUS. Dial the corresponding toll-free phone numbers and talk with these two systems. Describe under what kinds of conditions these systems will make mistakes.

VI: Mobile Computing and Pocket PC Programming

We have learned various aspects of digital speech processing in this design project. Apply what you have learned from the previous parts and design a simple application related to digital speech processing for Pocket PC using Microsoft eMbedded Tools. You can refer to “ENEE408G Multimedia Signal Processing Mobile Computing and Pocket PC Programming Manual” and extend the examples there. Please include sufficient documentation along with your source code and an executable.