Laboratory Exercise 1

Speech Modeling, Analysis, Synthesis and Compression

Multimedia and Video Communications (SSY150)

Prepared by: Irene Y.H. Gu, irenegu@chalmers.se
Department of Electrical Engineering,
Chalmers University of Technology,
41296, Göteborg, Sweden

Updated in March, 2018

1 Introduction: Basics and Theories for Speech Signal Processing

1.1 Mechanism for Human Sound Production

The human vocal organ starts at the lungs and ends at the mouth and nostrils. As shown in Figure 1, the parts of this organ form a connected tube. The upper part is the so-called vocal tract, with changing shape due to movement of the jaw, tongue, lips and inside the mouth. The nasal cavity can be separated from the rest of the vocal tract by raising of velum. Human sounds are produced through some excitation source, causing the air in the vocal tract to vibrate. These vibrations are then formed by the acoustic properties of the vocal tract into specific sounds. Speech has generally three different sources of excitation mechanisms that cause these vibrations:

1. The first type is when the vocal cords are strained, and air from the lungs is pushed through the gap between them, called the glottis. The interaction between air and vocal cords causes opening and closing of the glottis, which can also be seen as the vibration of the vocal cords. The periodical interruption of the airflow is the source of excitation, and the waves caused can be seen as asymmetrical triangular waves. The frequency of these waves can be adjusted by how strongly strained the vocal cords are, and the pressure from the lungs. Higher pressure and more strongly strained vocal cords give higher frequencies.

2. The second type of excitation is caused when airflow from the lungs passes a constriction in the vocal tract, and becomes turbulent. This excitation is noise-like.

3. The third type of excitation is caused when airflow from the lungs builds up a pressure, behind a point of total closure in the vocal tract, which is then suddenly released. This causes a transient type of excitation.

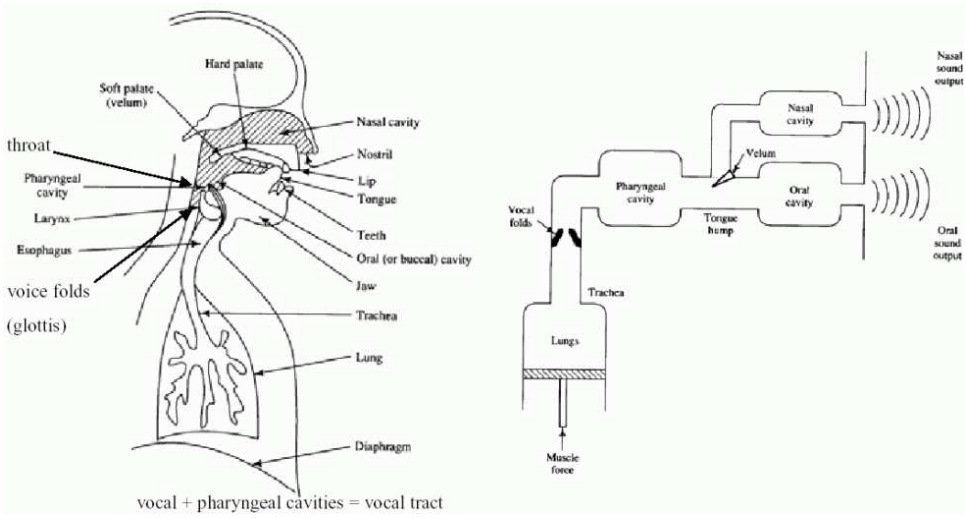

Figure 1: Mechanism of Human Sound Production. Left: the lung, pharyngeal, nasal and oral cavities as the vocal tract; (b) the tube model for sound production. (Courtesy of Dr. Roger Jang [4])

The sounds on which a language is built upon are called *phonemes*. Phonemes are used by linguists to represent the different speech sounds, e.g. with written symbols a, s, b. Phonemes are commonly grouped into 2 different classes, depending whether or not they contain the glottal excitation source. The first type in the above list contains the glottal excitation, and is referred to as the *voiced* sounds, and the remaining two types are referred to as the *unvoiced* sounds. However, a phoneme can contain more than one excitation source, e.g. t, containing both glottal and turbulent excitation. Roughly speaking we can categorize vowels as the voiced sound, and consonants as the unvoiced sound.

1.2 A Mathematical Model for Speech Production

A simplest model for speech production is depicted in Figure 2. In the model, the vocal tract is modeled by a linear time-varying all pole filter, and the vocal cord excitation is described by the input sequence $e(n)$ to the filter, which is either an impulse sequence (for the voiced speech), or white noise (for the unvoiced speech). The output of the filter is the speech signal $s(n)$.

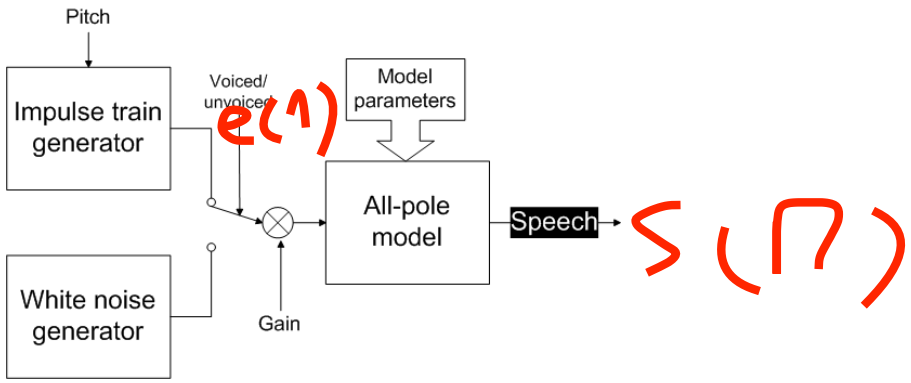

Figure 2: Basic mathematical model for speech production.

Let us consider a *stationary* speech signal, for example, a single tone speech, or a vowel or a consonant. The system function of the all pole filter that resembles the vocal tract shape is thus,

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 + \sum_{j=1}^{p} a_j z^{-j}} \tag{1}$$

where a_i is the model parameter, and p is the model order, and $H(z) = \frac{S(z)}{E(z)}$. From (1), the speech signal can be obtained through the linear prediction

$$s(n) = -\sum_{j=1}^{p} a_j s(n-j) + e(n) \tag{2}$$

where $e(n)$ is the input to the system.

Selecting the model order p: According to the speech production model, speech spectra typically include 3-4 formants (i.e., 3-4 spectral resonants) that are sufficient to characterize a speech. As each resonant peak in the spectrum is related to one pair of poles, a typical (empirical) choice of model order for the speech signal is $p = 10$ (when a sampling rate f_s is 8kHz) [1]. For audio signals, both the model order and the sampling rate are much higher.

Computing the residuals $e(n)$ and the variance: Once the model parameters are estimated from $s(n)$, the residual sequence $e(n)$, $n = 1, 2, \cdots$, can be estimated by using the inverse filter of $1/H(z) = A(z)$ with $s(n)$ as the input:

$$E(z) = A(z)S(z) \quad \Rightarrow \quad \hat{e}(n) = s(n) + \sum_{j=1}^{p} \hat{a}_j s(n-j) \tag{3}$$

Further, the variance of the residual sequence σ_e^2 in each block shall be computed since the energy of the filter input cannot generally be assumed to be a unit value (Noting, σ_e is equivalent to the gain G in Figure 2). The input of the filter is either an impulse sequence (for the voiced) or white noise (for the unvoiced). Therefore, the speech signal ($s(n) = Z^{-1}\{S(z)\}$) shall be obtained by multiplying the gain G

with the filter transfer function $S(z) = G \cdot H(z) = \frac{|b(0)|G}{A(z)}$, where $|b(0)| = 1.0$.

Vocal cord excitation for voiced and unvoiced speech:
Roughly speaking, the voiced speech is mainly related to vowels (e.g.'a','o'), while the unvoiced speech is mainly related to consonants (e.g. 'p','b','t'). For voiced speech signals, one can clearly observe the quasi-periodicity in the waveform. For unvoiced speech signals, the waveform looks more like noise. Figure 3 shows examples of voiced and unvoiced speech.

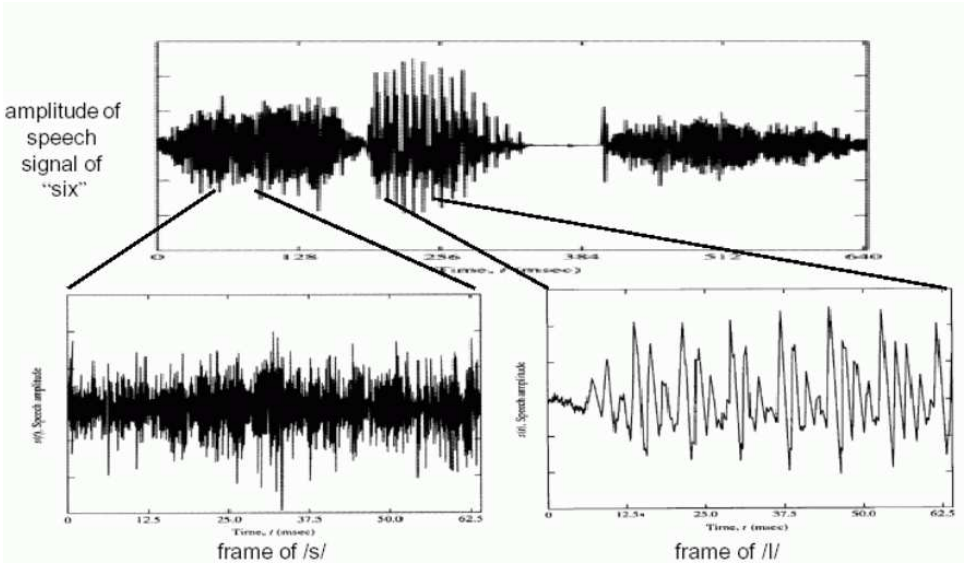

Figure 3: Typical waveforms of voiced and unvoiced speech: The speech signal is 'six', and the bottom left small plot shows the unvoiced sound 's' (noise-like), whereas the bottom right plot shows the voiced sound 'i' (quasi-periodic) (Courtesy of Dr. Roger Jang).

There are several ways that one may generate the input sequence as the vocal cord excitation of a speech.

1. A simplest way is to use the excitation model in Figure 2: for voiced speech, an impulse sequence with the pitch period interval is used; for unvoiced sequence, a white noise sequence is used.

2. A more advanced way is to use a codebook, where several prominent residual values in each speech block are picked up and matched to a codeword in an existing codebook. This is related to a so-called CELP (code-excited LPC) speech model (see Figure 4). This can be considered as an improvement to the voiced/unvoiced excitation model, since it is not always possible to draw a clear boundary between the voiced and unvoiced speech, especially near the transition between the voiced and unvoiced regions.

Speech compression: When discussing speech compression, we are usually interested in the "lossy compression" techniques. However, it is desirable that the compression is done such that the degradation of decoded speech quality after the compression is almost non-perceivable by the human auditory system.

It is not difficult to understand how compression can be achieved by using the speech modeling. For example, in the LPC speech model, each block of speech is described by a set of parameters $\{a_j, j = 1, \cdots, p = 10, \sigma_e^2\}$, plus an extra parameter to describe the pitch period. Assuming each parameter is encoded in 10 bits, we need 12x10=120 bits for encoding each block of speech. However, if directly encoding the speech waveform, e.g. a 20ms block of speech sampled at 8kHz, we need to encode 160 samples. We need 160x8=1280 bits for each block if each sample is encoded in 8 bits. This has led to an approximate compression ratio of 10.

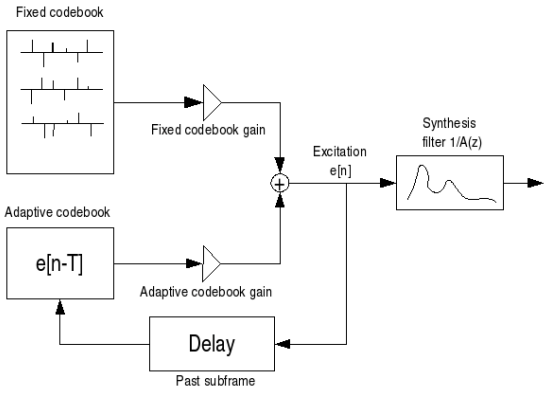

Figure 4: Code-Excited LPC (CELP) speech model. (courtesy of Wikipedia, the free encyclopedia [5])

1.3 Block-based processing of speech signals

Most speech signals are nonstationary if we wish to make meaningful speech sentences. We have learned from the signal processing course that one way to process nonstationary signals is to divide data into blocks (or frames), each of which can then be considered as stationary. After that, the well established signal processing theories for stationary signals can be applied. This is exactly the way that we are going to process the nonstationary speech signal: First, we shall divide the speech signal into blocks, each is of 5-20ms duration, such that the speech signal within each block can be considered as stationary. We then model the vocal tract by a time-invariant all-pole filter within each block, and the vocal cord excitation as either the impulse or white noise.

Block-based analysis: After dividing the speech signal into non-overlapped blocks, each contains L samples, where $L = BlockSize(second) * f_s$, and the total number of blocks is $\lfloor (length(SpeechFile)/L) \rfloor$. We then apply the LPC analysis to the speech signal within each block. This results in the estimated parameters associated with that block, i.e. $\hat{a}_j^{(i)}$ for i-th block. This process is then repeated for all blocks, resulting in the estimated LPC parameters and the variance of errors $\{\hat{a}_j^{(i)}, \sigma_e^{2(i)},\ j = 1, \cdots, p,\ i = 1, \cdots, \text{TotalBlocks}\}$.

Compute the residual sequence: When computing the residual sequence, the inverse filter $\frac{1}{H(z)} = A(z)$ (the corresponding parameters are estimated from the analysis) is used with the original speech $s(n)$ as the input. Comparing with the way in calculating the residuals in the stationary case, the only difference here is that when computing the residuals near the boundary of each block, one cannot simply assume the speech data outside the block is zero (this is only true for the first block), rather the continuity of speech must be taken care of. For the i-th block, the residuals can thus be computed by

$$e(z) = s(z)(1 + \sum_{j=1}^{p} a_j^{(i)} z^{-j}) \quad \Longrightarrow \quad \hat{e}(n) = s(n) + \sum_{j=1}^{p} \hat{a}_j^{(i)} s(n-j) \tag{4}$$

where computing the first p residual samples in a block requires to use the original speech samples from the previous block, further the estimated parameters are used to replace the true ones, leading to $\hat{e}(n)$ instead of $e(n)$.

Re-synthesize the speech:
(a) Use LPC residuals as the input of the filter
Once the LPC parameters and the residuals (or, the voiced (including estimated pitch periods) / unvoiced information) are estimated, one can proceed with the speech synthesis. For a given block i, the exact

speech signal $\hat{s}(n)$ can be synthesized if one uses the residuals as the excitations,

$$\hat{s}(n) = -\sum_{j=1}^{p} \hat{a}_j^{(i)} \hat{s}(n-j) + \hat{e}(n) \tag{5}$$

It is worth noting that for synthesizing the speech, the parameters $\hat{a}_j^{(i)}$ should be chosen from the corresponding block i. Also, synthesizing the first p samples in the i-th block requires the use of a few synthesized samples from the previous block $(i-1)$. This should be taken care of, so that the continuity of the synthesized speech signal will be maintained. To reduce the block effect, one may multiple the block data, e.g. by a hamming window, or one may use overlapped blocks.

(b) Use the estimated voiced/unvoiced information, and pitch period as the input
From whether the signal is quasi-periodic or aperiodic and noise-like, one may decide whether the speech block is voice or unvoiced. If it is a voiced block, then the pitch period needs to be estimated. If it is unvoiced, then the error variance needs to be estimated (which is estimated during the LPC analysis).

Range of pitch frequencies: On average, women have higher pitch frequencies (or, fundamental frequencies of speech) than men. Statistics show that the range of pitch frequencies f_p for men is approximately $\mathcal{F}_{men} \in [62, 535]$ Hz, and $\mathcal{F}_{women} \in [110, 1000]$ Hz for women. The corresponding range of pitch periods T_p (samples) can be computed by $T_p = f_s/f_p$, where f_s is the sampling rate of speech signal.

1.4 Methods for automatically estimating pitch periods

There are many different techniques that can be applied for estimating the pitch period. Accurately estimating the pitch period for each block of voiced speech is a non-trivial task, and is beyond the scope and the aim of this laboratory exercise. Below, we shall only briefly describe two basic methods that are simple to implement.

- pitch estimation by using ACF (autocorrelation function);
- pitch estimation by using cepstrum domain analysis.

ACF-based method

Set an analysis window size, e.g. L=512 samples. Compute the autocorrelation function in that block using all data samples within the block,

$$R_s(\tau) = \sum_{j=(i-1)L+1}^{iL} s(j)s(j+\tau), \qquad \tau = 1, 2, \cdots, L-1 \tag{6}$$

Chose the time lag value τ such that it corresponds to the first largest $R_s(\tau)$, where τ is within the range of possible pitch periods of men/women (i.e. $\frac{1}{\tau} \in \{\mathcal{F}_{men}, /\mathcal{F}_{women}\}$), resulting in an estimate of the pitch period. Often, pitch period estimate based on an individual block is unreliable. Since the frequency of vocal cord excitation cannot change abruptly, pitch periods usually change slowly through the blocks. This make it possible for further improve the estimated pitch by using inter-block pitch correlation. One may notice that peaks obtained by directly applying ACF to speech signal itself may not be always easy for estimating the pitch detection. Instead, one may compute the ACF for the residual sequence,

$$R_s(\tau) = \sum_{j=(i-1)L+1}^{iL} \hat{e}(j)\hat{e}(j+\tau), \qquad \tau = 1, 2, \cdots, L-1 \tag{7}$$

where the peaks in the ACF could be more prominent for detecting pitch.

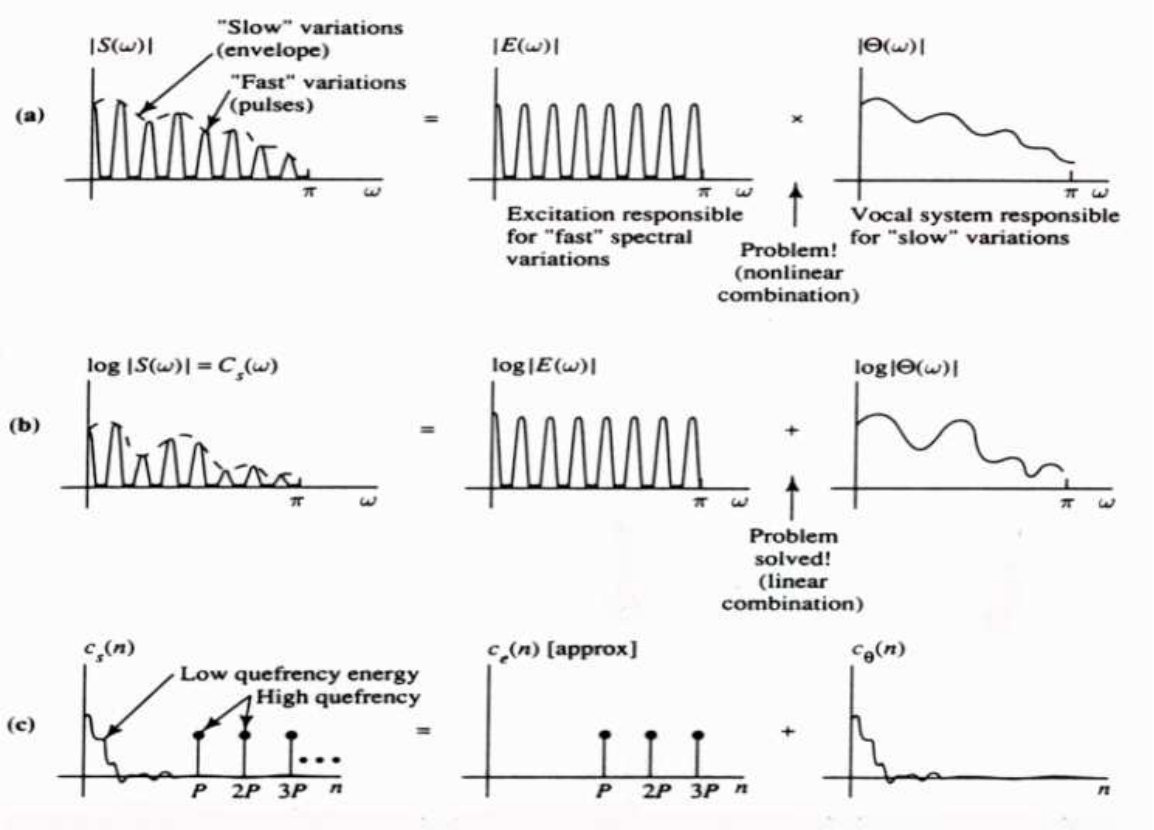

Figure 5: Cepstrum for estimating the pitch period of speech (Courtesy from Dr.Roger Jang). Row-1: speech magnitude spectrum $|S(\omega)|$ can be considered as the product of two components: the fast variation part $|E(\omega)|$ (2nd column) and the envelop $|\theta(\omega)|$ (3rd column); Row-2: By taking logarithm to $|S(\omega)|$, multiplication becomes addition of the two terms; Row-3: Due to the nature of two components, applying an inverse FFT results in two cepstral components, each occupies a relatively separable range of time. By applying a rectangular time window, one can thus separate the functions of vocal track and vocal cord excitation, and detect the pitch period values from the cepstrum $c_e(n)$.

Cepstrum-based method:

Cepstrum is another technique that can be applied to estimate the pitch periods. Cepstrum is a nonlinear transformation, described by

$$\mathbf{c}_i = \left| FFT^{-1}\{log(|FFT(\mathbf{x}_i)|\}\right| \tag{8}$$

where $\mathbf{x}_i(n) = s(L*(i-1)+1 : L*i)$ contains speech samples within the i-th block. The basic idea is that the magnitude of FFT spectrum of speech contains the product of two components: one is related to the fast periodic variation in the frequency domain $|E(\omega)|$, and another is related to the slow envelop variation in the frequency domain $|\theta(\omega)|$ (see Figure 5). By taking the logarithmic operation, the multiplication of these two components becomes the addition. Therefore, if applying an inverse FFT to $log(fft|\mathbf{x}(n)|)$, the fast periodic variation part becomes the line spectrum in a higher range of cepstrum $c_e(n)$ (related to the voiced excitation), and the slow variation part becomes the continuous spectrum in the low range of cepstrum $c_\theta(n)$ (related to the vocal tract filter). After applying a highpass window function in the cepstrum domain, we can see the first line in the cepstrum domain $c_e(n)$ is related to the pitch period (see the sub-figure in 3rd row and 2nd column of Figure 5). Figure 6 shows two examples that use the cepstrum to detect pitch periods. One speech segment is from a male speaker, another is from a female speaker.

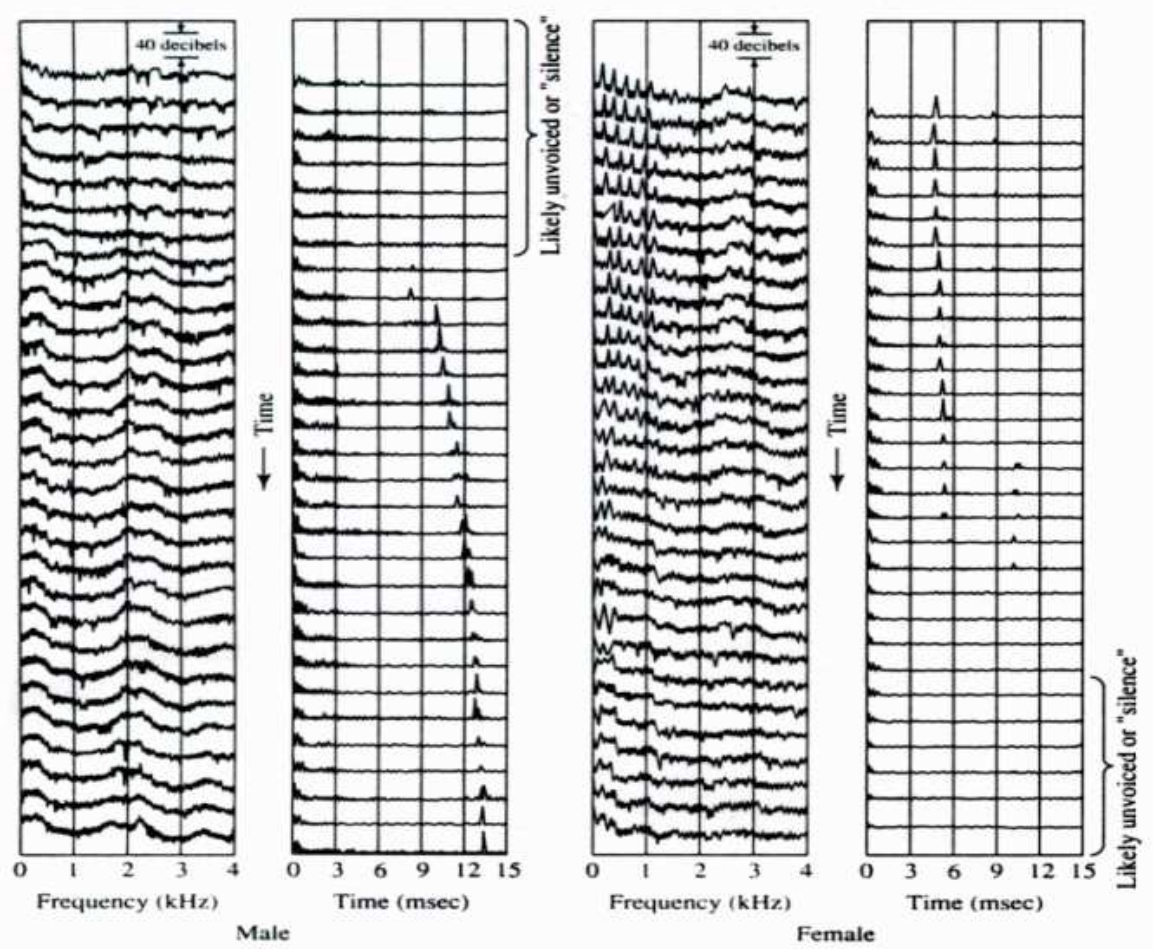

Figure 6: Resulting speech spectrum in the frequency domain and the cepstrum domain (Courtesy from Dr. Roger Jang). Columns 1 and 3: log magnitude spectrum $log(|S(\omega)|)$ in the frequency domain; Columns 2 and 4: cepstra $c_s(n) = \left|F^{-1}(log|S(\omega)|)\right|$ in the time domain, where the pitch periods are clearly visible.

References

[1] Lawrence R. Rabinar, Ronald W. Schafer, Digital Processing of speech signals, Prentice-Hall, Inc., 1978.

[2] John R., Jr. Deller, John H.L. Hansen, John G. Proakis, Discrete-time processing of speech signals, IEEE Press Classic Reissue, 1999.

[3] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck, Discrete-Time Signal Processing, 2nd edition, Prentice Hall, Inc. 1999.

[4] http://neural.cs.nthu.edu.tw/jang

[5] Wikipedia, the free encyclopedia on CELP: http://en.wikipedia.org/wiki/Code_Excited_Linear_Prediction

Laboratory Exercise

This laboratory exercise is aimed at hands-on learning on how digital speech signals can be synthesized and compressed by using model-based methods. Under the speech production model using LPC, a speech signal is divided into blocks, each is of 20 ms duration and is modeled by a time-invariant all-pole filter with an impulse sequence that either contains impulses with pitch interval or white noise. The laboratory exercise would enable you to obtain synthetic speech using the parameters you have estimated from speech analysis step, visualize the pitch periods from the residual sequence or the speech signal itself, listen to the synthetic speech, and compare the quality between the original and the synthetic speech. It will also enable you to touch the basics of the CELP speech encoder (see Figure 4) by using several prominent residuals in each block as the excitations, and the basics on pitch estimation methods.

Form of Laboratory Exercise, Report and Assessment

This laboratory exercise should be done by each group (maximum 3 students per group). After the laboratory exercise, each group will submit *one* short written report by the dead line (can be found in the course website). **The report shall include:**

- Results obtained from each step of your laboratory exercise (as specified below);
- **MATLAB codes you developed** (the codes for each step shall be clearly marked);
- A zipped file containing the speech files (original + re-synthesized) as mentioned in the required results in different steps.

Further, each student will attend an oral questioning/discussion session with the course teacher for the individual assessment.

Summary of Tasks

You are asked to make your Matlab programs to implement and test the following laboratory exercise tasks:

- Task-1: Record a (stationary) single vowel, and make Matlab programs for LPC analysis and synthesis of stationary speech;
- Task-2: Record a (nonstationary) speech sentence, and make Matlab programs for block-based LPC analysis and synthesis of nonstationary speech (using the residual sequence as the excitations);
- Task-3: Repeat the task 2, however, the excitations to the filter are replaced by using K prominent valued residuals in each block.
- Task-4: For the recorded speech sentence, decide whether each speech block is voiced or unvoiced. For those voiced blocks, using cepstrum to estimate the pitch periods.
- Task-5: For the synthetic speech, computing the spectral distance (between the synthetic and original speech) as the quality measure.

Task 1: Stationary (Single Tone) Speech Signals: Modeling, Analysis and Synthesis

Exercise:
Step (1.1) Generate a *stationary* speech signal file:

- Use a microphone to input a single tone sound (e.g. the vowel "a"), and record the sound in Matlab (set the sampling rate f_s=12kHz, sound duration=3 seconds, and the number of channels=1), and listen to the recoded sound.
- Save the recorded sound into a file (set the file name as, e.g., "MyVowel.wav").
- Read the saved file, and replay the sound again.
- Finally, plot the sound waveform in Matlab, zoom in on short segments to check the stationary.

Hint: useful Matlab functions: "audiorecorder", "audiowrite", "audioread", and "plot". Type *help "function-name"* in Matlab for detailed description of these functions.

Results to be included in the report:
1.1.a. Plot of the recorded sound waveform.
1.1.b. *.wav file of original single tone sound.

Step (1.2) Estimate the LPC model parameters:
Apply a LPC model of order p=12 to a speech block of 300ms (by cutting 300ms segment of speech from the recorded single tone speech in Step (1.1)), and estimate the corresponding model parameters $\hat{a}_j$, $j = 0, 1, \cdots, p$, and the error variance (var) using Eq.(1).
hint: use the Matlab function "LPC" to compute model parameters and the variance (power) of the prediction errors.

Results to be included in the report:
1.2.a. The number of samples contained in the block (block size = ms, f_s = kHz): L= samples
1.2.b. The estimated $H(z)$ LPC parameters are: $[a_j]_{j=1}^{p}$=
1.2.c. The variance of prediction errors =

Step (1.3) Calculate the residual sequence $e(n)$:
The residual sequence from the LPC predicted speech can be obtained by passing the speech signal through the inverse filter $A(z)$ using Eq.(3). Compute the residual sequence and display it.

Results to be included in the report:
1.3.a. Plot the waveform of the original speech signal, and the corresponding waveform of the residual sequence $\hat{e}(n)$.

Step (1.4) Re-synthesize the speech using the estimated parameters:
Use the residual sequence $\hat{e}(n)$ from Step (1.3) as the input to the all-pole filter $H(z)$, and compute the filter output (hint: using Eq.(1), replace a_i with the estimated $\hat{a}_i$). This is equivalent to using the vocal cord excitation to the vocal tract filter, resulting in a re-synthesize speech signal $\hat{s}(n)$.

Results to be included in the report:
1.4.a. Plot the waveforms of the original $s(n)$ and the re-synthesized speech signal $\hat{s}(n)$, and compare them.
1.4.b. *.wav file of re-synthesized single tone sound.

Task-2: Nonstationary Speech Signals: Modeling, Analysis and Synthesis

Exercise:
Step (2.1) Recording a "natural" speech signal containing a sentence.

Repeat the process as that in the Step (1.1) - however, this time recording a natural speech sentence (with a duration of about 15 seconds). Play back the sound and check the record, and then save it into a new speech file named "MySentence.wav".

Results to be included in the report:
2.1.a. original speech *.wav file.

Step (2.2) Block-based speech analysis:
Divide the speech sentence into blocks, each is of 20 ms duration. For each block, estimate the LPC parameters by using the data samples within the block and then save the parameters into a 2D matrix $a(j = 1, \cdots, p;\ i = 1, \cdots, \text{TotalBlocks}\}$, where $j = 1, \cdots, p$, p is the model order, and i is the block number.

(**hint**: apply the LPC analysis in Step (1.2) independently to each block without considering its neighboring blocks. Further, the data samples in the i-th block is related to the speech signal $s((i-1) * L + 1 : i * L))$, where L is the number of samples in each block.)

Results to be included in the report:
2.2.a. the number of samples in each block =
2.2.b. the total number of blocks: TotalBlocks =
2.2.c. the sampling rate f_s=

Step (2.3) Block-based estimation of residual sequence $\hat{e}(n)$:
For each block, compute the residual sequence $\hat{e}_i(m)$ using Eq.(4). Use a similar way as in Step(1.3), *except* for a few samples around the left boundary points in each block: when computing the first p residual samples $\hat{e}_i(m)$, $m = 1, 2, \cdots, p$ in the i-th block, one needs to use the signal samples $s_{i-1}(n)$ from the previous block (instead of setting them to zeros as in Step (1.3)). For the offline processing, a simple way to automatically include this continuity condition in the block-based processing is:

1. Initialize the residual sequence as a zero sequence $\hat{e} = zeros(length(s), 1)$;
2. compute each residual value $\hat{e}(n)$ within each block: e.g., for the i-th block, the start and the end indices of $\hat{e}(n)$ are $n = (i-1) * L + 1 : i * L$. Also, when compute the residuals in the i-th block one should use the estimated parameters $\hat{a}_j^{(i)}(1, \cdots, p\ )$ from the corresponding block.
3. Repeat the above process over all blocks, i.e. $i = 1, 2, \cdots$, TotalBlocks.

Observe the residual sequence by zooming in some segments from the residual sequence: is there any periodicity (pitch period or fundamental period) shown in the residual signal? Count the approximate number of samples within one cycle. Can you think a way to automatically estimate this period ?

Results to be included in the report:
2.3.a. Plot the entire residual sequence $\hat{e}(n)$.
2.3.b. Plot two blocks from the residual sequence and from the corresponding speech $s(n)$ (chose a voiced part!).
2.3.c. Compare the 2 plots in 2.3.b: which one (original speech signal, or residuals) is easier to observe the pitch period ?
2.3.d. Count the approximate number of samples in one pitch period (manually counted) =

Step (2.4) Block-based speech re-synthesis:
For each block, compute the synthesis speech signal $\hat{s}(m)$, where the range of index m for the i-th block is: $m = (i-1) * L + 1 : i * L$. This can be done in a similar way as that in Step (1.4) (using Eq.(5) *except* several synthetic samples around the left boundary in each block. For the first p samples in each block, computing the synthetic signal requires the use of some samples from the

previous block in order to ensure the continuity of the synthetic speech.

Similarly, for the off-line processing, we can use a simple implementation approach that automatically takes care of the above block-related issue:

1. Initialize the entire synthesis speech sequence as $\hat{s} = zeros(1 : length(s), 1)$;
2. Compute each synthesized speech value $\hat{s}(n)$ within a block: e.g. for the i-th block, the start and end indices of $\hat{s}(n)$ are $n = (i-1) * L + 1 : i * L$. Also, using the estimated parameters $\hat{a}_i(1, \cdots, p)$ in the i-th block for synthesizing the speech in that block;
3. Repeat the above step over all blocks, i.e. $i = 1, 2, \cdots$, TotalBlocks.

Results to be included in the report:
2.4.a. Plot the original speech $s(n)$, the re-synthesized speech $\hat{s}(n)$, and the residual sequence $\hat{e}(n)$ in one figure (use the Matlab function: subplot(3,1,1), etc.), and write your comments after comparing the original and the re-synthesized speech.
2.4.b. Listen to the synthetic speech and the original speech (use the Matlab function: wavplay), and compare them. Write your comments after the comparison.
2.4.c. Re-synthesized speech (sentence) in a *.wav file.

Task-3: Re-synthesize Speech by Using K Most Significant Residuals/Block as the Excitations

In this task, you will re-synthesize the speech signal again, however, you shall use a modified residual sequence $\tilde{e}(n)$ when re-synthesizing the speech.

Step (3.1) Using the residual sequence generated from Step (2.3), the modified residual sequence $\tilde{e}(n)$ is formed as follows:

- For each block, pick up the K (e.g. K=16) most significant residual values $e_j(n)$ (note: this shall include both the positive and negative values) and keep these values. Noting that, for simplicity, we choose to pick up K significant values, however, a more decent way is to use a codebook where the residuals in each block shall correspond to a codeword). Set the remaining residual values $e_j(n)$ in the block to zero.
- Repeat this process for all blocks to the residual sequence $\hat{e}(n)$ obtained from Step (2.3).

Step (3.2) Using the modified residual sequence $\tilde{e}(n)$ as the input, and the estimated LPC parameters from Step (2.2), re-synthesize the speech signal by repeating the Step (2.3).

Step (3.3) Change K values: Change K value to K=32, repeat the above steps, and listen to the synthetic speech.

Results to be included in the report:
3.a. Plot the original speech $s(n)$, the re-synthesized speech $\tilde{s}(n)$, and the modified residual sequence $\tilde{e}(n)$ in one figure (use the Matlab function: subplot).
3.b. Listen to the re-synthesized speech $\tilde{s}(n)$ and the original speech $s(n)$ (use the Matlab function: "audioplay"): write your comments after comparing these two speech signals.
3.c. Re-synthesized speech (sentence) in a *.wav file.
3.d. Write down your comment: what are the difference between the synthetic speech signals when using different K values?

Task-4: Using Cepstrum to Estimate Pitch Periods in Voiced Speech

In this task, you will try to estimate the pitch periods for voiced blocks from the given speech signal file $s(n)$.
Step (4.1): Choose the 'cepstrum-based method' for pitch estimation, you may follow the guide below. Further, it is worth noting that a few Matlab functions on cepstrum exist however they are not defined in the same way as in Eq.(8). So the best choice is probably to make your own Matlab codes. You shall compute block-based cepstrum for each speech block, i.e., for the i-th block, set $\mathbf{x}_i = s((i-1)*L+1 : i*L)$ and applying Eq.(8).

Some practical and essential guide for implementing the cepstrum include:

1. First, multiply the block of speech with a window function (e.g. a Hamming window, i.e., $\mathbf{x}_i = \mathbf{x}_i .* hamming(length(\mathbf{x}_i))$, to reduce the Gibbs effect due to cutting speech into blocks.
2. Then, pad zeros: the number of zeros shall be at least equal to the number of the data samples: this makes the subsequent FFT transform equivalent to a linear convolution in the time domain! ($\mathbf{y} = [\mathbf{x}_i \ zeros(length(\mathbf{x}_i), 1)]$;
3. You may decide to pack much more zeros than the original data length (e.g. $\mathbf{y} = [\mathbf{x}_i \ \ zeros(10 * length(\mathbf{x}_i), 1)]$). This can increase the number of spectral points, which makes it easier to visualize the peaks in the cepstrum domain.

Exercise:
Step (4.1.1) For each block, e.g. the i-th block of speech signal $\mathbf{x}_i = s((i-1)*L+1 : i*L)$, multiply the speech with a hamming window. (use the Matlab window function: hamming).

Step (4.1.2) Pad zeros to the data obtained from Step (4.1.1). (see the above "practical guide" for details)

Step (4.1.3) Apply Eq.(8) to the data obtained from the Step (4.1.2) to obtain the cepstrum for each block of speech. (do not forget to apply $abs(\dots)$ in Matlab).

Step (4.1.4) Repeat the above 3 steps (Steps(4.1.1)-(4.1.3)) for all blocks.

Step (4.1.5) Plot the block-based cepstra in a way similar to Fig.6.
(hint: an example of the Matlab codes is:
figure, hold;
for i=1:TotalBlocks
c2=ceptrum + i*c; plot(c2); end
Note: 'c' is a constant, e.g. c=0.5, such that the plots from different blocks are visually separable (or, shifted up by 'c' value)).

Results to be included in the report:
4.1.a. Plot the resulting cepstra for 20 blocks
4.1.b. Manually mark the pitch periods (for 20 blocks) on the above plot.

Task 5: Objective Measures for Synthetic Speech Quality

To evaluate the quality of synthetic speech, the best way is to use human auditory system. However, objective measures are often used to give quantitative description of the synthetic speech quality. Since our ears are insensitive to phase changes, waveform comparisons between the original speech and the synthetic speech do not really work well. In this Lab. exercise, you will use the LPC spectral

distance metric as the quality measure. From the signal processing theory, the LPC magnitude spectrum of speech signal for i-th block is

$$A^{(i)}(\omega) = \frac{b_0}{|\sum_{l=0}^{p} a_l^{(i)} e^{-j\omega l}|} \tag{9}$$

where $a_l^{(i)}$ is the LPC coefficients in the i-th block, $a_l^{(0)} = 1.0$, $l = 0, 1, \cdots, p$, and $b_0 = 1.0$. Noting that the gain $G_i = \sigma_e^{(i)}$ (see description in page 2) is removed from (9) since the gains of the original and synthetic speech could be different and would not affect the speech quality.

Step 5.1: For all blocks, $i = 1, \cdots$,TotalBlocks, compute the magnitude spectrum $A_{\tilde{s}}^{(i)}(\omega)$ from the synthetic speech $\tilde{s}(n)$ obtained in Task 3, and compute the spectrum $A_s^{(i)}(\omega)$ from the original speech $s(n)$ in the corresponding block. Compute the power spectra, $P_{\tilde{s}}^{(i)}(\omega) = |A_{\tilde{s}}^{(i)}(\omega)|^2$ and $P_s^{(i)}(\omega) = |A_s^{(i)}(\omega)|^2$. Plot: the two power spectra $P_s^{(i)}(\omega_m)$ $P_{\tilde{s}}^{(i)}(\omega_m)$ as a function of the discrete frequency (in radians) ω_m (i.e., the original and the synthetic speech) from one block (corresponding to the same voiced speech) in one figure, and compare the difference. Noting that $\omega_m = 2\pi f_m / f_s$ is the discrete frequency (in radians), f_m is obtained by equal-space sampled frequency $\in [0, B]$, $B = f_s/2$ is the bandwidth of speech signal, f_s is the sampling frequency, $|A_s^{(i)}(\omega_0)|^2 = |A_{\tilde{s}}^{(i)}|(\omega_0)|^2 = 1.0$,

Step 5.2: Compute the average distortion $d(i)$ by summed LPC spectral distance within each block i, as follows:

$$d(i) = \frac{1}{M} \sum_m 10\ log_{10} \left| A_s^{(i)}(\omega_m) - A_{\tilde{s}}^{(i)}(\omega_m) \right|^2 \qquad (dB)$$

where M is total number of discrete frequencies used in the summation of m, and $i = 1, \cdots,$ TotalBlocks is the index of block for the time-domain speech signals.
Plot the distortion $d(i)$ as a function of block number i.

Step 5.3: Based on the distortions you have obtained, make comments on the quality of your current speech synthesizer, also discuss which parameters could further affect your synthesizer's quality, and in which direction (better or worse).
(For example, observing and comparing the results by changing the LPC model order p in your synthesizer, or changing the number of residuals used in the synthesizer, followed by repeating the speech analysis, synthesis, and then measuring the distortions again).

Task 6: Writing the Report

Report should include all plots and required values according to the tasks in each step. **Your Matlab programs should also be included**. Further, the report shall include your summary, comments and insight to the subject, from observing the results from this laboratory exercise. In addition, the following discussions should be included in your report:

1. How speech signal compression is achieved in this Lab. work? (brief, in less than 2 lines of text!)
2. Suppose each speech signal sample consists of 8 bits, and the sample rate is f_s=12 kHz. How many bits are required to encode one block (20ms) of *uncompressed* speech signals?
3. Using the (LPC) speech model in Fig.2 where the all-pole model is specified by Eq.(1), list the *minimum* number of parameters that are required to encode one block (20ms) of speech.
4. Assume each parameter in the above step 3 is encoded (in average) by 8 bits. How many bits do you require to encode one block of compressed speech under such a model?
5. What is the compression ratio between the uncompressed speech (step 2) and the LPC compressed speech (steps 3 and 4) ?
6. There are many possible ways to compress audio/speech signals ? Write down *one name* of those speech compression methods that are based on non-parametric methods.