Low Complexity CELP Speech Coding at 4.8 kbps

by Y-H. Kao
Advisor: J.S. Baras

M.S. 92-10

ABSTRACT

Title of Thesis : Low Complexity CELP speech Coding at 4.8 kbps

Name of degree candidate : Yu-Hung Kao

Degree and year : Master of Science, 1990

Thesis directed by : Dr. John S. Baras
Professor
Electrical Engineering Department

Low bit rate, high quality speech coding is a vital part in voice telecommunication systems. The introduction of CELP (1982) (Codebook Excited Linear Prediction) speech coding provides a feasible way to compress speech data to 4.8 kbps with high quality, but the formidable computational complexity required for real-time processing has prevented its wide application. In this thesis, we reduce the computational complexity to 5 MIPS (million instructions per second), which can be handled by even inexpensive DSP chips, while maintaining the same high quality. We hope our contribution can finally make CELP coding a widely applicable technology.

Low Complexity CELP Speech Coding at 4.8 kbps

by

Yu-Hung Kao

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
1990

Advisory Committee :

Professor John Baras, Advisor
Professor Thomas Fuja
Professor Steven Tretter

ACKNOWLEDGEMENT

First I would like to express my sincere gratitude to my advisor Professor John S. Baras. He has created opportunities for me and let me have access to both academia and industry. It has been truly rewarding being one of his students.

I would like to thank Dr. Raja Rajasekaran for providing me the opportunity to do research at Texas Instruments, I got to know many good friends over there : George Doddington, Joseph Picone, Charles Hemphill, Barbara Wheatley, Wallace Anderson, and Abraham Ittycheriah. They have been very helpful and supportive.

Also I would like to thank National Security Agency for providing me with their CELP source code and documentation. Without their outstanding contributions in CELP speech coding, this work will not be possible.

Finally I am grateful to the financial support of SRC and NSF.

Table of Contents

Chapter 1	Introduction	1
Chapter 2	Overview	2
2.1	Preliminaries	2
2.2	Objectives	5
2.3	An Overview of CELP Analysis and Synthesis	5
2.4	CELP Bit Allocation Format	8
2.5	Organization of The Thesis	8
Chapter 3	LPC Analysis	10
3.1	The Computation of LSP	10
3.2	Combination of Root Searching and Quantization	19
3.3	Interpolation of LSP	20
3.4	Complexity of The LPC Analysis	21
3.5	Conversion of LSP to Predictor Coefficients	22
Chapter 4	Pitch Prediction	23
4.1	Pitch Codebook Search	23
4.2	Complexity of Pitch Search	26
4.3	Increased Pitch Prediction Resolution	26
4.4	Two Stage Pitch Search to Reduce Computation	31
4.5	Delta Coding for Pitch Index : to Reduce Both Computation and Bit Rate	32
4.6	Perceptual Weighting	32

Chapter 5 Final Speech Residual VQ 34

5.1 Complexity of Codebook Search 34
5.2 Structured Algebraic Codebook in CELP : for Fast Codebook Search 36
5.3 Fast Algorithm to Compute Inner Products 41

Chapter 6 Quality Results and Suggestions for Future Research 45

References 49

List of Figures

Fig. 2.1	Illustrating the extraction of information from the speech signal in CELP coding	3
Fig. 2.2	CELP Analysis	6
Fig. 2.3	CELP Synthesis	7
Fig. 3.1	Direct form LPC filter (analysis)	12
Fig. 3.2	Direct form LPC filter (synthesis)	12
Fig. 3.3	Vocal Tract Model	14
Fig. 3.4	Lattice Filter (synthesis)	14
Fig. 3.5	Lattice Filter (analysis)	14
Fig. 3.6	Interpolation of LSP	20
Fig. 4.1	Pitch Codebook Search	24
Fig. 4.2	Interpolation for pitch prediction	28
Fig. 4.3	Waveforms and Spectra of interpolated signals	29
Fig. 4.4	Ripple Effect	31
Fig. 5.1	Codebook Search	35
Fig. 5.2	Searching for the optimal codeword	37
Fig. 5.3	A typical codeword	40
Fig. 5.4	Codebook sub-vector combinations for inner product computations	42
Fig. 5.5	Computation of Inner Products	42

CHAPTER 1 : INTRODUCTION

The frame work of CELP (codebook excited linear predictive) coding was suggested by Bishnu S. Atal in 1982. CELP can achieve high speech quality (better than all 16 kbps coding and comparable to 32 kbps CVSD) at very low bit rate (4.8 kbps). But the formidable computational complexity (over 100 MIPS for real-time processing) prevents its application in telecommunication systems. Extensive efforts have been undertaken through the years to reduce CELP complexity in order to make it fit into current DSP technology. The bottleneck of complexity is the search of the stochastic codebook. Several groups [1, 2, 8, 9, 11] have published their results on reducing the codebook search complexity, but the best result known to date can only achieve minor reduction, i.e. the codebook search still dominates in the total computational complexity; and often at the price of non-optimal search (of course, this will degrade speech quality). These algorithms still require about 15 MIPS. We base our work on the NSA version of CELP (proposed federal standard). We find an algebraic codebook, which can be searched optimally for only 0.16 MIPS, and thus we achieve a 5 MIPS algorithm. This reduction in complexity is far better than any other algorithm known to date; and according to preliminary evaluations (by NSA and others), its quality is also better than any other low complexity CELP. Formal subjective evaluation tests are currently being prepared for the evaluation of this algorithm. Because a 5 MIPS algorithm can be easily implemented on a great variety of DSP chips, this technology can be used in practical communication systems. We hope the contribution of our work can finally make low bit rate, high quality speech coding become widely applicable in speech communication systems.

CHAPTER 2 : OVERVIEW

2.1 Preliminaries

As discussed in the introduction the primary attraction of CELP speech coding is that it provides high quality (almost equivalent to toll) speech coding at a low bit rate (4.8 kbps). An important step towards the establishment of CELP as a widely used coding technique is the development of standards. A group at the National Security Agency (NSA) recently proposed a *4.8 kbps* speech coding standard using *Code Excited Linear Prediction (CELP)* for digital radio transmission, Federal Standard 1016, Sept. 1989 [7, 9, 13]. In addition to digital radio applications, CELP is also very suitable for encrypted telephone communications and other applications wherein voice must be digitized prior to encryption. More specifically in order to achieve privacy in cellular techniques, low complexity CELP is required.

CELP is an *analysis by synthesis* type of technique. As in the following diagram (Fig. 2.1), speech information is extracted in three steps [8, 10, 12, 14]:

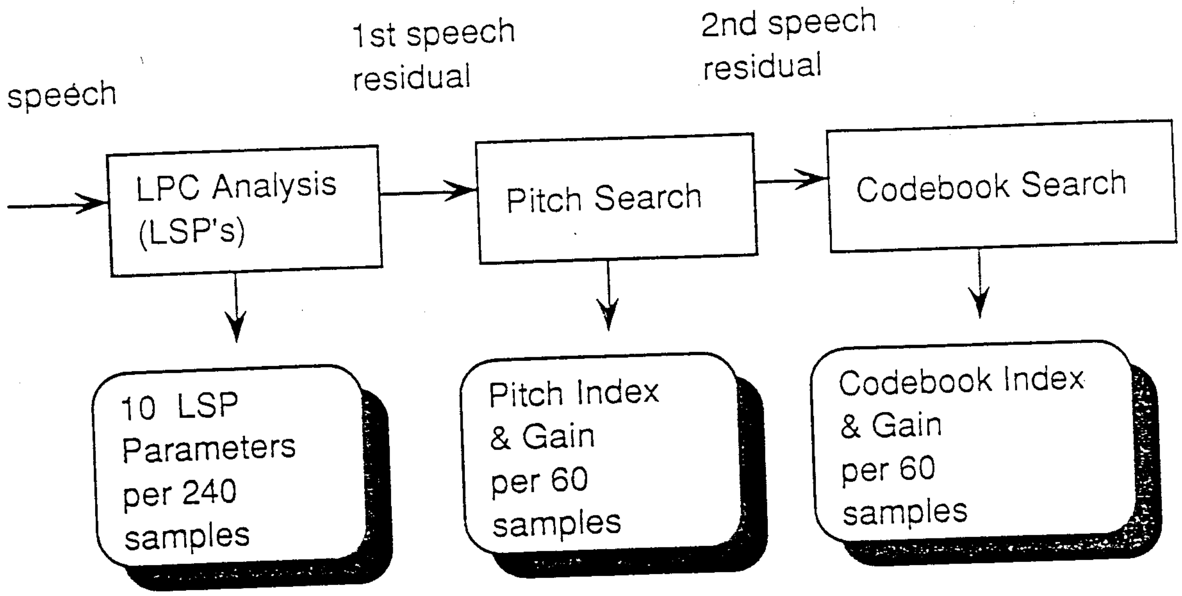

speech
1st speech residual
2nd speech residual
LPC Analysis (LSP's)
Pitch Search
Codebook Search
10 LSP Parameters per 240 samples
Pitch Index & Gain per 60 samples
Codebook Index & Gain per 60 samples

Fig. 2.1 Illustrating the extraction of information from the speech signal in CELP coding.

a) Short term (envelope) information is extracted by the LSP (line spectrum pair) parameters.
b) Long term (pitch) information is extracted by the pitch index and gain.
c) Finally Gaussian vectors with independent components are used to represent the speech residual (an approximation to the "innovations process").

We now provide a brief overview of the CELP vocoder, so that we can proceed with our problem definition. In the following chapters we will explain each stage and operation in detail.

Speech coders can be classified into two major categories : waveform coders and vocoders. Waveform coders [16] (PCM, ADPCM) encode the digitized speech signal "sample by sample", so they can achieve good quality at the price of high bit rate (> 32 kbps). However, if we look into the mechanism of generating the speech signal or simply by looking at the speech waveform, it is rather obvious that there are many redundancies in the signal. Therefore it is not necessary to encode speech "sample by sample". Instead, we can encode a "block of samples" by extracting "features" from the signal, this is precisely the idea of the vocoder.

The difference between waveform coders and vocoders can be stated as follows : waveform coders process "a sample" at a time, but vocoders process "a block of samples" at a time. Waveform coders can be used to encode all kinds of sound signals, because they do not explore the "nature" of the signals, for instance they can be used equally well to encode both music and speech. Vocoders, on the other hand, are "source dependent"; for example, the CELP vocoder is for speech only, because it explores the special mechanism of speech generation, which is not valid for music.

As is well known [16] the mechanism generating speech signals can be classified into two categories :

1) Voiced sound : the vocal chord generates a vibration, which is subsequently modulated by the vocal tract.
2) Unvoiced sound : there is no vocal chord vibration. There is only air flow, which is subsequently modulated by the vocal tract.

Therefore we can see that two kinds of information are involved in speech : 1) vocal chord vibration generating different frequencies, which can be treated as FM information; 2) vocal tract modulation shapes the envelope of the speech signal, which can be treated as AM information. If we look at a real speech waveform, it looks like an FM + AM signal.

The task of CELP vocoder is to extract these two types of information from the speech signal in an efficient way. LPC analysis models the vocal tract, which captures AM information. Pitch detection models the vocal chord vibration, which captures FM information. If we only extract AM and FM information, the reconstructed speech sounds rough. In CELP we use a third stage :

"vector quantization" (VQ) to encode the "speech residual" in order to make the reconstructed speech sound more natural. Of course, the quality of the reconstructed speech depends on the size of the VQ codebook; the larger the better. As mentioned earlier the critical problem is that the required codebook search is very expensive computationally. For a 512 codebook size, if we use a "random" codebook, CELP requires 100 MIPS for real time processing [9]. If we use "overlapped" codebook, CELP still requires 20 MIPS [9]. Ever since the beginning of CELP research in 1982 [10], "the reduction of computational complexity" has been the major concern [1, 2, 8, 9, 11]. This reduction is precisely the topic of this thesis.

2.2 Objectives

Due to the extensive codebook search involved, the CELP algorithm requires about *20 MIPS* processing power (for codebook size of 512) to run in real time [9, 13]. Our research interest is to find a new way to structure the codebook so that we can replace the time consuming linear search with some efficient heuristics. Together with other algorithmic approximations and heuristics that we will introduce, our objective is to show that the computational complexity can be reduced to *under 10 MIPS*; which can be handled by a single TMS320C30 chip. Actually further reduction can allow implementation in many inexpensive single chips, giving the way for a variety of applications.

2.3 An Overview of CELP Analysis and Synthesis

Figure 2.2 provides a schematic diagram of the analysis part of CELP speech coding . Figure 2.3 provides a schematic diagram of the synthesis part of CELP speech coding. The various blocks will be described in detail in the subsequent chapters. The analysis part determines the 10 LSP parameters, the pitch codebook index and gain, and the codebook index and gain that have to be transmitted to the decoder. The synthesis part is relatively straight forward as Figure 2.3 illustrates. As shown in the figure, traditional CELP synthesis uses the Gaussian codebook vector and the gain to scale it, the pitch codebook vector and the gain to scale it, to produce a combined (additive excitation) for the LPC filter, whose coefficients are updated on-line.

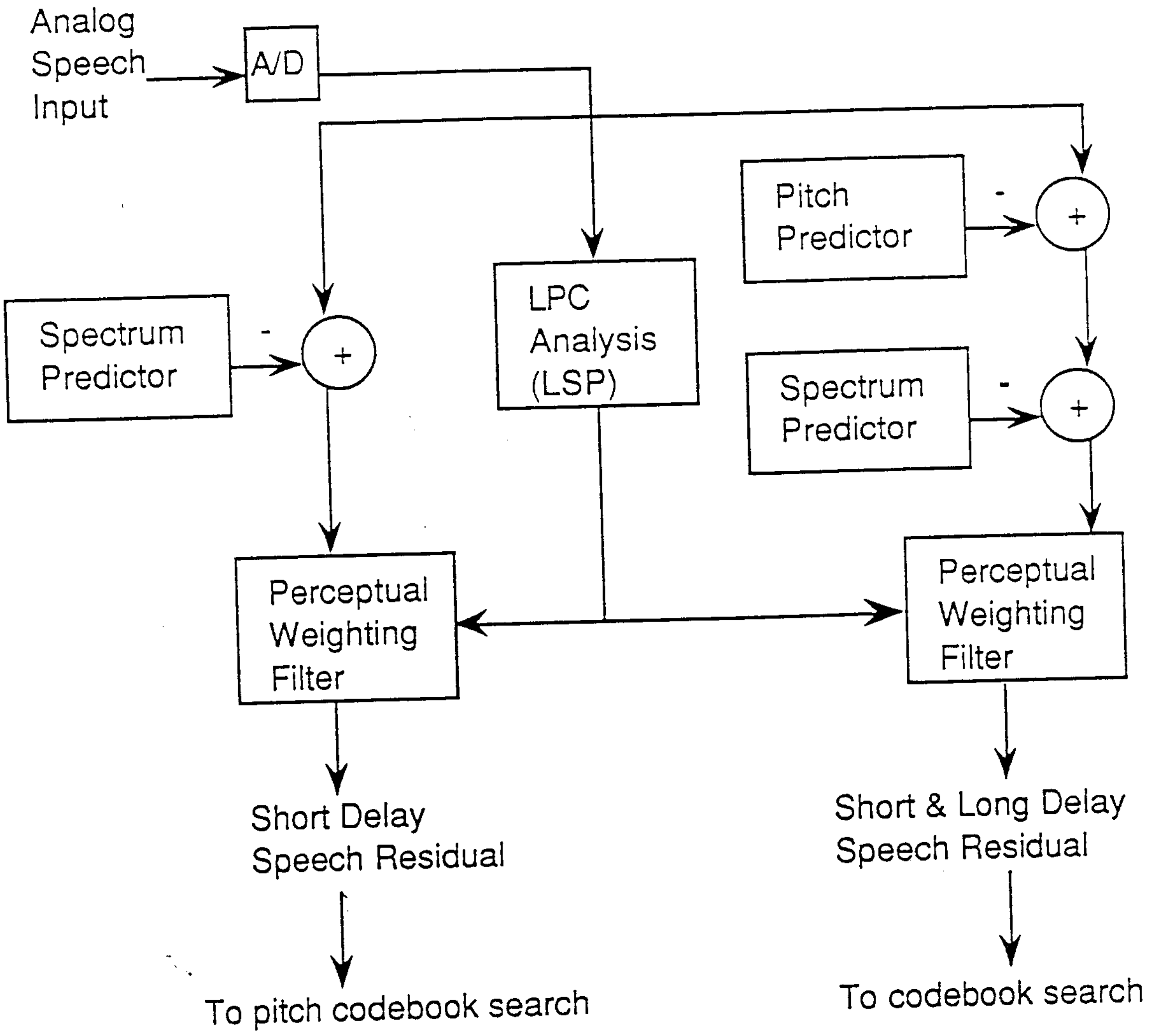


Fig. 2.2 CELP Analysis

1) middle column : LPC analysis (spectrum information). The computation complexity is negligible.
2) left column : subtract spectrum information from original speech, then perform pitch prediction. The computation complexity is about 3 MIPS.

3) right column : subtract both spectrum information and pitch information from original speech, then perform speech residual VQ. The computation complexity is 90 MIPS for random codebook, 9 MIPS for overlapped codebook (NSA), and 0.16 MIPS for the algebraically constructed codebook presented in this thesis.

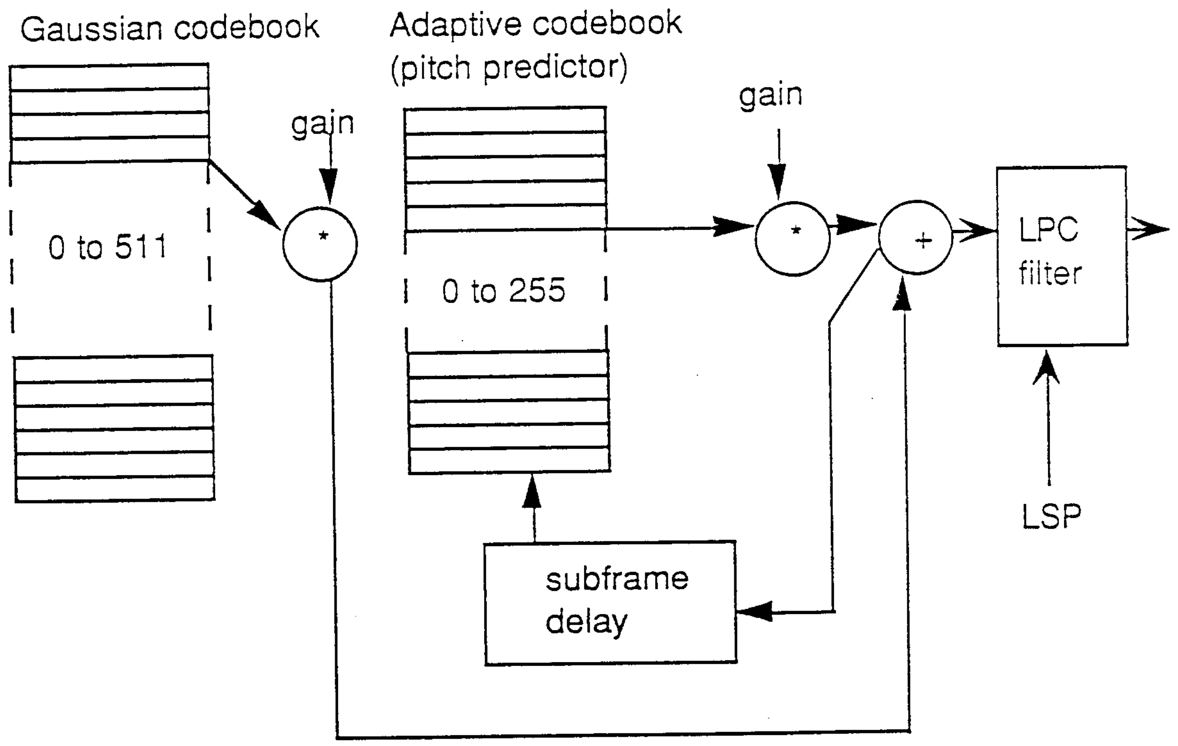


Fig. 2.3 CELP Synthesis

2.4 CELP Bit Allocation Format

Throughout this thesis we will use the following bit allocation scheme [9, 13].

	Spectrum	Pitch	Codebook
Update rate	30 ms 240 samples	30 / 4 = 7.5 ms 60 samples	30 / 4 = 7.5 ms 60 samples
Order	LPC 10	256 delays*60 & 1 gain	512 vectors*60 & 1 gain
Analysis	Open loop Correlation 30 ms Hamming	Closed loop MSPE VQ Delay range : 20 to 147	Closed loop MSPE VQ 512 vectors
Bit / Frame (240 / Frame)	34 bits for 10 LSP [3444433333]	index : 8+6+8+6 gain (-1, 2) : 5*4	index : 9*4 gain (1, 1550) : 5*4
Bit rate	1133.3 bps	1600 bps	1866.67 bps

Total bit rate = 4.6 K bps.

2.5 Organization of The Thesis

The difficult part of CELP is the analysis. It is very computationally intensive. This thesis is focusing on reducing the required computations. There are three steps in the analysis : LPC analysis, pitch prediction, and final speech residual vector quantization. They are discussed in detail in chapters 3, 4 and 5 respectively. We review the *theory* behind them, introduce the *basic algorithms*, and describe some *refinements* of these algorithms, (some of them to improve speech quality, some of them to reduce computation). *Computational complexity* of these algorithms is discussed throughout the thesis. Our contribution is primarily in simplifying the *codebook search algorithm*, which is described in chapter 5. After presenting the whole analysis procedure in chapters 3, 4, and 5; in chapter 6, we use some *objective measures* to evaluate the speech quality. Actually there are *no "good objective" measures for evaluating speech quality of the coders*; the only way to determine if the resulting speech quality is "good" is by *subjective listening tests*. These objective measures are used here for reference purposes only.

CHAPTER 3 : LPC ANALYSIS

3.1 The Computation of LSP

The first step of CELP analysis is short term prediction, i.e. to extract *envelope (spectrum) information*. The result of the LPC analysis is an all-zero predictor filter, or a corresponding all-pole synthesis filter. The parameters of this filter can be transmitted directly (*LPC coefficients*); or we can use the equivalent, lattice form reflection coefficients (*PARCOR*), to represent the filter. Usually PARCOR are preferred due to their low spectral sensitivity.

Recently, there has been a growing interest in the use of *Line Spectrum Pairs (LSP)* to code the filter parameters for LPC filtering [3, 4, 5]. The LSP representation is equivalent to the LPC and PARCOR. LSP can encode speech spectrum *more efficiently* than other parameters. This can be attributed to the intimate relationship between the LSP and the *formant frequencies*. Accordingly, the LSP can be quantized taking into account spectral features known to be important in perceiving speech signals. In addition, the LSP are suitable for frame-to-frame *interpolation* with smooth spectral changes because of their frequency domain interpretation.

It is important to emphasize here that all these three kinds of parameters (LPC, PARCOR, LSP) are *mathematically equivalent*. They are all derived from LPC analysis. If we use double precision numbers to represent these parameters, they will give the *same results*. The point is that because we want to *quantize* them to reduce data rate, quantization will cause inaccuracy of parameter values (quantization error), and same quantization error in different parameters will cause different degree of distortion of resulting speech quality. That is why we study the property of these different parameters, and try to choose the best one; i.e. the one which gives minimum distortion for a fixed data rate. (We'll see that PARCOR is better than LPC, and LSP is better than PARCOR).

Now we turn to *efficient computation of LSP*. This will involve an iterative root finding algorithm for a series representation in Chebyshev polynomials. Generally, it is not trivial to find the roots of a polynomial. Fortunately, the LSP polynomials have some nice properties that allow us to alleviate the complicated root finding problem.

We begin with the basic LPC-10 prediction error filter [16]:

$$A(z) = 1 - \sum_{k=1}^{10} a(k) z^{-k}$$

The $\{a(k)\}$ are the direct form predictor coefficients (LPC coefficients). The corresponding all-pole synthesis filter has transfer function $1/A(z)$. The analysis and synthesis operations using these filters are shown schematically in figures 3.1 and 3.2.

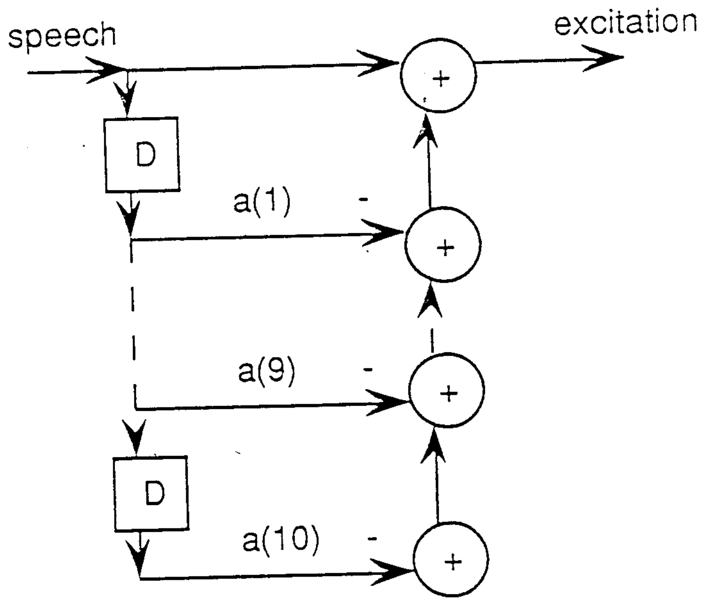


Fig. 3.1 Direct Form LPC filter (analysis)

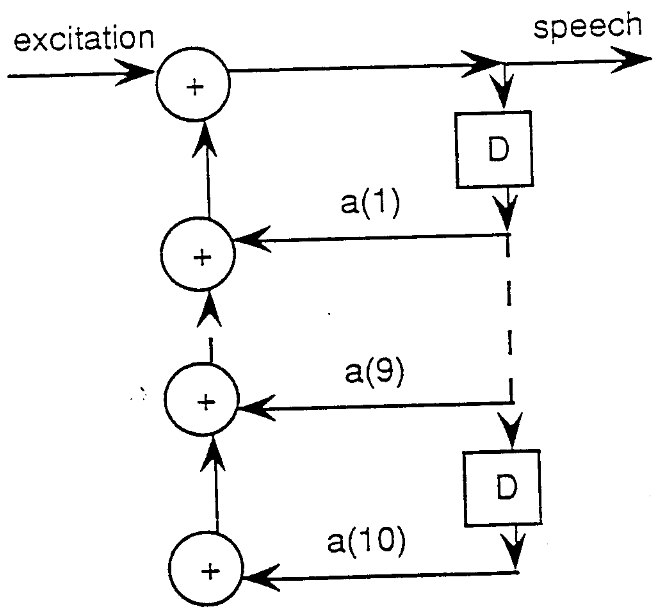


Fig. 3.2 Direct form LPC filter (synthesis)

A symmetric polynomial $F_1(z)$ and an antisymmetric polynomial $F_2(z)$, related to A(z), are formed by adding and subtracting the time-reversed system function as follows:

$$F_1(z) = A(z) + z^{-11}A(z^{-1})$$
$$F_2(z) = A(z) - z^{-11}A(z^{-1})$$

The roots of these two polynomials determine the LSP. At first glance, it seems worse, because now we need 22 (possibly complex (so it is 44 real) numbers) roots, instead of 10 real coefficients (both LPC and PARCOR are 10 real numbers), to represent the LPC filter, which is a 4.4 times increase in data. However, because of the special property of $F_1(x)$ and $F_2(x)$, we can actually encode the LPC filter more efficiently using LSP.

The *physical meaning* of the LSP is also important [5]. The two polynomials $F_1(x)$ and $F_2(x)$ have the interpretation of being the system polynomials for an 11 coefficient predictor derived from a lattice structure. The first 10 stages of the lattice have the same response as the original 10 stage predictor. An additional stage is added with reflection coefficients equal to +1 or -1 to give the response of $F_1(z)$ or $F_2(z)$, respectively. If the vocal tract characteristics can be expressed by $1/A(z)$, the vocal tract is modeled as a non-uniform section acoustic tube consisting of 10 sections of equal length. The acoustic tube is open at the terminal corresponding to the lips, and each section is numbered beginning from the lips. Mismatch between the adjacent sections n and n+1 causes wave propagation reflection. *The reflection coefficients are equal to the PARCOR parameters k_n*. Section 11, which corresponds to the glottis, is terminated by mismatched impedance. The excitation signal applied to the glottis drives the acoustic tube.

The PARCOR lattice filter is regarded as a digital filter equivalent to the following acoustical model in figures 3.3, 3.4, and 3.5 :

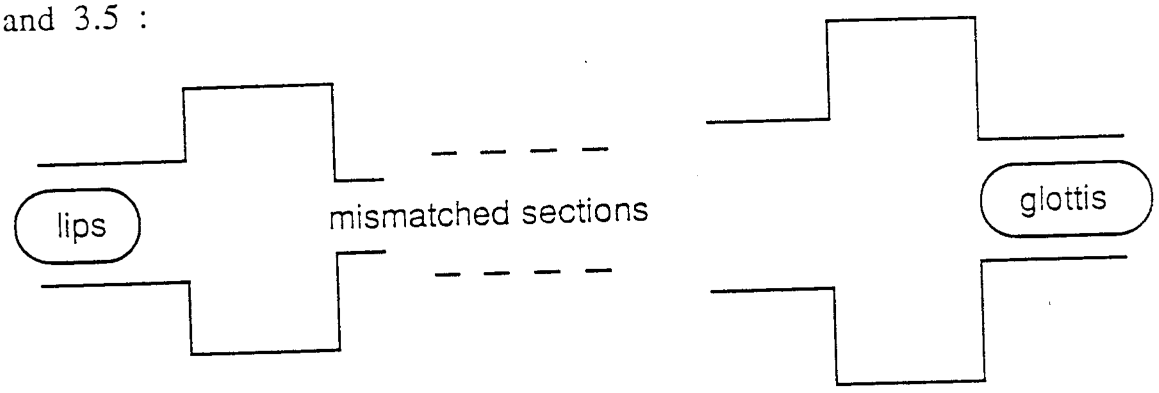


Fig. 3.3 Vocal Tract Model

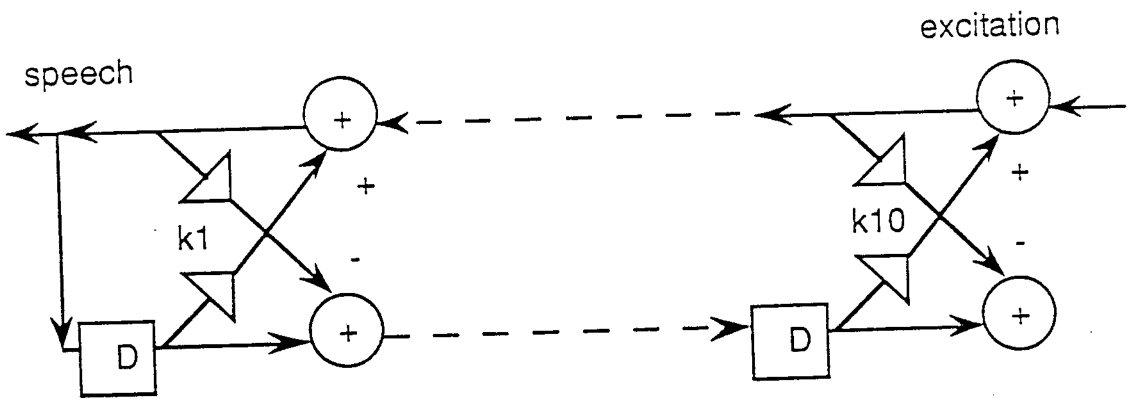


Fig. 3.4 Lattice Filter (synthesis)

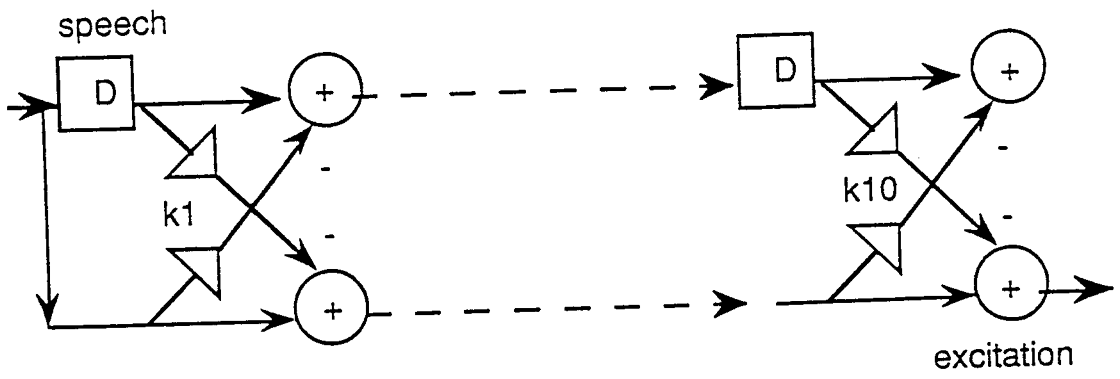


Fig. 3.5 Lattice Filter (analysis)

In PARCOR analysis, the boundary condition at glottis is impedance-matched. Now add the 11th stage, where the tube is completely closed or open at the glottis. These condition correspond to $k_{11} = 1$ and $k_{11} = -1$. A(z) should be changed to $F_1(z)$ and $F_2(z)$. The acoustic tube is now of the *loss-less* type, and the transfer function displays *resonant frequencies.*

The polynomials $F_1(z)$ and $F_2(z)$ being symmetric and antisymmetric, respectively, have roots at $z = 1$ and $z = -1$, which can be removed as follows :

$$G_1(z) = \frac{F_1(z)}{1+z^{-1}} \quad \text{and} \quad G_2(z) = \frac{F_2(z)}{1-z^{-1}}$$

The resulting $G_1(z)$ and $G_2(z)$ are symmetric polynomials of order 10. Since the roots occur in complex conjugate pairs, it is only necessary to determine the roots located on the upper semicircle. The roots of interest are of the form $\exp(j\omega_i)$, $i = 1, 2, \ldots, 10$. The LSP are the angular positions of the roots $0 < \omega_i < \pi$.

Since $G_1(z)$ and $G_2(z)$ are both symmetric polynomials of order 10, we can write them as

$$G_1(z) = 1 + g_{1,1}z^{-1} + g_{1,2}z^{-2} + \ldots\ldots + g_{1,5}z^{-5} + \ldots\ldots\ldots + g_{1,1}z^{-9} + z^{-10}$$
$$G_2(z) = 1 + g_{2,1}z^{-1} + g_{2,2}z^{-2} + \ldots\ldots + g_{2,5}z^{-5} + \ldots\ldots\ldots + g_{2,1}z^{-9} + z^{-10}$$

$G_1(z)$ contributes 5 pairs of conjugate zeros and $G_2(z)$ contributes 5 pairs of conjugate zeròs; all on the unit circle. The linear phase term can be removed to give two zero phase series expansions in cosine

$$G_1(e^{j\omega}) = e^{-j5\omega}G_1'(\omega)$$
$$G_2(e^{j\omega}) = e^{-j5\omega}G_2'(\omega)$$

where

$$G_1'(\omega) = 2\cos(5\omega) + 2g_{1,1}\cos(4\omega)+\ldots+2g_{1,4}\cos(\omega) + g_{1,5}$$
$$G_2'(\omega) = 2\cos(5\omega) + 2g_{2,1}\cos(4\omega)+\ldots+2g_{2,4}\cos(\omega) + g_{2,5}$$

Now we can see that we don't need 22 complex roots, as it appeared initially; instead, we only need 10 real frequencies, which is the same amount of data as in LPC or PARCOR parameters. Since LPC, PARCOR, and LSP are all 10 real numbers (same amount of data), it is important to compare their properties :

1) LPC : these parameters have no bound, so it is difficult to define the quantization region. In addition, they are very sensitive with respect to spectrum; small quantization error may cause unstable filter. So they are *not preferred for quantization and transmission.*

2) PARCOR : these parameters are bounded within [-1, 1], which is good for quantization. They are less spectrum sensitive than LPC, and as long as they are within [-1, 1], the filter is *stable.* So they are better than LPC parameters.

3) LSP : since the LSP are resonant frequencies, they are bounded by human physical features. And what is better than PARCOR is that we *don't need to specify the LSP order.* In PARCOR, we not only need to know the values of 10 real numbers, but also need to know which one is the first one, which one is the second one, etc . However, in LSP, knowing the 10 real numbers is enough (due to their *ordering property*). In addition, they are *less spectrum sensitive* than PARCOR. If we use same number of bits to encode PARCOR and LSP, the LSP will give less spectrum distortion.

Consider the substitution $x = \cos(\omega)$, and $\cos(m\omega) = T_m(x)$. $T_m(x)$ is the mth-order Chebyshev polynomial in x. It satisfies the following order recursion

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$
$$(\text{ i.e. } \cos(kx) = 2\cos(x)\cos[(k-1)x] - \cos[(k-2)x] \text{ })$$

with initial condition, $T_0(x) = 1$ and $T_1(x) = x$. The series expansion in cosines can be expressed in terms of Chebyshev polynomials

$$G_1'(x) = 2T_5(x) + 2g_{1,1}T_4(x) + \ldots + 2g_{1,4}T_1(x) + g_{1,5}$$
$$G_2'(x) = 2T_5(x) + 2g_{2,1}T_4(x) + \ldots + 2g_{2,4}T_1(x) + g_{2,5}$$

Once the roots $\{x_i\}$ of $G'_1(x)$ and $G'_2(x)$ are determined, the corresponding LSP are given by $\omega_i = \arccos(x_i)$. The mapping $x = \cos(\omega)$ maps the upper semicircle of the z-plane to the real interval $[-1, +1]$. Therefore, all the roots $\{x_i\}$ lie between -1 and +1, with the root corresponding to the lowest frequency LSP being the one nearest to +1.

$T_i(k)$ is a cosine function, which is difficult to evaluate, and $G'_1(x)$ and $G'_2(x)$ each needs to evaluate five $T_i(k)$'s. The search for roots needs to evaluate $G'_1(x)$ and $G'_2(x)$ at every grid point and look for sign changes. The evaluation of the cosine function is simply intolerable. Fortunately, using the recursion relationship, we can compute $G'_1(x)$ and $G'_2(x)$ with only one cosine evaluation instead of five.

To see this, suppose the polynomial to be evaluated is [3]

$$Y(x) = \sum_{k=0}^{5} c_k T_k(x)$$

and consider the backward recursive relationship

$$b_k(x) = 2xb_{k+1}(x) - b_{k+2}(x) + c_k$$

with initial condition $b_i(x) = 0$, for $i > 5$.

Then Y(x) can be represented in terms of $b_0(x)$ and $b_2(x)$, as follows :

$$\begin{aligned}
Y(x) &= \sum_{k=0}^{5} c_k T_k(x) \\
&= \sum_{k=0}^{5} [b_k(x) - 2xb_{k+1}(x) + b_{k+2}(x)]T_k(x) \\
&= [b_0(x) - 2xb_1(x) + b_2(x)]T_0(x) \\
&\quad +[b_1(x) - 2xb_2(x) + b_3(x)]T_1(x) \\
&\quad +[b_2(x) - 2xb_3(x) + b_4(x)]T_2(x) \\
&\quad +[b_3(x) - 2xb_4(x) + b_5(x)]T_3(x) \\
&\quad +[b_4(x) - 2xb_5(x) + b_6(x)]T_4(x) \\
&\quad +[b_5(x) - 2xb_6(x) + b_7(x)]T_5(x)
\end{aligned}$$

using $T_k(x) - 2xT_{k-1}(x) + T_{k-2}(x) = 0$
for k = 2, 3, 4, 5
and $b_6(x) = b_7(x) = 0$

$$\begin{aligned}
&= [b_0(x) - 2xb_1(x)]T_0(x) \\
&\quad +b_1(x)T_1(x)
\end{aligned}$$

using $T_0(x) = 1$ and $T_1(x) = x$

$$\begin{aligned}
&= b_0(x) - 2xb_1(x) + xb_1(x) \\
&= b_0(x) - xb_1(x)
\end{aligned}$$

using $b_0(x) = 2xb_1(x) - b_2(x) + c_0$

$$\begin{aligned}
&= xb_1(x) - b_2(x) + c_0 \\
&= \frac{b_0(x) + b_2(x) - c_0}{2} - b_2(x) + c_0 \\
&= \frac{b_0(x) - b_2(x) + c_0}{2}
\end{aligned}$$

Using $b_k(x) = 2xb_{k+1}(x) - b_{k+2}(x) + c_k$, we see that we only need to evaluate one cosine, i.e. x, then by the recursive relationship we can calculate $b_0(x)$ and $b_2(x)$ by 5 MUL and 10 ADD.

Since now we can calculate $G'_1(x)$ and $G'_2(x)$ *by only evaluating one cosine, 5 MUL, and 10 ADD,* the search for roots proceeds backwards from evaluating $G'_1(1)$, $G'_1(1-\delta)$, $G'_1(1-2\delta)$, $G'_1(1-3\delta)$, ..., to $G'_1(0)$.

We need to decide δ to avoid causing two or more roots to fall into one interval. Experiments indicate that, $\delta = 0.02$ is small enough. Because the roots of $G'_1(x)$ and $G'_2(x)$ are interleaved one by one, we also have to make sure each pair of roots (one from $G'_1(x)$ and one from $G'_2(x)$) keep their correct order. Again experimentally, the root uncertainty ε must be smaller than 0.0015.

Now we can solve for 10 roots $\{x_0 < x_1 < x_2 ..... < x_9\}$, where $\{x_i, \text{ i even}\}$ are roots of $G'_1(x)$, $\{x_i, \text{ i odd}\}$ are roots of $G'_2(x)$. Following the last paragraph they have to satisfy the following precision : $\min(x_i - x_{i-2}) > \delta$ and $\min(x_i - x_{i-1}) > \varepsilon$. The LSP are given by $\arccos(x_i)$.

3.2 Combination of Root Searching and Quantization

Since we need to quantize the LSP, *it doesn't make sense to solve for the roots to a high precision and then lose it because of quantization.* According to the above analysis, to keep the correct order of roots, we need $\varepsilon = 0.0015$ precision. Since the roots are in [-1, 1], this means that we have to evaluate 2 / 0.0015 = 1333 grid points in the search. Since we use only 3 or 4 bits to quantize each root, such a precision is not necessary. We can combine the root search with quantization, making the whole process faster. For example, if we use 4 bits to quantize a particular root, we only need to locate this root within 16 possible intervals, which means only 16 grid point evaluations instead of 1333. And because we only need to evaluate very few grid points, we can use a *cosine table* to get rid of the cosine evaluation by table lookup. Thus the search for one root can be done with only a few hundreds of operations.

There is a problem with this scheme : the correct order of roots' values may be switched due to low precision. However, since we know the correct order, if $x_i > x_j$, where $i < j$, we can simply assign x_i to its next lower quantization value or x_j to its next higher quantization value to switch them back to the right order.

3.3 Interpolation of LSP

Because the LSP corresponds to resonant frequencies of the vocal tract, they will not change radically. We can *interpolate them without causing too much distortion.* That is why we can use one set of LSP per 240 samples (frame) instead of 60 samples (sub-frame). In the sub-frame processing, we need LSP to calculate the speech residual; these LSP are the results of interpolating. This is illustrated in figure 3.6 below.

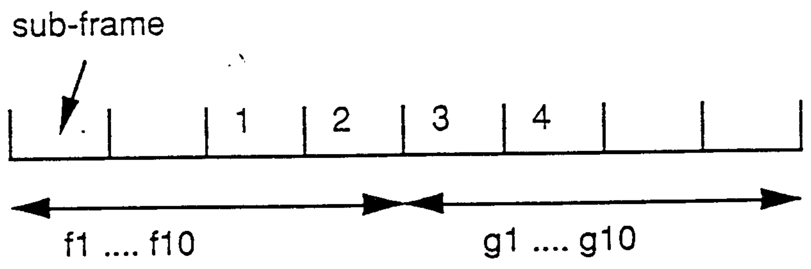


Fig. 3.6 Interpolation of LSP

Let f_1 ... f_{10} be the LSP calculated from the LPC analysis of the first 4 sub-frames, and g_1 .... g_{10} from the following 4 sub-frames. The LSP for sub-frame 1 are interpolated by (7/8)f + (1/8)g, (5/8)f + (3/8)g for sub-frame 2, (3/8)f + (5/8)g for sub-frame 3, and (1/8)f + (7/8)g for sub-frame 4.

This interpolation scheme will cause two sub-frame (15 ms) encoding delay.

3.4 Complexity of The LPC Analysis

Using the combined root-search & quantization schemes described in this chapter, let us suppose that we use 4 bits to encode each of the 10 LSP (actually, for some we use 3 bits (see section 2.4), we calculate an upper bound of the computational complexity). For each root, we have to evaluate 16 grid points, each grid point evaluation needs 1 cosine table lookup, 5 MUL, and 10 ADD. That's (1+5+10) * 16 = 256 operations for each root, 2560 operations for 10 LSP. This LPC analysis is done per 240 samples (30ms). So the computational complexity is 2560 / 30 ms = 85333 = 0.0853 MIPS, which is negligible compared to the pitch codebook search or codebook search.

3.5 Conversion of LSP to Predictor Coefficients

There exists a filter structure which can use the LSP directly in its series of second-order sections. However, this kind of structure requires more arithmetic operations than a direct form filter using predictor coefficients. The trade-off is between this extra computation and the (LSP => predictor coefficients) conversion. For reasonable frame lengths, the (LSP => predictor coefficients) conversion and direct form filter result in lower operations. So we decide to convert the LSP to LPC and then use the LPC filter instead of using the LSP filter directly.

The conversion is very straightforward. Each ω_i gives rise to a second order polynomial $(1 - 2\cos(\omega_i)z^{-1} + z^{-2})$. These polynomials can then be multiplied to form the predictor polynomial utilized in LPC.

CHAPTER 4 : PITCH PREDICTION

4.1 Pitch Codebook Search

The second step in CELP analysis is to extract pitch information, which is also called long term prediction. It is simply using one of previous frames (20 to 147 delays) to represent the current frame. The search scheme is illustrated in the following diagram of figure 4.1.

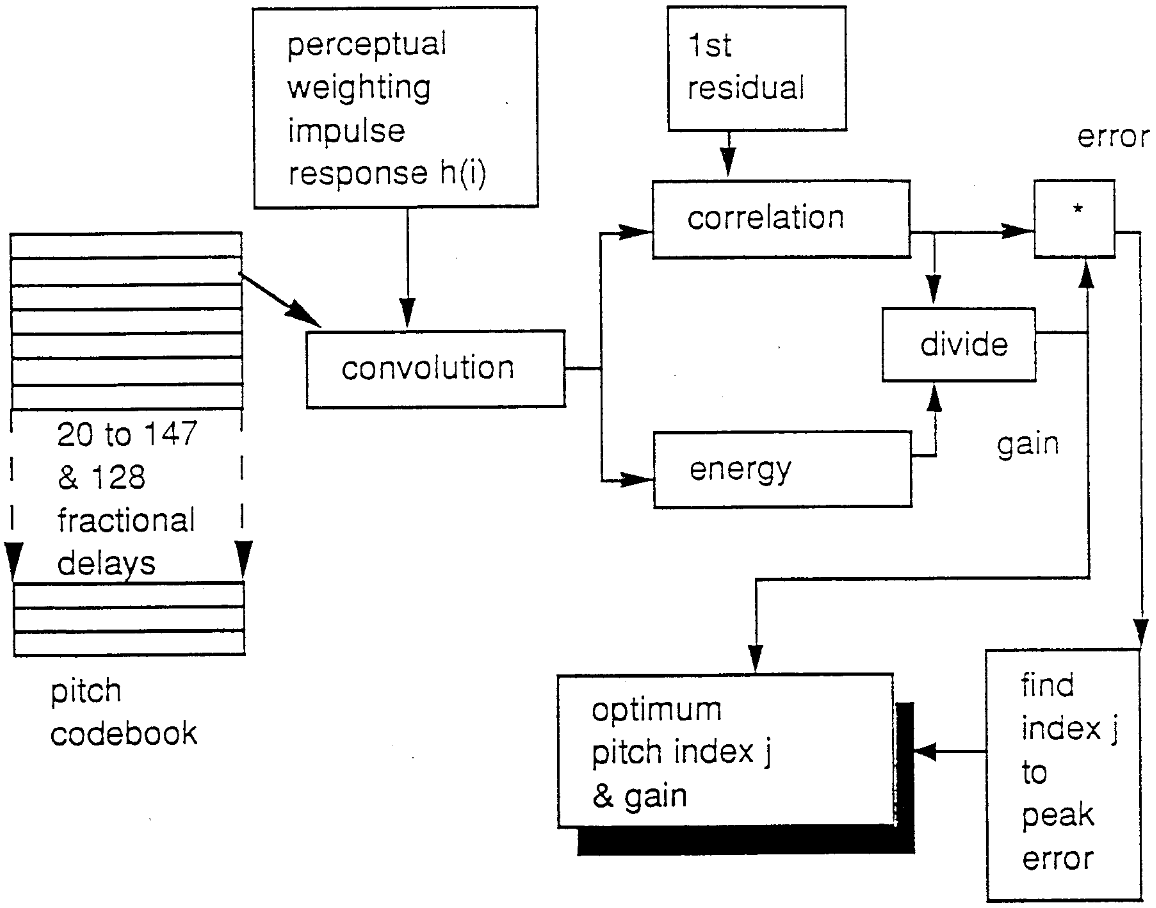

perceptual weighting impulse response h(i)
1st residual
error
correlation
*
divide
convolution
20 to 147 & 128 fractional delays
energy
gain
pitch codebook
optimum pitch index j & gain
find index j to peak error

Fig. 4.1 Pitch Codebook Search

Because the pitch codebook is *overlapped* (i.e. each vector (frame, 60 samples) is just a shift of the previous vector, and contains only 1 new element), we can use the *end point correction* technique [9, 13] to reduce the operations needed in computing the perceptual weighted vectors.

Suppose the first codebook vector is $\{v(0), v(1),\ldots, v(59)\}$, the perceptual weighting impulse response is $\{h(0), h(1),\ldots, h(9)\}$, and the vector after perceptual weighting is $\{y_0(0), y_0(1),\ldots, y_0(59)\}$. Then

```
                    9 8 7 6 5 4 3 2 1 0             h's
y0(0)  =                              0 1 2 3 4 .... v's
y0(1)  =                            0 1 2 3 4 5 .... v's
y0(2)  =                          0 1 2 3 4 5 6 .... v's
...............................................
y0(59) =  ...........55 56 57 58 59                  v's
```

The next codebook vector, $\{v(1), v(2),...., v(60)\}$, $\{y_1()\}$ will be given by,

```
                    9 8 7 6 5 4 3 2 1 0             h's
y1(0)  =                              1 2 3 4 5.... v's
y1(1)  =                            1 2 3 4 5 6.... v's
y1(2)  =                          1 2 3 4 5 6 7.... v's
...............................................
y1(59) =  ...........56 57 58 59 60                 v's
```

Therefore, more precisely

$$
\begin{aligned}
y_0(0) &= h(0)*v(0) \\
y_0(1) &= y_1(0) + h(1)*v(0) \\
y_0(2) &= y_1(1) + h(2)*v(0) \\
y_0(3) &= y_1(2) + h(3)*v(0) \\
&\cdots\cdots \\
y_0(9) &= y_1(8) + h(9)*v(0) \\
y_0(10) &= y_1(9) \\
&\cdots\cdots \\
y_0(59) &= y_1(58)
\end{aligned}
$$

By the above end point correction, we can start from computing $\{y_{127}()\}$, then $\{y_{126}()\}$, $\{y_{125}()\}$,.... to $\{y_0()\}$. (Until now, we only considered integer delays)

4.2 Complexity of Pitch Search

The computation can be attributed to 3 major parts : convolution, correlation, and energy.

Convolution :
For the first vector, it needs 1+2+...+10+10+....+10 (60 terms) = 555 MUL, and 1+2+...+9+9+...+9 (59 terms) = 495 ADD. For all the following vectors, it only needs 9 MUL and 9 ADD for end point correction. Thus totally we need 555 + 495 + 18*127 = 3336 operations for 128 vectors.

Correlation :
For each correlation, we need 60 MUL and 59 ADD. So totally we need (60+59)*128 = 15230 for 128 correlations.

Energy :
Same as correlation: 15230 operations.

These operations have to be done per 60 samples (7.5 ms), therefore we need (3336+15230+15230) / 7.5 ms = 4.5 MIPS. This is not final, as we shall see we have some other ways to cut the complexity further.

4.3 Increased Pitch Prediction Resolution

We find that *pitch resolution* is very important, especially for high pitched speakers. However, the resolution of pitch prediction is bounded by the sampling rate, 8 kHz. In order not to increase the original speech data sampling rate, we need to *interpolate* our speech samples, which means increasing the sampling rate "internally".

Suppose we want to increase the sampling rate by a factor L. This process implies that we have to interpolate L - 1 new samples between each pair of original samples of x(n). This process is similar to digital-to-analog conversion, in which all continuous-time values of a signal x(t) must be interpolated from the sequence x(n) [15].

Suppose L = 3, and the input signal x(n) is "filled in" with L - 1 = 2 zeros between each pair of samples of x(n), giving

$$w(m) = \begin{cases} x(\frac{m}{L}) & ,m = 0, \pm L, \pm 2L, \pm 3L, \ldots\ldots \\ 0 & ,\text{otherwise} \end{cases}$$

The resulting Z-transform becomes

$$\begin{aligned} W(z) &= \sum_{m=-\infty}^{\infty} w(m) z^{-m} \\ &= \sum_{m=-\infty}^{\infty} x(m) z^{-mL} \\ &= X(z^L) \end{aligned}$$

where X( ) is the Z-transform of x( ).

Evaluating W(z) on the unit circle, $z = \exp(j\omega')$, gives the result

$$W(e^{j\omega'}) = X(e^{j\omega' L})$$

which means the spectrum of w(m) contains not only the baseband frequencies of interest ($-\pi/L$ to π/L) but also *images* of the baseband centered at harmonics of the original sampling frequency $2\pi/L$, $4\pi/L$, ... To recover the baseband signal and eliminate the unwanted image components, it is necessary to filter the signal w(m) with a digital low pass filter (anti-imaging) which approximates the ideal characteristic

$$H(e^{j\omega'}) = \begin{cases} G & |\omega| \le \frac{\pi}{L} \\ 0 & \text{otherwise} \end{cases}$$

The signal after anti-imaging filtering is

$$Y(e^{j\omega'}) = H(e^{j\omega'})X(e^{j\omega' L})$$

The interpolation system should look like the following diagram in figure 4.2 :

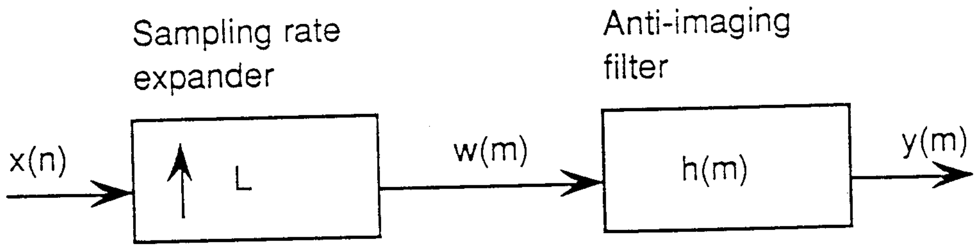


Fig. 4.2 Interpolation for pitch prediction

Typical waveforms and spectra for interpolation by an integer factor L are shown in figure 4.3 below :

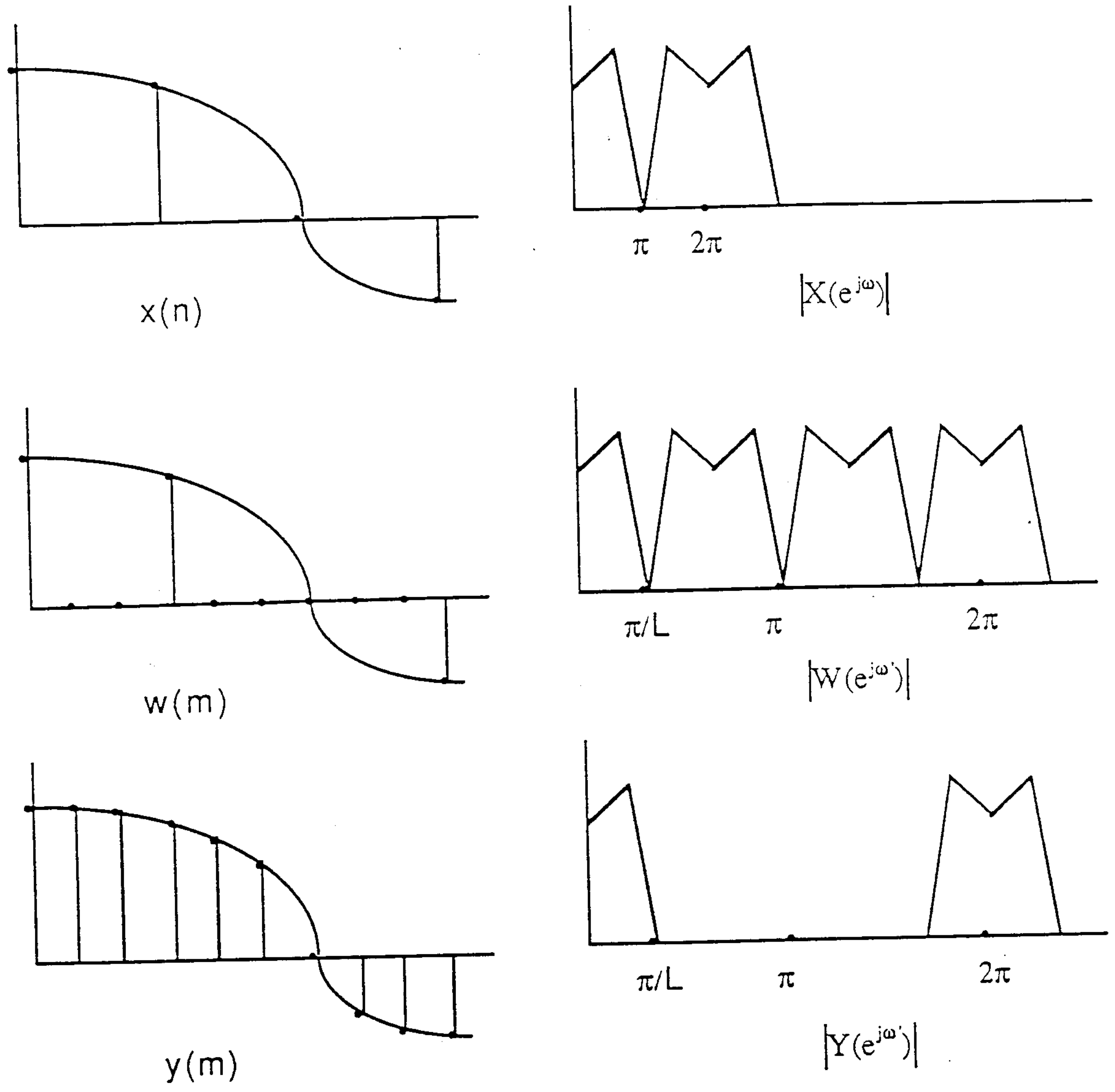


Fig. 4.3 Waveforms and Spectra of interpolated signals

According to the above analysis, to eliminate the images completely, we need an ideal low-pass filter (rectangular spectrum, sinc impulse response). This leads to the well known reconstruction equation. We can recover the continuous-time signal, x(t), from its discrete-time samples, x(n), as follow,

$$x(t) = \sum_{n=-\infty}^{\infty} x(n)\frac{\sin[\pi(t-nT)/T]}{\pi(t-nT)/T} = \sum_{n=-\infty}^{\infty} x(n)\operatorname{sinc}[(t-nT)/T]$$

and then get the interpolated values in between,

$$y(n+d) = x(t)|_{t=(n+d)T}$$

where n is integer, d is fractional delay.

In the above reconstruction equation, the summation is over infinite terms. Because the sinc function diminishes toward infinity, in the actual implementation we only use 7 terms, n = -3 to 3. And we use a 7-tap Hamming window to smooth the spectrum ripple.

Experimental evidence indicates that including fractional delays can reduce the roughness of high-pitched speakers. It can also reduce noise, because increased pitch prediction resolution can reduce the noisy speech residual and therefore improve the similarity between speech residual and codebook excitation vector.

The integer delay is from 20 to 147 (8K / 20 = 400 Hz to 8K / 147 = 54.4 Hz); that is 128 integer delays. We add another 128 fractional delays, *nonuniform*, which are designed to gain the greatest improvement in speech quality by providing highest resolution for typical female speakers and lower resolution for typical male and child speakers.

Actually we can use simple linear interpolation instead of sinc impulse response. Linear interpolation is equal to triangle impulse response, and its spectrum is sinc^2, which means there are ripples outside the baseband; the images are not eliminated completely. Even if we use windowed sinc function, the images are not eliminated completely. To eliminate images completely, we need infinite sinc impulse response, which is impossible. We can only use a window to make the impulse response finite and reduce the ripples outside the baseband. See figure 4.4 below.

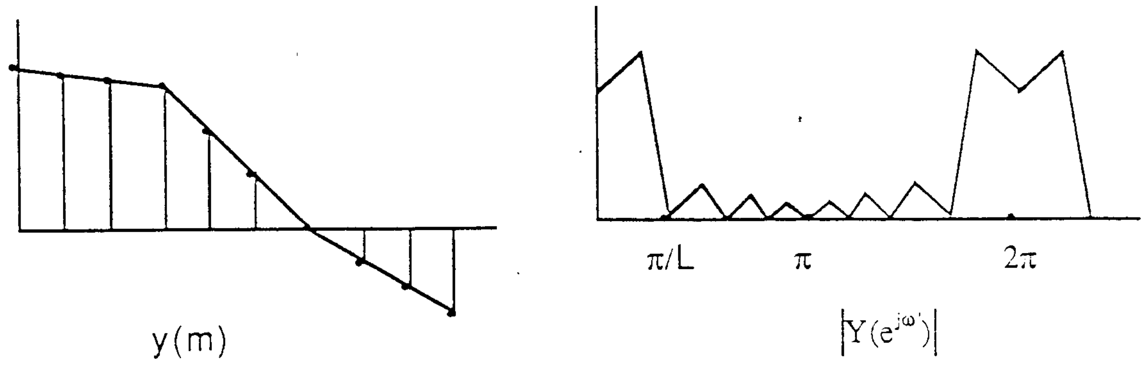


Fig. 4.4 Ripple Effect

We can pre-compute the sinc values. For example, suppose we want to increase the sampling rate three times "internally". In the reconstruction equation :

$$x(t) = \sum_{n=-\infty}^{\infty} x(n)\frac{\sin[\pi(t-nT)/T]}{\pi(t-nT)/T} = \sum_{n=-\infty}^{\infty} x(n)\,\mathrm{sinc}[(t-nT)/T]$$

we need to evaluate x(t) at t = 0, T/3, 2T/3, T, 4T/3, 5T/3, 2T, ... So we only need to pre-compute sinc(1/3), sinc (2/3), sinc(1) = 0, sinc(4/3), ... , take finite number of them, weigh them by the window, and then we can find the necessary sinc values by table lookup. Thus linear interpolation and sinc interpolation have the same computational costs. So we choose sinc interpolation for better results.

4.4 Two Stage Pitch Search to Reduce Computation

In the actual implementation, we do not search the whole 256 (both integer and fractional) delays at once. That is inefficient and unnecessary. We use a two stage search instead. First, we search integer delays only and find the best integer delay. Then we *fine tune* this integer delay by searching its *neighboring fractional delays* (6 neighbors). The computation needed for the second stage fractional delay search is negligible compared to integer delay (128 delays) search.

4.5 Delta Coding for Pitch Index : - to Reduce Both Computation and Bit Rate

Pitch index will not change radically. Especially in steady vowel sound, pitch index will stay around a particularly value for several sub-frames (60 samples). So we do not have to search for the whole delay range (20 to 147) every sub-frame. There are 4 sub-frames in each frame, 0, 1, 2, and 3. For sub-frame 0, we search the whole delay range, find the best delay (need 8 bits to encode); for sub-frame 1, we search the neighboring 64 delays only (need only 6 bits to encode). Same for sub-frame 2 and 3. Using this delta coding scheme, we can save encoding bits and reduce the computation from 4.5 MIPS to about 3 MIPS.

4.6 Perceptual Weighting

Perceptual weighting is very important in CELP coding, we use it in pitch search and codebook search. It is used for frequency domain weighting. *In the spectrum where signal levels are high, the noise is somewhat masked by signal and has a smaller contribution to audible distortion than where signals levels are low.* This suggests we can *weigh the noise according to speech spectrum* to get best perceptual results. Here is the transfer function of the perceptual weighting filter (first we generate impulse response and use convolution instead of using this IIR filter directly) :

$$W(z) = \frac{A(z)}{A(z/\alpha)} = \frac{1 - \sum_{k=1}^{10} a_k z^{-k}}{1 - \sum_{k=1}^{10} a_k \alpha^k z^{-k}}$$

where $0 < \alpha < 1$, and A(z) is the predictor error polynomial.

For $\alpha = 1$, W(z) is an all-pass filter : no weighting. For $\alpha = 0$, W(z) is the inverse of the spectrum, which means the noise is weighted more at a spectrum valley and less at a spectrum peak. For any value between 0 and 1, the weighting filter is between these two extremes. By listening tests, we use $\alpha = 0.8$ [9, 13].

CHAPTER 5 : FINAL SPEECH RESIDUAL VQ

5.1 Complexity of Codebook Search

After short and long term predictions, we have already extracted the spectrum (envelope) information and pitch information. The speech residual is a *noise like sequence*. This residual should not retain much of the original speech information. Can we throw it away? No! Although this residual retains little information, we still need it. The key idea in CELP coding is to use a *noise-like codebook* to encode this residual. We use a 512-size codebook. Of course, the larger the codebook size, the better the result. The speech residual is an approximation to the so called "innovation" sequence associated with the sampled speech data. If y(n) represents the speech samples and F(y, n-1) the information contained in the past samples, before n, the innovations sequence is defined by $w(n) = y(n) - E\{y(n) \mid F(y,n-1)\}$. The extraction of short and long term predictions approximates the term $E\{y(n) \mid F(y, n-1)\}$. It is because of this approximation and the fact that real speech signals are not Gaussian that we still need the residual. In theory, w(n) is a white-noise, Gaussian sequence.

Most of the CELP computational complexity is attributed to codebook search for the residual. As in the following diagram of figure 5.1 :

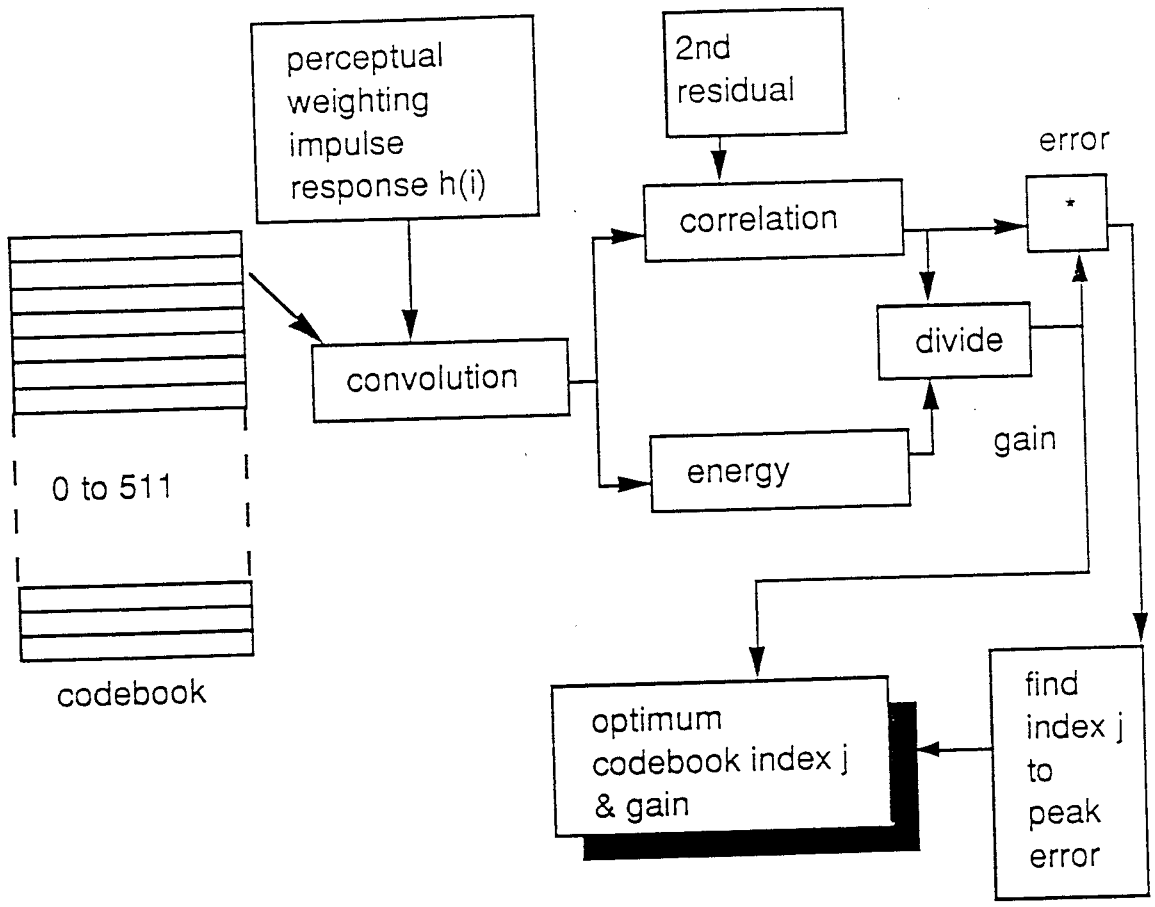

perceptual weighting impulse response h(i)
2nd residual
error
correlation
*
divide
convolution
gain
energy
0 to 511
codebook
optimum codebook index j & gain
find index j to peak error

Fig. 5.1 Codebook Search

In the above diagram, the computation can be attributed to 3 major parts : convolution, correlation (inner product), and energy. Let us assume that the length of the perceptual weighting impulse response h(i) is 10, and estimate the cost of computation.

<u>Convolution :</u>
For each vector, we need 1+2+3+....+10+10+....+10 (60 terms) = 555 MUL, and 1+2+3+...+9+9+....:.+9 (59 terms) = 495 ADD to generate the perceptual weighted vector. There are 512 vectors, that is (555+495)*512 = 537600 operations are needed.

<u>Correlation :</u>
For each correlation, we need 60 MUL and 59 ADD. For 512 correlations, we need (60+59)*512 = 60930 operations.

Energy :
Same as correlation, 60930 operations.

These operations have to be done per 60 samples (7.5 ms), which results in a complexity of (537600+60930+60930) / 7.5 ms = 88 MIPS.

The speed of current DSP chips is about 10 MIPS, 88 MIPS is far beyond this limit. The NSA proposed standard [13] suggests an overlapped codebook, which can reduce the convolution computation by the end point correction technique (just like in the pitch search). This can reduce the total computation to about 8 MIPS. Together with the 3 MIPS needed by the pitch search and other overhead, we still need about 20 MIPS for real time processing.

5.2 Structured Algebraic Codebook in CELP : for Fast Codebook Search

We know that the speech residuals (after short and long term predictions) are Gaussian distributed, so we use stochastic codebooks (generated by a Gaussian process) in CELP speech coding. But as stochastic codebooks are generated randomly, there are no special structures to organize them, so we need to use exhaustive search to find an optimum vector.

Although some people (NSA [13]) propose overlapped codebook to reduce the complexity of convolution (in doing perceptual weighting) by end-point correction, the computational complexity is still very high (8 MIPS). Furthermore the use of overlapped codebook here is an approximation which degrades quality. It is not exact as in the pitch codebook case.

The fundamental question we ask is : Is it necessary to use an un-structured stochastic codebook? Is it possible to construct an organized codebook and by its regular structure to find some efficient ways to search instead of exhaustive search?

To arrive at the answer let us discuss the physical meaning of finding an optimum excitation vector in the codebook :

In CELP, for a given "speech residual" vector, we want to find a vector in the codebook, which, *after scaling*, will produce minimum square error from the speech residual vector. Because of the scaling factor, the criterion is NOT the same as "nearest neighbor" in the Euclidean distance sense. To see this suppose that we have a speech residual vector $\vec{r}$ and a codebook vector $\vec{x}$. The criterion is equivalent to maximizing

$$\frac{|\vec{r}\cdot\vec{x}|^2}{|\vec{x}|^2} = \frac{|\vec{r}|^2|\vec{x}|^2\cos^2\theta}{|\vec{x}|^2} = |\vec{r}|^2\cos^2\theta \quad , \qquad \forall\ \vec{x} \in \text{codeboo:}$$

Because $|\vec{r}|^2$ is fixed in the comparison, we are maximizing $\cos^2\theta$. Referring to the following diagram of figure 5.2 :

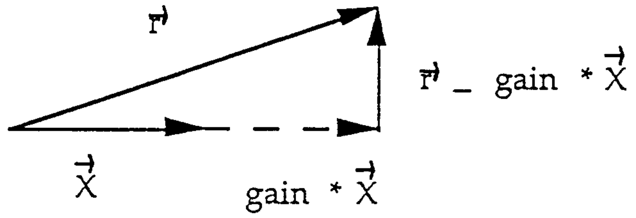


Fig. 5.2 Searching for the optimal codeword

to maximize $\cos^2\theta$ is equivalent to minimize $\sin^2\theta$, thus minimizing the difference between these two vectors, $|\vec{r} - \text{gain} * \vec{x}|$. Therefore the criterion is to maximize $\cos^2\theta$, since scaling can do the rest. To simplify the problem, suppose we want to maximize $\cos\theta$. To maximize $\cos\theta$ means to find a codebook vector which is most *parallel* to the speech residual.

By the above discussion, we know the criterion for a "good" codebook : it must *span the n-dimensional sphere as uniformly as possible [1]*. For a fixed number of vectors, they will have the best *direction representation ability* if they are uniformly distributed over the n-dimensional space. We now see very clearly that it is NOT necessary to use an un-structured stochastic codebook; rather, we can construct a codebook which can span the n-dimensional sphere "more uniformly" than a randomly generated stochastic codebook. This means we can even construct a codebook which is actually better than a stochastic codebook. We call such a codebook "algebraic codebook". Such codebooks have been proposed for CELP coding in earlier works [1, 2]. Our codebook is substantially different however.

The reason people use randomly generated stochastic codebooks is that when we plot the histogram of the speech residuals, we can see that they are approximately Gaussian distributed. So people use an i.i.d. Gaussian process to generate the codebook, and fortunately, the result comes out not bad. When we construct our algebraic codebook, we must take this "Gaussian distribution" property into consideration. We shall see later that this step is also necessary in order to bring down the codebook size from terribly large to somewhat manageable.

We know that a good codebook needs to span the n-dimensional sphere uniformly. To simplify things, we restrict the elements of our codebook vectors to be ternary, i.e. -1, 0, and 1. Note that all the simplifications made here can always be justified by experimental results; and if this is not considered satisfactory, we may argue that in the search process stated above, the "direction" of a vector is used as the matching criterion rather than its "exact location" : so the ternary restriction should be able to retain the "directional" representativity of each vector.

In the NSA CELP standard [13], n = 60. This means that even with the ternary restriction, there are 3^{60} - 1 (zero vector) possible vectors in the 60-dimensional space. That is of course too large! To achieve 4.8 kbps encoding (the original speech is 12 bits/sample, sampling rate 8K, so it is 96 kbps), we only have 9 bits reserved for the codebook index, which means that our codebook size can only be 2^9. To bring down the size, first we consider the Gaussian distribution property of speech residuals. We know that most of the residuals are fairly small, so we can let a fair amount of our vector's components be zero and thus reduce the size of the codebook. But how many? There is no way to derive the cutoff threshold theoretically. Based on experiments, the NSA team uses a 77% zero codebook and have reported a fairly good performance. So we follow this idea (make it 80% zero, because 77% of 60 is not an integer) and let our vectors have 48 zeros out of the total 60 components, and the remaining 12 are 1's or -1's. After these simplifications, the size becomes

$$S_w^n = 2^w \times \binom{n}{w} = \frac{2^w \times n!}{(n-w)! \times w!} = \frac{2^{12} \times 60!}{48! \times 12!}$$

where n is dimension, w is weight. As we can see, this size is still too large, much bigger than the desired 2^9.

How can we further reduce the size of the codebook?

We know that the residuals are time sequences, we also know that human ears are insensitive to phase shifts in the speech waveform [16]. So, the positions of those 12 1's and -1's may not be that important. Why not choose 12 fixed positions to put those non-zero spikes, and only play with the signs of them! After doing so, the size is reduced to 2^{12}, quite close to the ideal 2^9. We still need some more restrictions to reduce the codebook size!

By now, it seems we have run out of intelligent ways to come up with other restrictions. So, let us make a guess : put some restrictions about all possible combinations of these 1's and -1's.

Now we have two decisions to make : to choose 12 non-zero positions out of the total 60, and set some restrictions on the combinations of those 1's and -1's.

First let us choose 12 non-zero positions out of the 60 vector length. Intuitively, it seems reasonable to place them uniformly over the 60 positions : only elements with index 5n are non-zero , i.e. X0000X0000X0000....... , each X can be either 1 or -1. This configuration has a wonderful property when we consider the calculation of "perceptual weighting". We shall see this a little later.

Now we have a 60-dimensional vector which has 12 spikes uniformly distributed. It looks like this :

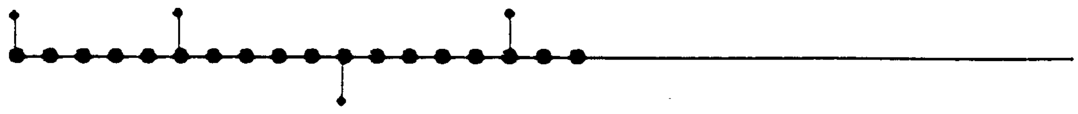


Fig. 5.3 A typical codeword

The spikes can flip up and down to form the 2^9 vectors needed. We may see this vector this way : since we want to use it to represent the *noise like* speech residual, by flipping up and down those spikes we will have the best *shaping abili:* at least better than randomly generated vectors.

Now we have 2^{12} vectors in our codebook. We further need to put some restrictions on the combinations of 1's and -1's. Let us first partition the vector into 3 equal length sub-vectors. The length of each sub-vector is 20 and there are 4 non-zero elements in each of them. We pose the restriction that we allow only even numbers of -1's out of these 4 non-zero elements. So, now we can have 4 1's (1 combination), 4 -1's (1 combination), and 2 1's & 2 -1's (6 combinations) for each sub-vector. Each sub-vector has 8 combinations, that means each vector has $8^3 = 2^9$ combinations. Finally, we get a codebook size of 2^9, that takes 9 bits for the codebook index ; exactly what we need!

It is important to note that because this is an algebraic codebook, we *don't even need to store the codebook*, <u>the codebook index, alone, already specifies each vector exactly.</u>

The arguments above are not easy to provide a theoretical justification. The problem is that we are trying to span a 60-dimensional sphere with only 2^9 vectors; and this number is definitely not enough to do a good job. We've already taken the Gaussian distribution into consideration, which is the only thing a randomly generated stochastic codebook is meant for. Besides that, we have just tried to make these vectors spread more uniformly. The good behavior of our codebook is heuristically suggested by extensive experiments [17], indicating that the "size" of the excitation signal sphere is of secondary importance compared with the long term predictor.

5.3 Fast Algorithm to Compute Inner Products

To calculate the 2^9 inner products of the speech residual vector with respect to each of our codebook vectors, we note that there are only 1's and -1's in the codebook vectors, so there is actually no need for multiplications. We only need to pick the right components in the speech residual vector, and then add or subtract them. We will come back to perceptual weighting later.

Using combination of different terms, we can calculate all the 2^9 inner products with an extremely small amount of operations. Here is how we do it :

Starting from the length-20 sub-vector, since in our codebook vectors, only elements with index 5n are non-zero, and they are all 1's and -1's, we only need elements with index 5n of the speech residual vector to calculate all the inner products. For each of the 3 sub-vectors, we calculate 8 sums corresponding to 8 combinations of codebook sub-vectors :

	r0	r5	r10	r15
0.	+	+	+	+
1.	+	+	-	-
2.	+	-	+	-
3.	+	-	-	+
4.	-	+	+	-
5.	-	+	-	+
6.	-	-	+	+
7.	-	-	-	-

Fig. 5.4 Codebook sub-vector combinations for inner product computations.

Now, for each sub-vector we have 8 sums, pick one from each of them and we have 3 sums, add these 3 sums and we have one inner product. There are 8^3 ways to pick up 3 sums from 3 sub-vectors, and that is exactly the 2^9 inner products we need! This is illustrated in figure 5.5 below.

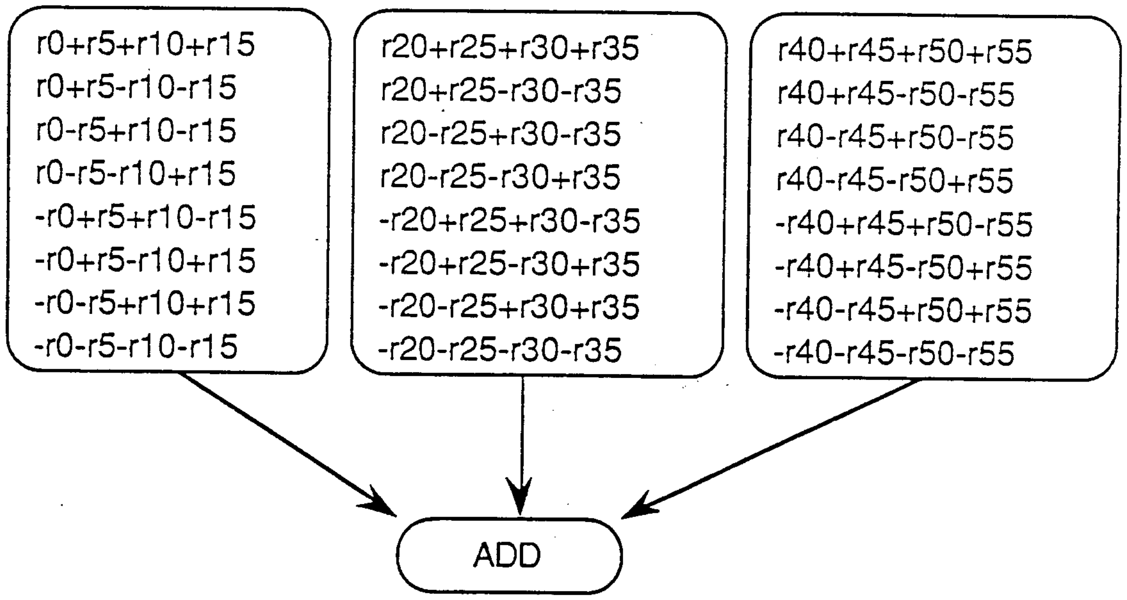


Fig. 5.5 Computation of inner products.

As shown we just pick one from each of the above columns and add them to get the 2^9 inner products we need!

Now we consider perceptual weighting. We use impulse response $h_0, h_1, h_2, \ldots\ldots\ldots$ to characterize the filter, i.e. FIR filter. That means we need to do convolutions of the impulse response h with each of our codebook vectors. Since all of our codebook vectors have 4 zeros between two non-zero elements, if we cut the impulse response length to 5, keep only h_0, h_1, h_2, h_3, and h_4, (because the lower order coefficients are more important than higher order ones, this should not cause too much distortion). We can now keep the above tree structure, as follows.

The codebook vector after perceptual weighting should look like :

$$\underbrace{h_0h_1h_2h_3h_4}_{+\text{ or }-}\underbrace{h_0h_1h_2h_3h_4}_{+\text{ or }-}\underbrace{h_0h_1h_2h_3h_4}_{+\text{ or }-}\cdots\cdots\cdots\text{(60 } h_i\text{'s)}$$

with some sign changes, but each group of (h_0 h_1 h_2 h_3 h_4) should be of the same sign.

Keeping the same structure as in figure 5.5, we can replace the above

r0 with r0*h0+r1*h1+r2*h2+r3*h3+r4*h4
r5 with r5*h0+r6*h1+r7*h2+r8*h3+r9*h4
....
r55 with r55*h0+r56*h1+r57*h2+r58*h3+r59*h4

We see that we can still get all 2^9 inner products using an extremely small amount of operations.

We also need to calculate the energy for each vector after perceptual weighting, and fortunately, as the vectors after perceptual weighting look like :

$$\underbrace{h_0 h_1 h_2 h_3 h_4}_{+\text{ or }-}\ \underbrace{h_0 h_1 h_2 h_3 h_4}_{+\text{ or }-}\ \underbrace{h_0 h_1 h_2 h_3 h_4}_{+\text{ or }-}\ \cdots\cdots (60\ \ h_i\text{'s})$$

and their energies are all the same :

$$12 * \sum_{i=0}^{4} h_i^2$$

The spirit of this algorithm is : *Because all the codebook vectors are just different combinations of signs, so the "components" in all inner products are the same : it is not necessary to re-compute these components, just play with the combinations of signs and we can get all the inner products.*

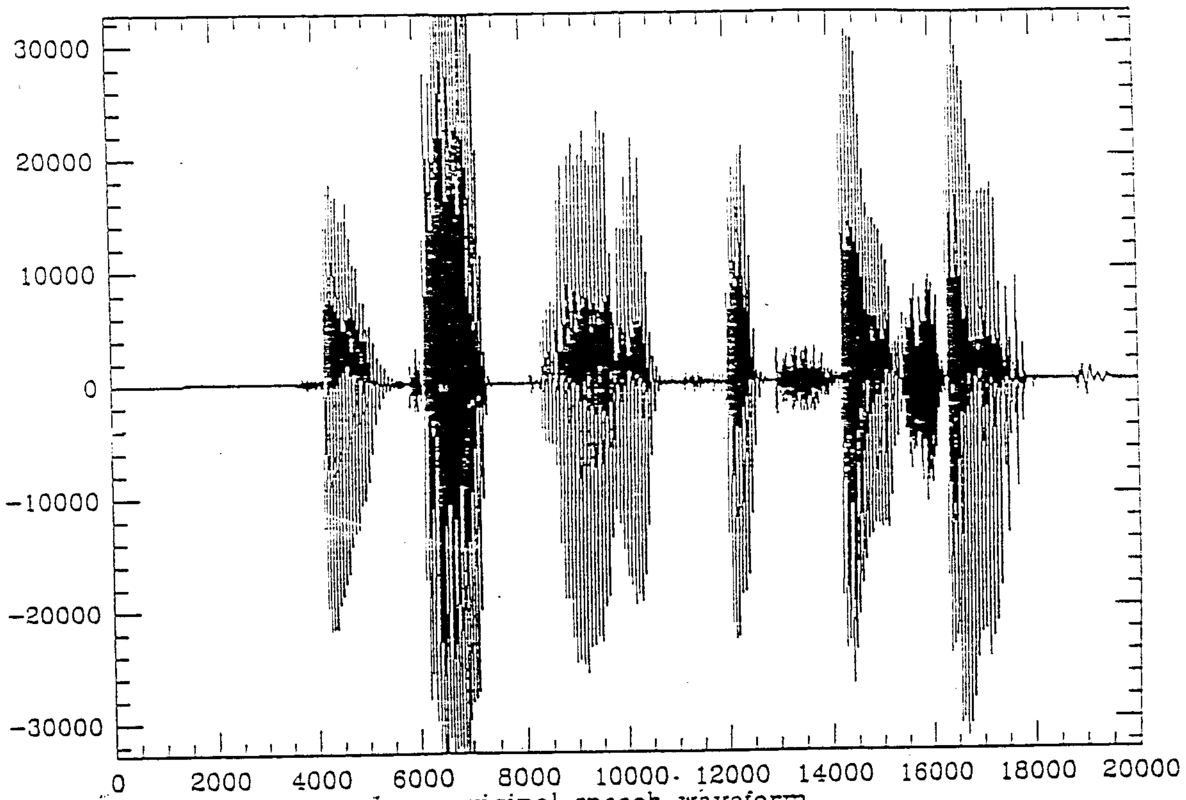
To compute the exact number of operations needed, first, we need 5*12 = 60 MUL and 4*12 = 48 ADD to compute the above r0, r5, ..., r55; and then need 4*8*3 = 96 ADD or SUB to compute 8(combinations)*3(sub-vectors) terms; finally we need 2*512 = 1024 ADD to compute the 512 inner products. **That's only 1228 operations to get 512 inner products!** And only 60 of them are MUL, others are ADD or SUB. 1228 / 7.5 ms = **0.16 MIPS**. Compared with the brute force search requiring 80 MIPS (512 * 60 codebook), this represents an **improvement of 500.** Compared with overlapped codebook's 8 MIPS, this represents an **improvement of 50.** Originally the codebook search dominated the complexity of CELP analysis, now the computations needed for codebook search is negligible compared to pitch search.

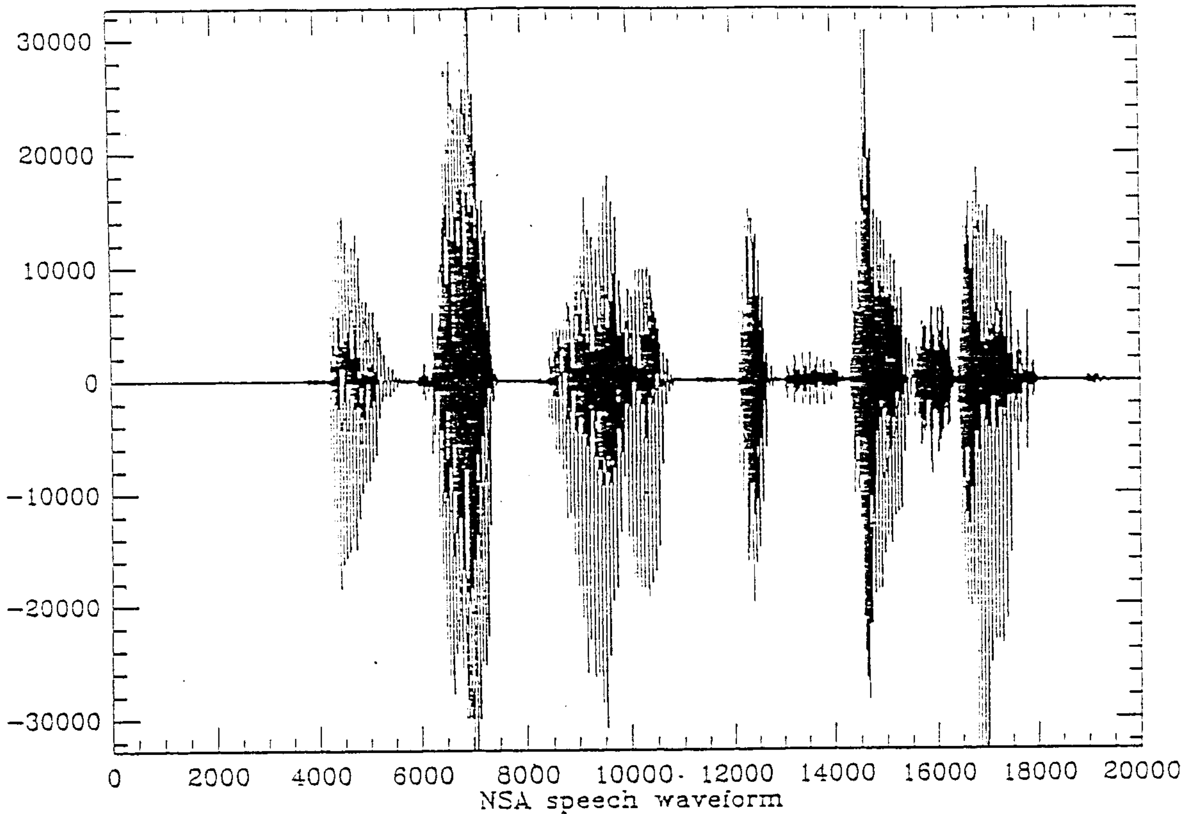
Actually the codebook doesn't have to be so restrictive (as stated above) to benefit from the above algorithm. As long as the non-zero positions in all codebook vectors are fixed, and their absolute values are the same, which means *the only difference among all vectors is different sign combination*, then we can use the above algorithm.

CHAPTER 6 : QUALITY RESULTS AND SUGGESTIONS FOR FUTURE RESEARCH

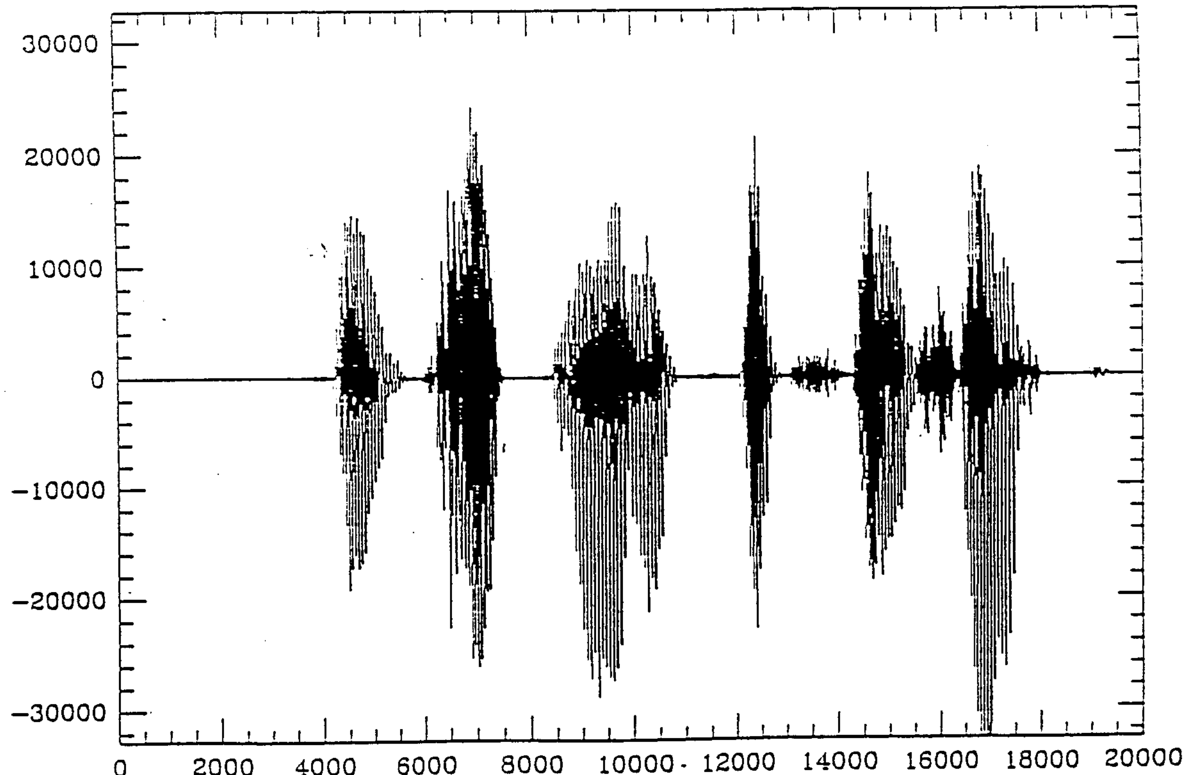
We have tested the proposed codebook, and compared the result with the standard NSA CELP code. The resulting speech qualities are basically indistinguishable. As stated earlier there are no good objective measures for speech quality. We calculate the S/N ratio for comparison. For the structured codebook, SNR = 6.5, v.s. SNR = 7.5 for NSA overlapped codebook. (This SNR is averaged over 6 sentences, 3 male and 3 female speakers) And the speed is about tripled, because the codebook search computation (2/3 of all computations) is nearly totally removed. Using this codebook, the CELP algorithm needs less than 5 MIPS, can easily fit into any single one DSP chip for real time processing.

Future directions include further reduction of complexity by focusing on pitch prediction, actual implementation in various DSP chips and the analysis and performance evaluation in noisy environments (telephone lines, offices, etc).

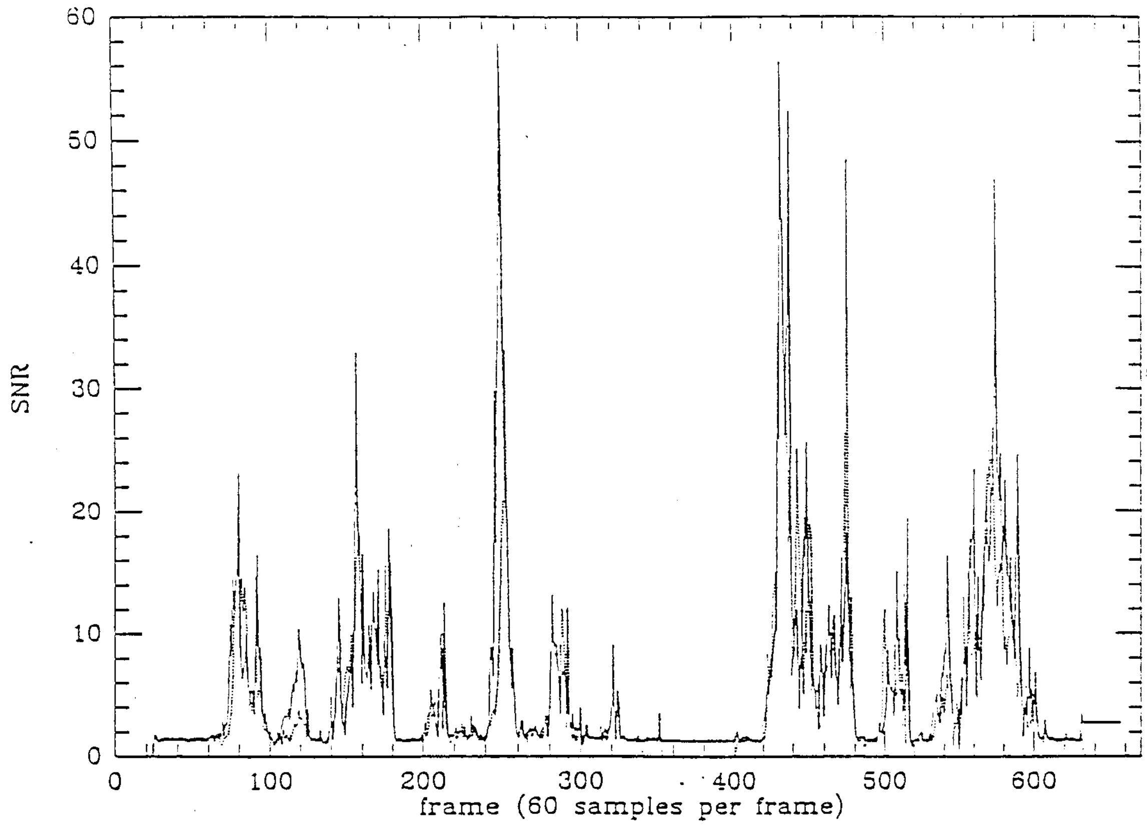

30000
20000
10000
0
-10000
-20000
-30000
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
original speech waveform



NSA speech waveform



algebraic codebook reconstructed speech waveform



Solid line : NSA
Dashed line : Algebraic codebook

References

1. M. A. Ireton and C. S. Xydeas, "On improving vector excitation coders through the use of spherical lattice codebooks (SLC's)" ICASSP 1989, 57 - 60.
2. C. Lamblin, J. P. Adoul, and S. Morissette, "Fast CELP coding based on the Barnes-Wall lattice in 16 dimensions" ICASSP 1989, 61 - 64.
3. Peter Kabal and Ravi Prakash Ramachandran, "The Computation of Line Spectral Frequencies Using Chebyshev Polynomials" IEEE Transaction on ASSP Vol. ASSP-34, NO. 6, Dec.1986, 1419 - 1426.
4. Frank K. Soong and Biing-Hwang Juang, "Line Spectrum Pair (LSP) and Speech Data Compression" ICASSP 1984, 1.10.1 - 1.10.4.
5. Noboru Sugamura and Fumitada Itakura, "Speech Analysis and Synthesis Methods Developed at ECL in NTT - from LPC to LSP-" Speech Communication 5 (1986) North-Holland, 199 - 215.
6. Augustine H. Gray, JR and John D. Markel, "Distance Measures for Speech Processing" IEEE Transaction on ASSP Vol. ASSP-24, NO. 5, Oct. 1976, 380 - 390.
7. David P. Kemp, Retha A. Sueda, and Thomas E. Tremain, "An Evaluation of 4800 bps Voice Coders" U.S. Government, DoD.
8. Bishnu S. Atal and Manfred R. Schroeder, "Code Excited Linear Prediction (CELP) : High quality speech at very low bit rates" ICASSP 1985.
9. Thomas E. Tremain, Joseph P. Campbell, JR, and Vanoy C. Welch, "A 4.8 K bps Code Excited Linear Predictive Coder" U.S. DoD.
10. Bishnu S. Atal, "predictive Coding of Speech at Low Bit Rates" IEEE Transaction on Communications Vol. COM-30, NO. 4, Apr. 1982, 600 - 614.
11. I. M. Trancoso and B. S. Atal, "Efficient Procedures for Finding the Optimum Innovation in Stochastic Coders" ICASSP Apr. 8-11 1986.

12. B. S. Atal, "High Quality Speech at Low Bit Rates : Multi-Pulse and Stochastically Excited Linear Predictive Coders" ICASSP 1986.
13. Joseph P. Campbell, JR, Vanoy C. Welch, and Thomas E. Tremain, "An Expandable Error Protected 4800 bps CELP Coder (U.S. Federal Standard 4800 bps Voice Coder)" , U.S. DoD.
14. Manfred R. Schroeder and B. S. Atal, "Stochastic Coding of Speech Signals at Very Low Bit Rates : The Importance of Speech Perception" , Speech Communication 4, North-Holland.
15. Ronald E. Crochiere and Lawrence R. Rabiner, "Multirate Digital Signal Processing" by Prentice-Hall.
16. L. R. Rabiner and R. W. Schafer, "Digital Processing of Speech Signals" by Prentice-Hall.
17. R. C. Rose and T. P. Barnwell, III, "Design and Performance of an Analysis-by-Synthesis class of predictive speech coders" IEEE Trans. on ASSP, Vol. ASSP-38, NO. 9, Sept. 1990, 1489 - 1503.