SCHM Final Report: An Implementation of CELP-based Vocoder

Cupjin Huang
Institute for Interdisciplinary
Information Sciences
Tsinghua University
Beijing, China
Email: pertox4726@gmail.com

Jiawen Liang
Institute for Interdisciplinary
Information Sciences
Tsinghua University
Beijing, China
Email: taobingxue001@126.com

***Abstract*—In this report we present a linear prediction-based speech compression mechanism. In particular, we adopt codebook-excited linear prediction(CELP) method, which significantly lowered the bit rate as well as maintaining the signal-to-noise ratio(SNR).**

I. Introduction

Speech coding has been a hot topic during the past few decades, due to limited bandwidth resource and increasing need to communicate distantly. In fact, among all kinds of sounds, speech signals have been intensively studied, and many effective and efficient models of speech coding are invented.
Two mainstream categories of speech coders are waveform coders and vocoders. Waveform coders focus on the relationship between time domain and frequancy domain, and they compress signal mainly by throwing away some information on either domain. Vocoders are more specifized on sppech signals. Theys are based on the theory of human sound production, and many of them intend to simulate human sound by applying the source-filter model of human voice production. Examples of vocoders include formant coders and linear predictive coders.
Linear predictive coders (a.k.a. LP coders) are based on the linear-prediction theory, that is, future values of a discrete time signal are estimated as a linear function of previous samples. Due to the quasi-consistency of human sound, linear prediction can be applied to estimate the values of speech signal during a short time window, where the signal can be approximated by a stochastic process.
Linear predictive coders are widely used nowadays. In fact, LPCs are adopted in two well-known standards of telephony speech coding, namely FS-1015 and FS-1016. In particular, FS-1016 is designed upon a variation of LPC, namely codebook excited linear prediction (CELP), which significantly increased the sound quality while not increasing the bit rate too much.
Upon this course, we are required to design a speech codec based on linear predictive coding. After a few tries, we decided to simulate the FS-1016 standard, including framewide STP (short-time prediction including linear predictive coefficient estimation, linear spectral pair estimation) , subframewide LTP (long-time prediction including pitch estimation and innovation signal estimation), and encoding using codebooks. Below in this section, we introduce some basic ideas in linear predictive coding, involving the theory of linear prediction, the source-filter model and the CELP mechanism. In the next section, we will present things related to our project, including the intuitions came into our minds, problems encountered, analysis to the result and possible future extensions. In the conclusion part, we will give a concise conclusion of our project.

A. Introduction to Basic LP Theory

A discrete-time signal can be represented by a sequence of numbers, denoted by $s[n], n \in \mathbf{Z}$. Given the signal having some stochastic characteristics (for example, periodity), we can estimate the future values of the signal as a linear function of previous samples, i.e.

$$\hat{s}[n] = \sum_{i=1}^{K} a_i s[n-i].$$

We want our estimation to be the most accurate, so we want the error (sometimes also called excitation)

$$e[n] = s[n] - \hat{s}[n]$$

to attain minimum in some sense. Here, we use the least square estimation, that is, we want to minimize

$$\sum_i e[i]^2 = \sum_i (s[i] - \hat{s}[i])^2.$$

Once the error introduced is minimized, we can recover the original signal using only the transmitted linear prediction signals $\{a_i\}_{i=1}^{K}$.
For LP vocoders, it is known that voice signals are quasi-consistent during a short time period, which is usually 20-30ms. Therefore, it is possible to first divide the signals into windowed segments, then encode them seperately by LP coefficients. More specifically, denote the window length $N = t \cdot f$, where t denotes the time of the window and f denotes the sample rate, then what we want to minimize becomes

$$\sum_{i=1}^{N} e[i]^2 = \sum_{i=1}^{N} \left(s[i] - \sum_{j=1}^{K} a_j s[i-j] \right)^2.$$

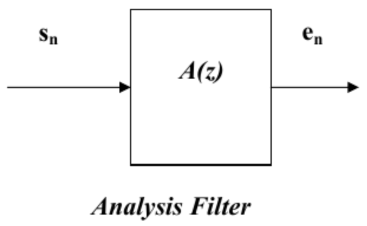

Fig. 1. Analysis Filter Transforms $s[n]$ to $e[n]$.

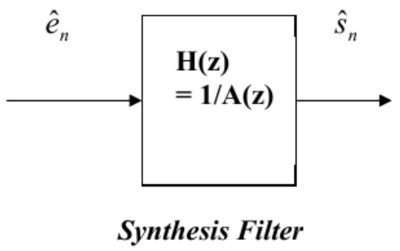

Fig. 2. Synthesis Filter Transforms $\hat{e}[n]$ to $\hat{s}[n]$.

Signals outside the range $[N]$ are regarded as zero.
This is a typical problem in quadratic programming, which has been studied quite in depth. In fact, there already exists an algorithm, called Levinson-Durbin algorithm, which can solve this problem much more efficiently than the usual QP problems by making use of the Toeplitz matrix appeared in the middle of the problem. We adopted it in our design of the speech coding mechanism. This is not the main point of the report, so I'll not go into details about state-of-art algorithms.
The linear predictive coefficients, also called LPCs, can be seen as a filter for signals, i.e. the filter

$$A(z) = 1 - \sum_{i=1}^{K} a_i z^{-i}$$

can transform the original signal $s[n]$ to the excitation signal $e[n]$, and the inverse filter,

$$H(z) = \frac{1}{1 - \sum_{i=1}^{K} a_i z^{-i}} = \frac{1}{A(z)}$$

can transform the excitation signal $\hat{e}[n]$ back to $\hat{s}[n]$, as shown above. Therefore, a typical LP coding mechanism may have the following steps:

- In the encoding phase, we analyze the speech signals to determine the coefficients of the analysis filter $A(z)$;

- In the decoding phase, we use these transmitted coefficients to construct the synthesis filter $H(z)$, then by inputting excitation signal we can possibly reconstruct the speech signal $\hat{s}[n]$.

To reconstruct the voice, we need not only the LPCs to construct the filter $H(z)$, but also the input impulses $\hat{e}[n]$. It is of no meaning to directly transmit the original excitations, as there are as much as information in excitations and in original signals. By revealing some properties of human voice production, there is a way to approximate $e[n]$ more reasonably.

B. *Introduction to Source-Filter Model*

In source-filter model of human voice production, speech is regarded as a combination of a sound source, such as the vocal cords, and a linear acoustic filter, the vocal tract. It is often assumed that the source and the filter is independent, and in our case, the synthesis filter $H(z)$ can be regarded as the filter, while the excitation signal resembles the source.
Moreover, different phonemes can be approximately classified into two categories: the voiced sound is produced when the source signal generates almost periodic impulses, and the unvoiced sound is produced when the source signal is close to white noise. Therefore, by by examining the periodity of the excitation signal $e[n]$, one can tell whether the phoneme is voiced. This information can help to partly reconstruct the source signal in synthesis phase, i.e. $\hat{e}[n]$.

C. *Codebook Excited LP Mechanism*

However, the sound produced using solely the LPCs and a number indicating the pitch sounds synthetic and unnatural. This is because that some sounds cannot be clearly classified as voiced or unvoiced; there are some voices between voiced and unvoiced. Apart from this, sudden changes of filters between two frames may cause feelings of discontinuity. Therefore, the so-called long-time predictive analysis is introduced to reveal some periodic relationship between frames. More specifically, we can view the excitation signal with a periodic component, i.e.

$$e[n] = \beta e[n - T] + c[n],$$

where β is called the pitch gain and T is called the pitch delay. These two numbers reveal long-time stochastic properties of speech signals such as the pitch of the voice. As for the remaining innovation signal $c[n]$, we can introduce a codebook C to deal with it. Using some classification scheme such as K-means, one is able to find the clustering properties of innovation signals, thus those occurs more frequently can be compressed using less bits, while the quality of speech will not decrease much stochastically.

The intuition of codebook-driven encoding can be applied further in other part of the coding mechanism. When we encode the LPCs, it will save much more bits if we try to transmit a quantized version rather than the original one, and for quantized version, we can use a codebook to transmit only the indices on the codebook. The same idea also goes with the encoding for pitch delays and pitch gains. Moreover, for innovation signals we also have a gain factor indicating the amplitude, called codebook gain, which can also be quantized and codebook-driven. As all parts of the coding mechanism is codebook-driven, the overall bit rate can be reduced to 4.8kbps in FS-1016 standard.

II. Implementation And Analysis

For the implementation, we first tried a prototype version including solely LP schemes, then we tried to construct a complete version of CELP mechanisms. For both implementations, we will first dig inside the implementation, and then do some analysis on the outcomes. Both implementations are written in c++.

A. First Try: Barebone LP Coding Mechanism

1) Main Implementation Scheme: In our first implementation, we applied barebone linear prediction mechanism. That is, for a 20ms-long window (i.e. $N = 160$ samples):

- We first applied Levinson-Durbin algorithm to compute a 10-order analysis filter $A(z) = 1 - \sum_{i=1}^{10} a_i z^{-i}$.
- To encode the speech signal, we simply transmitted the LPCs ($\{a_i\}_{i=1}^{10}$ in `double`) as well as the head of the original signal, i.e. $s[1] - s[10]$ in `short`.
- To get excitation, we computed the cross-correlation coefficient of the excitation signal $e[n] = s[n] - \hat{s}[n]$, and chose a local optimum as the speech pitch delay T. Pitch gain β is then computed using energy of the exitation signal.
- To reconstruct the signal, we constructed the synthesis filter $H(z) = \frac{1}{A(z)}$ using the LPCs and generated periodic excitation signals using white noise and pitch synthesis filter $\frac{1}{1-\beta z^{-T}}$, and then put it into $H(z)$ to reconstruct $\hat{s}[n]$.

2) Tests Taken: We were very glad to see that the program works, as after hours of debugging there eventually turns out to be meaningful sound coming out. We tested it on some voice corpus provided in the HTK pack available in Xuetang Website, including `speechrecognition1.wav` and `zhongguohaoshengyin.wav`. Some of the waveforms are listed below:

Here, the red line stands for $s[n]$ and the blue one $\hat{s}[n]$. We can see from Fig. 3 that the barebone LP mechanism can by some extent catch the characteristics of the speech signal, but the prediction performs very poor when the time becomes longer and longer.
Perceptually, the sound is recognizable but very noisy. The main cause of noise seems to be the original white noise: the phonemes are more or less unvoiced after all, indicating that the pitch synthesizer doesn't work so well.

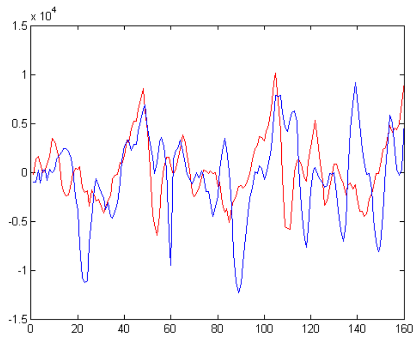

Fig. 3. Comparison between original and synthesized speech during a short window, corpus `zhongguohaoshengyin.wav`

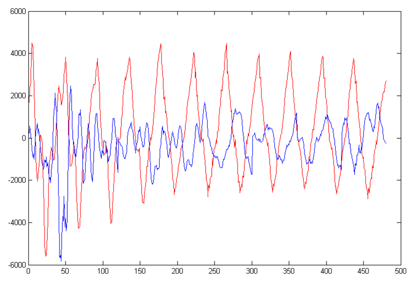

Fig. 4. Comparison between original and synthesized speech during a relatively long window, corpus `zhongguohaoshengyin.wav`

3) Requirements Met: Unfortunately, no requirements are met except the window length.

- Bit Rate: Bits taken in each 20ms-long frame takes can be calculated as follows:
  - $a_1 \sim a_{10}$: 10 `doubles` $\times$ 4 bytes per `double`$=$ 320 bits;
  - $s[1] \sim s[10]$: 10 `shorts` $\times$ 2 bytes per `short`$=$ 160 bits;
  - T : 1 `short`$=$ 16 bits;
  - β : 1 `double` $=$ 32 bits.

  The total bit rate is 26kbps, much more than afforded. The main reason of this is we didn't come up with the idea of quantization. However, even with quantization, it will still be difficult to insert both the LPCs and the header of the signal into a limited sized frame.
- SNR: Signal-to-noise ratio (SNR) is an important criterion of the quality of lossy compression, indicating the proportion of noise compared with meaningful information, i.e. the original signal. As it has never been formally defined how to calculate SNR, we applied the method Prof. Ching said at the final presentation.
  Given two sequences of signals $s[n]$ and $\hat{s}[n]$, we want to estimate the excitation without disturbance by phase differences, as humans ears are not sensitive to phase

differences. To put it more specifically, define

$$err_{\min} = \min_{i=0}^{\alpha N} \frac{N}{N-i} \sum_{j=0}^{N-i} (s[j] - \hat{s}[i+j])^2.$$

Here $\frac{N}{N-i}$ is the normalization factor, eliminating the factor introduced while the two segments overlaps less and less, and α is a constant to guarantee the lower bound of overlapped samples. In our testing scheme, α is set to 0.5. We use $err_{\min}$ as the noise energy and

$$E_{sig} = \sum_{i=0}^{N-1} s[i]^2$$

as the signal energy. SNR can then be calculated once we have both noise and signal measure. That is,

$$\sigma = 10 \log_{10} \frac{E_{sig}}{E_{noise}} = 10 \log 10 \frac{E_{sig}}{err_{\min}}.$$

Taking the average over all frames and we can get the result.
In our case, the SNR measurement using 160ms-long window simply gets a result about -0.012dB, i.e. there are as much noise as the signal information. Considering that we did subtrtaction first then squaring, the actual situation may be worse.

To conclude, basic problems arised in our first try include:

- Bit Rate: We spent too many bits to transmit float-point (or even double) numbers. This is the main cause of bit rate exceeding. Although I thought of using a quantized version, but of stability concern I chose to use the original FP number at last. This was significantly improved in out later implementation version.

- Noise: It was not clear whether the pitch gain β is calculated correctly, but it seemed not to be working very well. With input impulses nearly white noise, our synthesized signal sounded harsh and unnatural, and the SNR requirement wasn't met.

B. *Second Try: FS-1016 Simulation*

In our second try, we found an implementation of FS-1016 encoding scheme on the Internet, so we decided to follow that to establish a codec with a much better quality. This one is far more mature than the previous one. New extensions include:

- High pass filter is used to eliminate the low frequency noise in the original speech;

- Transmitting line spectral pairs (LSPs) rather than LPCs. LSPs are far less sensitive to quantization than LPCs, therefore quantizing LSPs can be a better idea than quantizing the original LPCs.

- Perceptual weighting is used to improve the perceptual quality of synthesized speech.

- Subframe-level reconstruction is used to eliminate the discontinuities between consecutive frames. Interpolation of LSPs in subframe level is used to provide a more smoothly changing synthesis filter.

- More state-of-art method of pitch analysis and pith synthesis is used. Both pitch delay and pitch gain is searched using an adaptive codebook, where the entries are concurrently calculated with a couple of previous frames. This makes it more accurate to estimate the pitch, and also makes it possible to deal with pitches with a long cycle.

- Innovation signal is quantized using an existing codebook.

This scheme is kind of state-of-art, and the codebooks provided are relatively fixed. For this reason, our window length is constrained to be 30ms (i.e. 240 samples in total).
We tried to fully understand the main coding scheme, but there are still things left nott understood so well, due to the limited time. Although we have some doubts concerning the coding scheme, I will explain the fully understood steps and try my best to figure out the meaning of the steps I cannot understand so well.
In the presentation, I divided the whole coding scheme into two main phases: one deals with the entire 30ms-long frame, while the other analyze each subframe. Here I'd prefer call them Phase A and Phase B.

1) Phase A: Phase A deals with the entire frame. In Phase A, the frame signal is first sent to a high pass filter (HPF) to eliminate the low-frequency noise, then windowed using a Hamming window to guarantee continuity when calculating LPCs. Then, LPCs and LSPs of the entrie frame is calculated. Finally we quantize the LSPs and send them to the next phase, where LSPs of each subframe is used to reconstruct a perceptually weighted synthesis filter.

- High Pass Filter: We used a filter of the form

  $$H(z) = H_{zero}(z) H_{pole}(z)$$

  where $H_{zero}(z) = 0.946 - 1.892z^{-1} + 0.946z^{-2}$ is a zero filter and $H_{pole}(z) = \frac{1}{1-1.889z^{-1}+0.895z^{-2}}$ is a pole filter. I'm not sure how it comes, but it does well to prevent the undesired low-frequency components. This pre-processed signal is used as our original signal, namely $s[n]$.

- Hamming window: Hamming window is a pre-processing step before calculating the LPCs. As the length of the window is fixed, the Hamming window is presented not as a function but as a codebook. Here we use

  $$\bar{s}[n] = s[n] \cdot ham[n]$$

  to calculate the windowed speech. The hamming window and a segment of windowed speech is presented below.

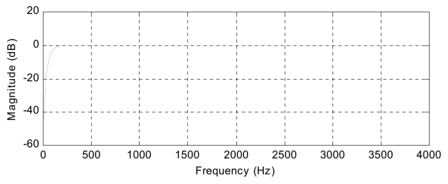

Fig. 5. Performance of the high pass filter $H(z)$ given varying frequancies. It can be seen that the low-frequency component is surely blocked out.

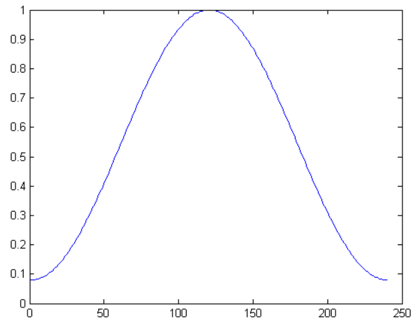

Fig. 6. Hamming window $ham[240]$ plotted with Matlab.

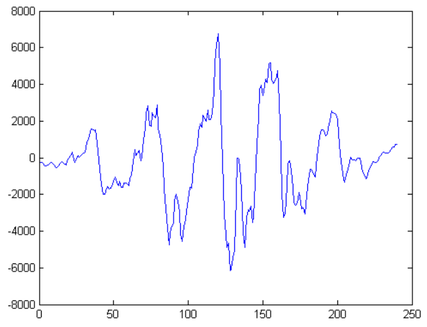

Fig. 7. Windowed speech segment using the Hamming window, corpus `zhongguohaoshengyin.wav`

- LPC calculating: LPC is calculated the same way we did in the first try, i.e. using Levinson-Durbin's algorithm to solve the equation

$$a = \Phi^{-1}\Phi_0,$$

where the cross-correlation matrix Φ is Toeplitz.

- LSP calculating: The LP polynomial $A(z) = 1 - \sum_{i=1}^{K} a_i z^{-i}$ can be decomposed into:

$$P(z) = A(z) + z^{-(K+1)}A(z^{-1}),$$

$$Q(z) = A(z) - z^{-(K+1)}A(z^{-1}).$$

Now we know that

  - All the roots of $P(z)$ and $Q(z)$ lie on the unit circle
  - The roots of $P(z)$ and $Q(z)$ are interspersed
  - $P(z)$ corresponeds to the vocal tract with the glottis closed and $Q(z)$ with the glottis open (though I don't know why),

We can just use the roots of $P(z)$ and $Q(z)$ to represent the LPCs. All the roots of $P(z)$ and $Q(z)$ lie on the unit cycle, so it would not be possible to directly solve the roots using bisection. Instead, note that

$$P(\omega) = 0 \Leftrightarrow P(\bar{\omega}) = 0,$$

We can concentrate on the upper half of the unit cycle, and use bisection to solve the equations

$$P(e^{-i\pi\theta}) = 0, \theta \in [0,1]; Q(e^{-i\pi\theta}) = 0, \theta \in [0,1].$$

The interspersity of the root ensures us to find all the 10 roots. We call them line spectral pairs (LSPs).

- We know that the LSPs reveal some properties of human vocal tract, therefore it is possible to quantize them based on statistical results on human vocal tract. In fact, in quantization step we used a codebook lspTbl again to find the indices with entries the closest to the solved LSPs. These indices are the first part of the frame pack, which takes 10 `char` $\times$ 8 bit per `char` = 80 bits.

2) Phase B: After solving the quantized LSPs, we further explore the short-time characteristics of the frame. Here we divide the frame into 4×7.5ms subframes, each with 60 samples. We will first construct a perceptual weighted synthesis filter, and then use this filter to search for excitation signals and pitch delays in a closed-loop manner.
This phase use a kind of search method called Analysis-by-Synthesis (AbS) search. We use the perceptually weighted input signal as the reference, and we try to minimize the perceptually weighted error by choosing the best possible parameters. Pitch delay T and pitch gain β are calculated and quantized first, from when we can use the pitch synthesis filter $P(z) = \frac{1}{1-\beta z^{-T}}$ as granted. The excitation signal is then chosen from a overlapping codebook using the mean square error criterion. Finally we pack the parameters (pitch delay, pitch gain, codebook excitation signal, codebook excitation gain) together for one subframe. After all four subframes are finished, the pack is sent to the bitstream or stored into a file.

- Interpolate LSPs: To make the filter changing more smoothly, we construct the subframe synthesis filter using LSPs which are interpolated from the two consecutive frames. One thing needs to be pointed out is that when we do subframe-level analysis, the subframes we handle are not the four subframes within the current frame, but the last two of the previous frame plus the first two of the current frame. This causes a 15ms delay, which is tolerable provided that the computation process is efficient.

- Reconstruct LPCs: With these LSPs, we can reconstruct the trigonometric polynomials

$$P(e^{-i\pi\theta}), Q(e^{-i\pi\theta}).$$

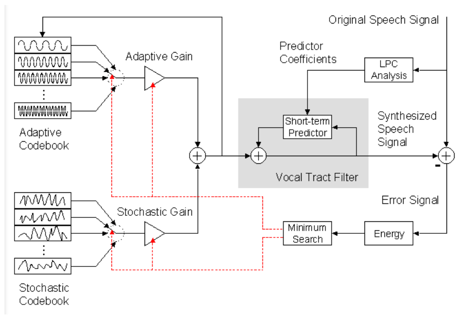

Fig. 8. The Closed-Loop LTP included in Analysis-by-Synthesis

From these coefficients in the two trigonometric polynomials, we can reconstruct the LPCs $\{a_i\}_{i=1}^{10}$. These LPCs will be used to construct the perceptually weighted synthesis filters later on.

- Perceptual weighting: CELP codecs minimizes the mean square of the noise in the perceptually weighted domain. This means that a perceptual noise weighting filter $W(z)$ is applied to the error signal in the encoder. In our program, $W(z)$ is a pole-zero weighting filter derived from the LPCs using bandwidth expandion. More specifically, we have

$$W(z) = \frac{A(z)}{A(z/\gamma)}, \gamma = 0.8.$$

For efficiency concerns, the filtering step is seperated in the referencial signal and the synthesized signal.

- AbS search for pitch parameters: From here I'm a bit confused with some mechanisms.
Fig. 8 shows a typical closed-loop mechanism of AbS. The stochastic codebook is used to generate innovation signals, and the adaptive codebook, which consists of previous sample values, is used to generate periodity in a long-time sense. By calculating the cross-correlation coefficents, we can find the periodity of the error signal $e[n] = s[n] - \bar{s}_{lpc}[n]$. Then by trying all possible choices of the pitch delay and pitch gain, we can obtain a long-time predictive (LTP) filter $P(z)$ for our further search of innovation signal.

- Codebook Search: Now that the LTP pitch synthesis filter $P(z) = \frac{1}{1-\beta z^{-T}}$, the perceptually weighted STP synthesis $H(z) = \frac{1}{A(z/\gamma)}$ are already determined, we can now look up the codebook signal to find the entry which best fits the innovation signal. Similar to the LTP analysis, this step is done also by identifying the peak in the cross-correlation of the codebook entries and the innovation signal. Codebook gain is then calculated using the ratio between the innovation signal energy and the codebook entry signal energy.
One thing worth mentioning is the stochastic codebook adopted by FS-1016 standard. It is an overlapping codebook, and each entry there is either 0, 1 or -1. More specifically, the stochastic codebook can be regarded as an array with 1083 entries. If we peak the entry 1, the innovation signal we will get is $x[0] \sim x[59]$, and if we peak the entry 2, we will get $x[4] \sim x[63]$ as the innovation signal (actually this is not the real case, but a choice of simplicity). Overlapping codebook reduces the amount of calculation, thus suitable for the case where vector quantization (VQ) is usually inefficient.

- The encoding scheme ends up with packing. Unlike FS-1016, we don't tend to squeeze out bits to achieve a low bit rate: as our pack storage data consists of purely indices within 8 bits (except the codebook index, which takes 9 bits), we can assume that all things are stored in `char`. Recall all things get packed:
    - For the whole frame, we packed 10 `char`s of LSP coefficients;
    - For the 4 subframes, we each packed a pitch delay index, a pitch gain index, a stochastic codebook index and a codebook gain index, which is in total $4 \times (1+1+2+1) = 20$ `char`s.
    - We preserved the sync bit in the original FS-1016 standard, but it is unused.

Therefore, a pack in total takes $10 + 20 + 1 = 31$ bytes. Then the bit rate can be calculated as

$$\frac{31 \times 8 \textbf{ bits}}{30\textbf{ms}} \approx 10.3\textbf{kbps}.$$

This bit rate falls in our requirement of $8 \sim 16$kbps.

3) Decoding Scheme: Compared to encoding, decoding is relatively easier. The decoding scheme only reads all the indices, finds the corresponding entries of LSPs, LTP parameters and innovation parameters, and then reconstruct a source-filter system to get back the synthesized signal. More specifically, for each subframe:

- The LSPs are first read, and then interpolated with the LSPs in the previous frame. These interpolated parameters are used to reconstruct the synthesis STP filter $H(z) = \frac{1}{1-A(z)}$;

- The LTP parameters are then read, and the LTP filter $P(z) = \frac{1}{1-\beta z^{-T}}$ is reconstructed.

- Finally, the stochastic codebook entry and the codebook gain is read. To look up the stochastic codebook, one can get the innovation signal entry. Put it into the filter and we can finally get the synthesized speech.

4) Tests Taken: We tested the exactly same two corpora as in the previous analysis. Moreover, to Prof. Ching's advice, we tried to test the two additional corpora (recorded by myself on my laptop):

- `An apple a day keeps doctors away.`

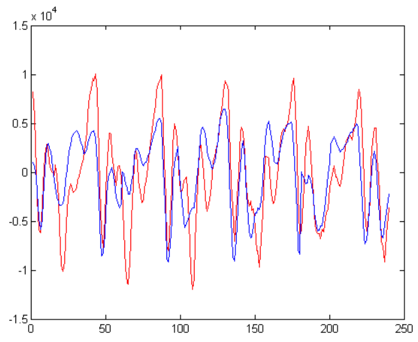

Fig. 9. Comparison between original and synthesized speech during a short window (30ms), corpus `zhongguohaoshengyin.wav`

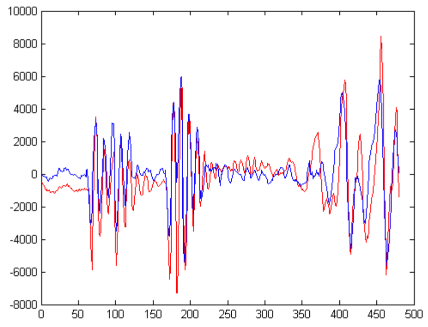

Fig. 10. Comparison between original and synthesized speech during a relatively long window (60ms), corpus `zhongguohaoshengyin.wav`

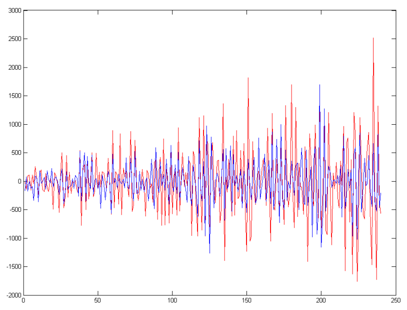

Fig. 11. Comparison between original and synthesized speech during a relatively short window (60ms), corpus `speechrecognition1.wav`. Easy to see, this phoneme has a high pitch and looks unvoiced, which has a great chance to be "-ch" in "speech".

- `The quick brown fox jumps over the lazy dog.`

However, the variety of `wav` wavform format is too difficult to convert to the specific one we need in our program (8kbps with 1 sound track), we finally gave up and focused on the previous two corpora.
The overall quality of the synthesized speech is much better than the first version of the codec. Some comparisons between the original signal and synthesized signals are shown in Fig. 9 ∼ 12.

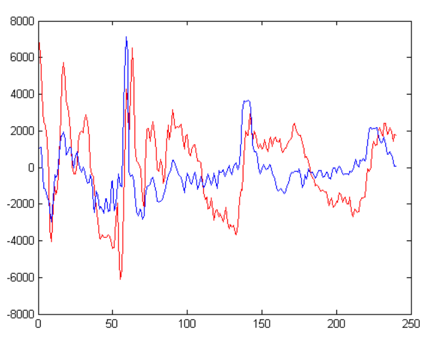

Fig. 12. Comparison between original and synthesized speech during a relatively short window (60ms), corpus `speechrecognition1.wav`. The reconstruction unexpectedly performed very poor here, maybe because of the aperiodity of the signal.

As is shown, the codec performs well even if the window length is chosen such that quasi-consistency does not hold anymore. Also note that the low-frequency noise signal is blocked out in the synthesized signal (note the difference between silences at the beginning of Fig.10).
Also, note that in Fig. 12 the reconstruction performs very poor, especially on the last half. The exact reason of this is not known yet, but I think there may be some problems in our reconstruction of the filters.

5) *Requirements Met::*

- Bit Rate: As we have already calculated, this codec can achieve a bit rate of 10.3kbps, and the bit rate constraint is now met. Furthermore, in the FS1016 coding standard, the bit rate is further lowered to 4.8kbps, as every bit allocation in the standard is carefully written. I used far more bits than actually needed, as the codebooks usually have $2^4 \sim 2^6$ entries, and I just used a `char` to represent all of them.

- SNR: Although the sound hears much more natural than the previous one, the SNR seems not to be improved much. The SNR of corpus `speechrecognition1.wav` is 2.76dB and that of `zhongguohaoshengyin.wav` is 2.99dB. Although this sounds no nice, but it is much better than the SNR of our first try. Two things remain to explain:
  - I am not sure if the way to calculate SNR actually make sense. For example, I didn't take volume (i.e. the overall energy) into account, which may cause a big disturbance to the calculated SNR; also, it seems to me that a minor difference between the original pitch and the estimated one may cause big error even in a short window, while it may not be sensible to human ears. Clearly defining "signal" away from "noise" is a difficault task, and I think there will be a more reasonable way to calculate it.

- ○ Due to limited time, I cannot be one hundred percent sure that the CELP mechanism we implemented is correct in principle. Many places, such as the codebook index determining or codebook entry extracting, may not be coded correctly. To understand the full scheme and to implement a version without noticable bug need more time than provided.

To conclude, we used much more robust mechanisms this time to avoid harshness and unstability which arised in the former implementation. These mechanisms worked well, and there is a considerable improvement in perceptual feeling. Bit rate requirement has been met, but the SNR is relatively low. This can be the problem of the calculation of SNR or be the problem of the mechanism, but it is not known yet.

III. Conclusion

In this report, we introduced the basic knowledge and ideas relating to linear prediction and linear predictive analysis, source-filter model and codebook excited linear prediction, which is used widely as standards of speech coding. Then we presented two of our implementation of CELP speech codec, where the first one is a barebone LP coding scheme, and the second one is based on an existing implementation of FS-1016 standard codec. The second one performs much better than the first one, but there are still problems need to be handled.
Altogether, it is an unforgettable experience to really dig deep inside the speech coding area. Unlike theoretical problems we meet every day in our study, engineering problems need more experiences to solve. The theories involved are not as difficult as expected, but to implement them has been a totally different problem.
Finally, we are very excited, for our programs are able to work at last!

Acknowledgment

The authors would like to thank Prof. from CUHK for having a great time studying and cooperating in the human-machine speech communication field. We've learned a lot from your teaching.
Especially, we would like to thank Prof. Ching for giving advice to the problems we encountered during the project.
Also, we would like to thank Gao Weihao from JK00, Feng Qiwei from JK10 to discuss the LP mechanism. We basically adopted different coding schemes (even coding languages), but the ideas shared are of great value.
Finally, I searched a lot on the Internet, here are some of the links I find very useful:

- http://www.speex.org/docs/manual/speex-manual/node9.html. Speex is a opensource speech coding software based on CELP, and in this page they introduced a lot about the ideas of codebook excitation linear prediction.
- http://www.speex.org/docs/manual/speex-manual/node9.html. This is a pdf which contains the implementation of FS-1016 standard speech coding. All the codebooks are from this pdf, while we wrote all functions by our own, given the indicators in this pdf.
- For histories and applications, we basically searched them in Wiki.

References

[1] T. Ogunfunmi and M. Narashimba, *Principles of Speech Coding*, CRC Press, 2010.