A SIMULATION TOOL FOR INTRODUCING ALGEBRAIC CELP (ACELP) CODING CONCEPTS IN A DSP COURSE

Venkatraman Atti and Andreas Spanias

Department of Electrical Engineering, MIDL – TRC
Arizona State University, Tempe, AZ, 85287-7206, U.S.A
E-mail: [atti, spanias]@asu.edu

ABSTRACT

This paper presents an educational tool[1] for introducing Code Excited Linear Prediction (CELP) coding concepts in senior undergraduate and graduate DSP-related courses. The tool consists of a user-friendly graphical interface along with a complete MATLAB[2] realization of all aspects of the Algebraic CELP G.729 algorithm [2]. This simulation software is accompanied by a series of computer experiments and exercises that can be used to provide hands-on training to class participants. The exercises designed based on the simulation tool may be used by instructors in a class setting to demonstrate key signal processing concepts associated with the processing of telephone-based speech. The MATLAB ACELP tool is being used in Arizona State University undergraduate DSP courses as well as in a graduate course on speech coding and in a continuing education short course. Evaluation of the tool and the exercises is being performed by an educational software assessment specialist.

1. INTRODUCTION

In the last ten years we have witnessed a series of breakthroughs in speech coding followed by several standardization efforts [1]. Most of the standardized algorithms are based on CELP coders. Although speech coding researchers and practitioners are well aware of the fundamental ideas used in CELP, students do not get much of an opportunity in courses to study these algorithms. A software simulation tool, implementing the ACELP algorithm has been developed for the purpose of introducing speech coding and the associated signal processing concepts to both undergraduate and graduate students. We choose an Algebraic Code Excited Linear Prediction (ACELP) algorithm as a basis for this educational tool because of the wide proliferation of algebraic codebooks in cellular standards. The algorithm was coded in a modular manner and in its entirety using MATLAB. The tool is based on a user-friendly graphical user interface (GUI) that allows the student to study and verify through graphics the various aspects of the algorithm such as: the LP analysis, the open-loop pitch search, the adaptive codebook search (pitch search), the fixed codebook search, and the bit allocation patterns. We choose MATLAB as the implementation platform because it allows the user to easily understand the complex parts of the algorithm whose function is not immediately evident from the standard documentation or from the C-code obtained from ITU.

2. THE MATLAB ACELP TOOL

The graphical user-interface of the MATLAB ACELP tool is shown in Figure 1. This consists of a block-by-block graphical representation of the ACELP algorithm. Each of the blocks is associated with the relevant signal processing functions that can be activated by opening the corresponding context menus.

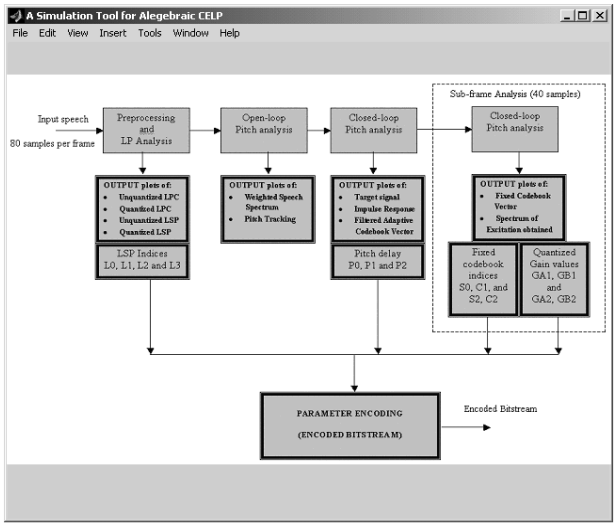

Figure 1. The GUI of the ACELP simulation tool.

The algorithm operates on 16-bit linear PCM signals at a bit rate of 8-kb/s. The speech signal is analyzed for speech frames of 10ms corresponding to 80 samples at a sampling rate of 8000 samples per second. The five important stages associated with the encoding principle of CS-ACELP include: the pre-processing stage, the LP analysis stage, the open-loop pitch search, the closed-loop pitch search, and the algebraic codebook search. These five stages appear explicitly in the GUI shown in Figure 1. A brief description of all the stages is provided in the subsequent sections.

2.1. Pre-processing and LP analysis block

The input signal is high-pass filtered and scaled in the pre-processing stage. A second order pole-zero filter with a cutoff frequency of 140Hz is used to perform the high-pass filtering. To

[1] Major portions of this work have been supported by MIDL-ASU.

[2] MATLAB is a registered trademark of The Math Works, Inc.

reduce the probability of overflows in the fixed-point implementation, the scaling operation is performed. The pre-processed signal, s(n), is windowed using a 30ms (240 samples) asymmetric window. The LP analysis window consists of half a hamming window and quarter of a cosine function cycle. The window operates on 120 samples from the past speech frame, 80 samples from the present speech frame and 40 samples from the future speech frame (a total of 240 samples). Using the Levinson-Durbin algorithm, the LP coefficients are computed from the autocorrelation coefficients corresponding to the windowed speech. The LP coefficients obtained are converted to line spectral pairs (LSP). These are later quantized and interpolated. Figure 2 shows the various steps involved in the LP analysis stage.

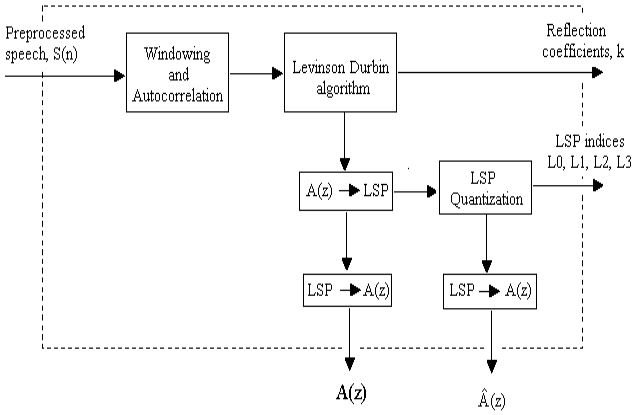

Figure 2. LP analysis block

A two-stage vector quantizer is used to quantize the LSP coefficients based on the minimization of the weighted mean squared error. This is given by

$$\text{MSE} = \sum_{i=1}^{10} w_i (\omega_i - \hat{\omega}_i)^2 \qquad (1)$$

where, w_i are the adaptive weights, and ω_i and $\hat{\omega}_i$ are the computed and predicted line spectral frequencies respectively. The codebook structure used to vector quantize the LSPs is shown in Figure 3.

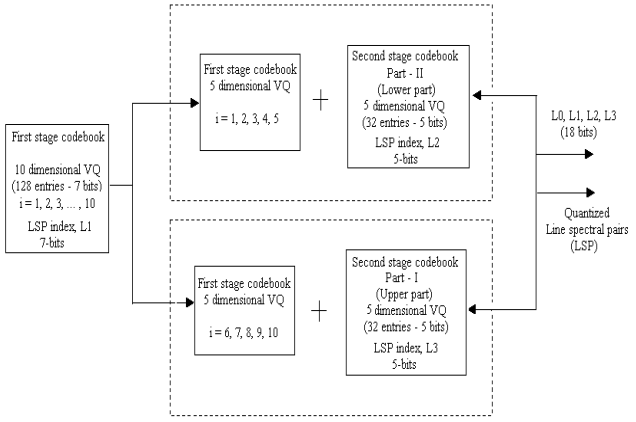

Figure 3. Codebook structure used to quantize the LSP

The first stage uses a 10-dimensional codebook *L1* with 128 entries (7 bits). The second stage is implemented as a split vector quantizer using two 5-dimensional codebooks, i.e., *L2* and *L3* containing 32 entries (5-bits) each. The codebook, *L1* is searched and the vector, L1 that minimizes the mean squared error (Eq-(1)) is selected. Similarly the vectors, L2 and L3 are selected from the codebooks, *L2* and *L3* respectively. The line spectral frequencies are obtained from the sum of two codebooks as follows:

$$\hat{l}_i = \begin{cases} L1 + L2 & i = 1, 2, \ldots, 5 \\ L1 + L3 & i = 6, 7, \ldots, 10 \end{cases} \qquad (2)$$

Two 4th order moving average (MA) predictors (L0 = 0 or 1) are used to predict the line spectral frequencies. The MA predictor that results in least mean squared error (MSE) is selected. The quantized LSP coefficients, $\hat{q}_i$, are computed using the expression, $\hat{q}_i = \cos(\hat{\omega}_i)$. The quantized and unquantized LSP coefficients are interpolated on a subframe (40samples) basis as follows:

$$\begin{aligned} &\text{Subframe-1}: \hat{q}_i = 0.5 \cdot \hat{q}_i^{(\text{previous})} + 0.5 \cdot \hat{q}_i^{(\text{current})}, && i = 1, 2, \ldots, 10 \\ &\text{Subframe-2}: \hat{q}_i = \hat{q}_i^{(\text{current})}, && i = 1, 2, \ldots, 10 \end{aligned} \qquad (3)$$

The interpolated quantized and unquantized line spectral pairs (q_i) are converted back to LP coefficients (a_i) to compute the LP synthesis (Eq-(4)) and weighting filters (Eq-(5)).

2.2. Open-loop pitch search block

The reflection coefficients, k, obtained as a by-product from Levinson-Durbin algorithm are used to compute the adaptive weight factors, γ_1 and γ_2. The unquantized LP coefficients, a_i are used to perform the perceptual weighting.

The LP synthesis filter is given by,

$$\frac{1}{\hat{A}(z)} = \frac{1}{1 + \sum_{i=1}^{10} \hat{a}_i z^{-i}} \qquad (4)$$

Where, $\hat{a}_i$, i = 1, 2, . . . 10 are the quantized LP coefficients.

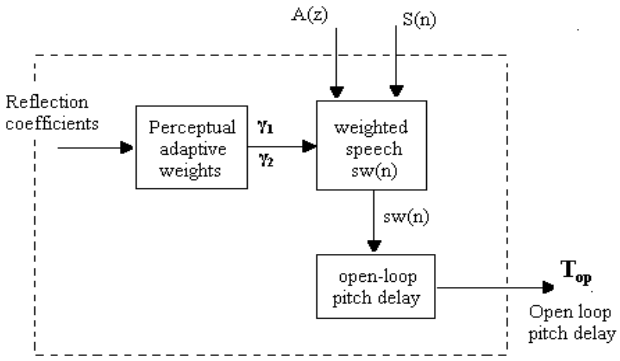

Figure 4. Open-loop pitch search block

The perceptual weighting filter, w(z) is given by,

$$w(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} = \frac{1+\sum_{i=1}^{10} \gamma_1^{\,i} a_i z^{-i}}{1+\sum_{i=1}^{10} \gamma_2^{\,i} a_i z^{-i}} \quad (5)$$

The weighted speech signal, sw(n) is computed as follows:

$$sw(n) = s(n) + \sum_{i=1}^{10} a_i \gamma_1^{\,i} s(n-i) - \sum_{i=1}^{10} a_i \gamma_2^{\,i} sw(n-i), \quad n = 0, \ldots, 39 \quad (6)$$

where, s(n) is the pre-processed speech, γ_1 and γ_2 are the adaptive weights, and a_i, i = 1, 2, . . . , 10 are the unquantized LP coefficients. The autocorrelation of the weighted speech signal, (Eq-(6)) is computed and the delay (20 to 153) at which the maximum occurs is selected as the open-loop pitch, T_{op}.

2.3. Closed-loop pitch search block

The impulse response, h(n) of the weighted synthesis filter, $W(z)/\hat{A}(z)$ is computed for each subframe. The target signal x(n) is obtained by filtering the residual signal r(n), through the combination of $1/\hat{A}(z)$ and $W(z)$ as shown in Figure 5

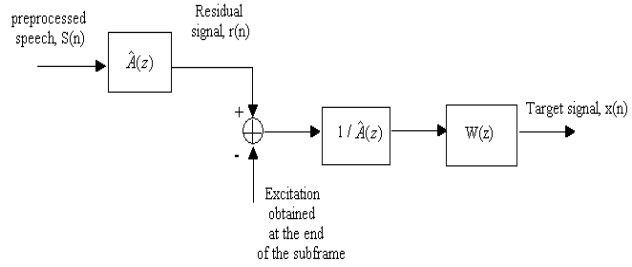

Figure 5. Computation of the target signal, x(n)

Closed-loop pitch analysis is performed on a subframe (40 samples) basis. The open-loop pitch delay, T_{op}, is used as reference for the first subframe. Pitch delay, T_1, is found by searching a small range of samples around T_{op}. The search resolution used is 1/3 in the range $\left[19\ \frac{1}{3},\ 84\ \frac{2}{3}\right]$, and increments of '1' in the range [85, 143].

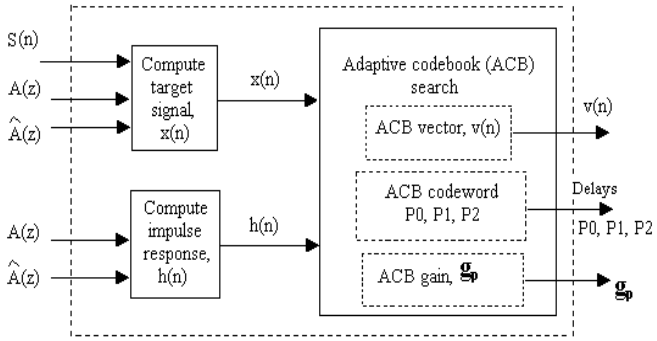

Figure 6. Closed-loop pitch (adaptive codebook) search block

For the second subframe, the optimum delay, T_2, is searched around T_1. The search resolution used is 1/3 through out the range, $\left[\text{int}(T_1) - 5\frac{2}{3},\ \text{int}(T_1) + 4\frac{2}{3}\right]$. The pitch delay, T_1 is encoded with 8 bits in the first subframe, and for the second subframe the relative delay, T_2 is encoded with 5 bits. The various steps involved in the closed-loop search (adaptive codebook search) are shown in
Figure 6.

The adaptive codebook vector, v(n), is computed by interpolating the past excitation signal. The adaptive codebook gain, g_p, is computed as follows:

$$g_p = \frac{\sum_{n=0}^{39} x(n)\, y(n)}{\sum_{n=0}^{39} y(n)\, y(n)} \quad (7)$$

$$y(n) = \sum_{i=0}^{n} v(i)\, h(n-i), \quad n = 0, \ldots, 39 \quad (8)$$

where, x(n) is the target signal, y(n) is the filtered adaptive codebook vector, v(n) is the adaptive codebook vector and h(n) is the impulse response of the weighted synthesis filter.

2.4. Algebraic (fixed) codebook search block

The name Algebraic CELP implies the structure of the codebook used to select the excitation codebook vector. The codebook vector consists of a set of interleaved permutation codes containing few nonzero elements [3, 4]. The fixed codebook structure is given by,

Table-1: Fixed codebook structure used in ITU –T G.729

Track (k)	Signs	Pulse positions (p_k)
$i_0 = 0$	s_0: ±1	p_0: 0, 5, 10, 15, 20, 25, 30, 35
$i_1 = 1$	s_1: ±1	p_1: 1, 6, 11, 16, 21, 26, 31, 36
$i_2 = 2$	s_2: ±1	p_2: 2, 7, 12, 17, 22, 27, 32, 37
$i_3 = 3$	s_3: ±1	p_3: 3, 8, 13,18, 23, 28, 33, 38 4, 9, 14, 19, 24, 29, 34, 39

where, 'p_k' is the pulse position, 'k' is the pulse number, 'L' is the interleaving depth (= 5) and 'j' ranges from 0 to $2^M - 1$, 'M' is the number of bits (=3) describing the pulse positions.

The codebook vector, c(n), is determined by placing the 4 unit pulses at the found locations (p_k) multiplied with their signs (±1) as follows:

$$c(n) = s_0\delta(n-p_0) + s_1\delta(n-p_1) + s_2\delta(n-p_2) + s_3\delta(n-p_3), \quad n = 0, \ldots, 39 \quad (9)$$

where, δ(n) is the unit impulse function. Figure 7 shows the various steps involved in computing the algebraic codebook vector, c(n). The adaptive codebook contribution is subtracted from the target signal to obtain the new target signal x'(n)as follows:

$$x'(n) = x(n) - g_p . y(n), \quad n = 0, \ldots, 39 \quad (10)$$

where, x(n) is the target signal, y(n) is the filtered adaptive codebook vector (Eq-(8)), and g_p is the adaptive codebook gain (Eq-(7)).

The correlation signal, d(n), is computed as follows:

$$d(n)=\sum_{i=n}^{39} x'(n).h(i-n), \quad n = 0, \ldots, 39 \tag{11}$$

The algebraic codebook vector, c_i at the index 'i' is searched by maximizing the term,

$$\Delta_i = \frac{\left[\sum_{n=0}^{39} d(n) \cdot c_k(n)\right]^2}{c_k^T . H^T . H . c_k} \tag{12}$$

where, d(n) is the correlation signal (Eq-(11)), and H is lower triangular Toepliz matrix with diagonal h(0) and lower diagonals h(1), h(2), . . . , h(39).

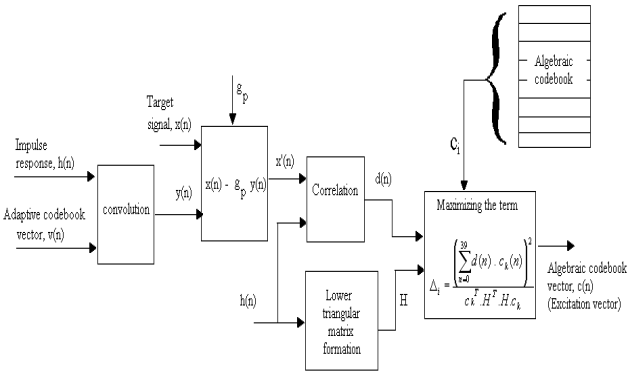

Figure 7. Algebraic codebook search block

The pulse positions and the signs of the pulses obtained are encoded as follows: the indices i_0, i_1, i_2 are encoded with 3 bits each, and the index i_3 is encoded with 4 bits. The signs of i_0, i_1, i_2 and i_3 are encoded with 4 bits. Hence a total of 17 bits are required to encode the fixed codebook parameters for each subframe.

The adaptive codebook vector, v(n), and the fixed codebook vector, c(n), are used to search the conjugate-structure codebook to quantize the adaptive codebook gain, g_p, and the fixed codebook gain, g_c.

$$g_c = 10^{(E_m + \bar{E} - E)/20} \tag{13}$$

$$\text{where, } E = 10.\log\left(\frac{1}{40}\sum_{n=0}^{39} c(n)^2\right) \tag{14}$$

$\bar{E}$ = 30dB, is the mean energy of the fixed codebook excitation. E_m = mean-removed energy that can be predicted using a 4th order moving average predictor. The predicted energy is given by, $\tilde{E}_m$.The predicted gain g_c' is given by,

$$g_c' = 10^{(\tilde{E}_m + \bar{E} - E)/20} \tag{15}$$

The actual fixed codebook gain, g_c and the predicted gain g_c' are related as follows,

$$g_c = \gamma \cdot g_c'$$

where γ is the correction factor. Instead of vector quantizing the fixed codebook gain, g_c', it is preferred to quantize the correction factor, γ. A two-stage vector quantizer is used to quantize the adaptive codebook gain and the correction factor. The first stage consists of a 3-bit two-dimensional codebook, and the second stage consists of a 4-bit two-dimensional codebook. In both the stages, the first element refers to adaptive codebook gain, g_p, and the second element refers to the correction factor, γ.

Table-2: Bit allocation of 8kbps CS-ACELP algorithm

Parameter			Parameter description	Number of bits
LSP Parameters		L0	MA predictor index of LSP quantizer	1
		L1	First stage vector of LSP quantizer	7
		L2	Second stage, lower vector of LSP quantizer	5
		L3	Second stage, higher vector of quantizer	5
SUBFRAME – I	Pitch details	P1	Pitch delay	8
		P0	Parity bit for pitch delay	1
	Algebraic Codebook	S1	Signs of fixed-codebook pulses	4
		C1	Pulse positions of fixed-codebook	13
	Conjugate structure codebook	GA1	First stage of gain codebook	3
		GB1	Second stage of gain codebook	4
SUBFRAME – II	Pitch details	P2	Pitch delay	5
	Algebraic Codebook	S2	Signs of fixed-codebook pulses	4
		C2	Pulse positions of fixed-codebook	13
	Conjugate structure codebook	GA2	First stage of gain codebook	3
		GB2	Second stage of gain codebook	4
TOTAL NUMBER OF BITS				80

3. COMPUTER EXERCISES

This section provides an overview of the exercises that have been designed to highlight the salient features of the algorithm.

3.1. Split vector quantization

The direct quantization of the LPC results in a poor perceptual quality of the reconstructed speech, since they are very sensitive to the quantization error. The concept of the split vector quantization technique used in ACELP to quantize the LSP parameters can be emphasized by a simple experiment. The frequency response of the quantized and the unquantized LPC (Figure 8) can be viewed with the help of the pre-processing and the LP analysis context menu[3]. The four options provided in the GUI to choose the structure of the codebook (CB) include: VQ with both the first and the second stage codebooks (*L1*, *L2* and *L3*), VQ with the first stage CB (*L1*) only, VQ with the first stage CB and the lower part of the second stage CB (*L1* and *L2*), and VQ with the first stage CB and the upper part of the second stage CB (*L1* and *L3*). The variations in the frequency response of the quantized LPC can be

[3] Context menus can be activated by right clicking on the corresponding blocks.

noticed by choosing the various options provided in the GUI. This exercise is designed to highlight the concepts of the split vector quantization, the filter parameter transformation, and the design of vector codebooks.

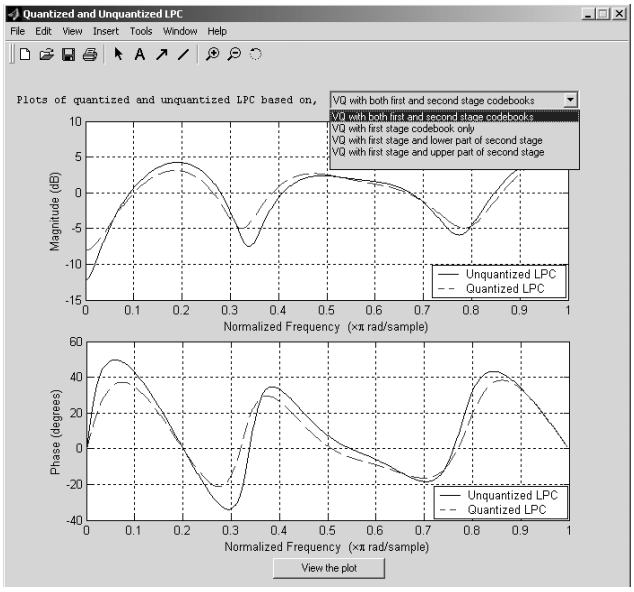

Figure 8. LP analysis context menu showing the plots of the quantized and the unquantized LPC spectra

3.2. Perceptual weighting

The open-loop pitch analysis context menu provides the options to view the frequency response of the perceptual weighting filter, the LP synthesis filter, etc. There are options provided to view the plots of the weighted speech and the autocorrelation of sw(n), etc. With the help of this exercise, one can understand the concept for adaptive perceptual weighting. This example also demonstrates the steps involved in the computation of open-loop pitch, T_{op}.

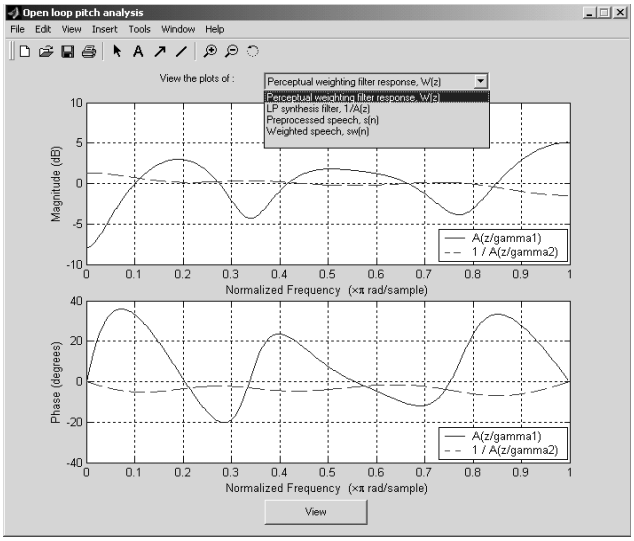

Figure 9. Open-loop pitch analysis context menu showing the frequency response plots of the perceptual weighting filter

3.3. ACB and filtered ACB vector

The plots of the impulse response (h(n)), the ACB vector (v(n)), and the filtered ACB vector (y(n)) can be viewed using the closed-loop pitch analysis context menu. From the equations (Eq-(7) and Eq-(8)), it is very difficult to understand the concepts of the adaptive codebook. With the help of the GUI, one can easily understand, how the past excitation signal is interpolated to compute the adaptive codebook vector.

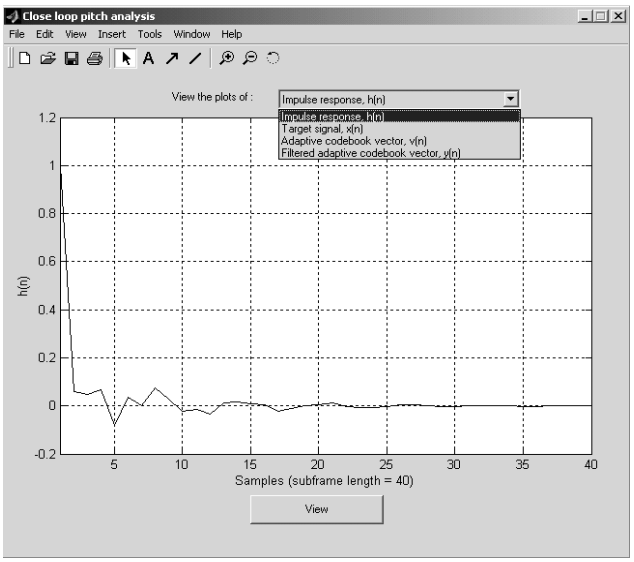

Figure 10. Closed-loop pitch analysis context menu showing the impulse response plot

This exercise helps the student understand the concepts of closed-loop pitch search. This exercise also highlights the innovative method of computing the target signal, by filtering the residual with the LP synthesis and weighting filters.

3.4. Multi-rate codec

Another tricky experiment that can be performed is to vary the rate of the ACELP coder by changing the number of pulses that represent the fixed-codebook vector [5]. A significant variation in the rate of speech codec can be noticed. The algorithm for the fixed (algebraic) codebook search is written in such a way that one can easily change the number of pulses used to represent the fixed codebook vector. The LSP parameters (18 bits), the closed-loop pitch (14 bits), and the codebook gains (14 bits) constitute a total of 46 bits. The number of bits (and hence the bit-rate of the speech codec) to encode the algebraic codebook depends on the number of pulses and the signs of the CB.

Case – 1:

For the number of pulses, k = 5. This results in a total of 15 (5x3) bits to encode pulse positions and 5 bits for signs. Hence, the number of bits required to encode the algebraic CB = 40 bits/frame. The resulting bit rate of the speech codec = 8.6kb/s

Case – 2:

For the number of pulses, k = 3. This results in a total of 11 (3+4+4) bits to encode pulse positions and 3 bits for signs. Hence, the number of bits required to encode the algebraic CB = 28 bits/frame. The resulting bit rate of the speech codec is

7.4kb/s. For the number of pulses, k = 4, the G.729-algebraic codebook (Table-1) is obtained.

Case – 3:

For the number of pulses, k = 10 (2 pulses per track). This results in a total of 30 (10x3) bits to encode pulse positions and 10 bits for signs. Hence, the number of bits required to encode the algebraic CB = 80 bits/frame. The resulting bit rate of the speech codec = 12.6kb/s. By not encoding the sign of the second pulse in the above case, one can achieve a bit rate of 11.6kb/s.

Hence, by varying the number of pulses and the sign bits of the algebraic codebook, one can implement a multi-rate codec. This exercise familiarizes the student with the concepts of adaptive multi-rate codec. Simply, this exercise demonstrates the various governing factors that decide the rate of the speech codec.

3.5. LPC and FFT spectra

This exercise explains the relationship between the LPC and the FFT spectra of the pre-processed speech. From Figure 11, it can be noticed that the LPC spectrum is nothing but the envelope of the FFT spectrum of the pre-processed speech.

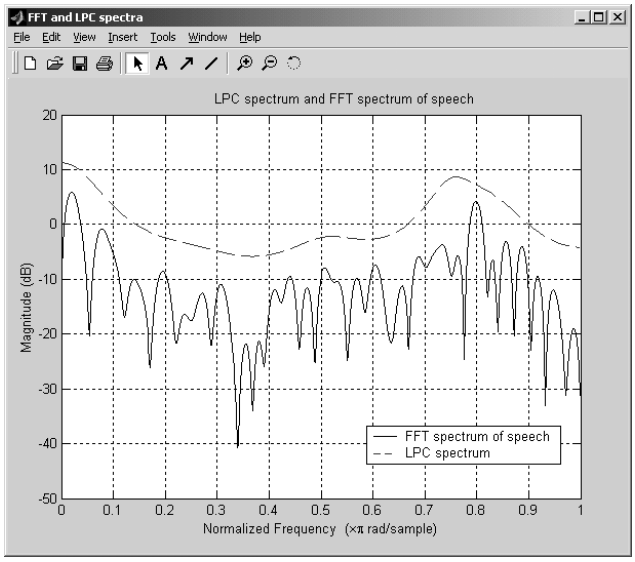

Figure 11. The LPC and the FFT spectra of the pre-processed speech

3.6. Effect of masking specific sets of bits on speech quality

The students are asked to mask specific sets of bits and observe the perceptual quality of the reconstructed speech. The significance of, the LSP, the pitch, the gain and the algebraic codebooks are noted by masking the corresponding parameter bits and examining the perceptual quality of the speech. This experiment gives an overview of the importance of the parameters, and can lead to analyze the trade-offs between the number of bits to represent the parameters and the perceptual quality of speech.

3.7. Performance over various frequency selective channels

One can perform experiments to test the performance of the algebraic CELP codec over various frequency selective/non-selective channels. The exercises that have been outlined here in this section as well as additional ones not described here are given to students in detailed write-ups that have a theory section and step-by-step instructions for operating the software and observing the intended results. A quiz is also part of the exercise and is intended to examine the knowledge gained by the student.

Select exercises will be given to our students as part of a computer project in our undergraduate DSP class in the Fall 2002 semester. The students perform the experiments and respond to the questions in the quiz section. They then have to submit a typed report that describes the results obtained in the computer exercise along with several relevant figures and graphs that are copied in their report using a drag-and-drop process. Select exercises for our graduate class in speech and audio coding included subjective evaluations based on records that are obtained by modifying the speech parameters directly in the MATLAB ACELP code.

4. CONCLUDING REMARKS

This paper presented an education tool used to teach students some of the signal processing concepts embedded in ACELP coder. The tool is developed in MATLAB and represents an entire simulation of the ACELP (G.729) standard [2] along with a user-friendly GUI. This tool will be evaluated in our undergraduate DSP class at ASU. Although complete assessment results are not available as of yet preliminary evaluation by an assessment professional revealed motivation on behalf of the students to learn the concepts behind this Algebraic CELP.

5. REFERENCES

[1] Andreas Spanias, “Speech Coding: A Tutorial Review”, in the *Proceedings of IEEE*, Vol. 82, No.10, pp. 1541-1582, Oct. 1994

[2] ITU-T Recommendation G.729, “Coding of speech at 8kbps using conjugate-structure algebraic code excited linear prediction (CS-ACELP)”, Dec 1995

[3] A. Kataoka et al., “An 8-kb/s Conjugate Structure CELP (CS-CELP) speech coder”, in the *IEEE Trans. on Speech and Audio Processing*, Vol. 4, No. 6, pp. 401-411, Nov 1996

[4] R.Salami et al., “Design and Description of CS-ACELP: A Toll Quality 8 kbps Speech Coder”, *IEEE Trans. Speech and Audio Processing*, vol. 6, No. 2, pp.116-130, Mar 1998

[5] E. Ekudden, et al., “The Adaptive Multi-rate Speech Coder”, in *Proceedings of IEEE Workshop on Speech Coding*, Porvoo, pp.117-119, 1999.