Optimized Implementation of Speech Processing Algorithms

Sara Grassi

THÈSE SOUMISE À LA FACULTÉ DES SCIENCES
DE L'UNIVERSITÉ DE NEUCHÂTEL POUR L'OBTENTION
DU GRADE DE DOCTEUR ÈS SCIENCES

IMPRIMATUR POUR LA THÈSE

Optimized Implementation of Speech Processing Algorithms

de Mme Sara Grassi

UNIVERSITÉ DE NEUCHÂTEL
FACULTÉ DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel sur le rapport des membres du jury,

MM. F. Pellandini (directeur de thèse),
H. Hügli, M. Ansorge et M. Kunt (EPF Lausanne)

autorise l'impression de la présente thèse.

Neuchâtel, le 20 février 1998

Le doyen:

[illegible]

F. Stoeckli

Blank page

Abstract

Several speech processing applications such as digital hearing aids and personal communications devices are characterized by very tight requirements in power consumption, size, and voltage supply. These requirements are difficult to fulfill, given the complexity and number of functions to be implemented, together with the real time requirement and large dynamic range of the input signals. To meet these constraints, careful optimization should be done at all levels, ranging from algorithmic level, through system and circuit architecture, to layout and design of the cell library. The key points of this optimization are among others, the choice of the algorithms, the modification of the algorithms to reduce computational complexity, the choice of a fixed-point arithmetic unit, the minimization of the number of bits required at every node of the algorithm, and a careful match between algorithms and architecture.

The optimization method is explained and then applied to two typical speech processing applications: noise reduction/speech enhancement for digital hearing aids and spectral analysis and quantization in the CELP FS1016 speech coder.

Blank page

Résumé

Les exigences relatives à la consommation d'énergie, la taille et l'alimentation sont très sévères pour un certain nombre d'applications du traitement de la parole, par exemple les aides auditives digitales ou les appareils de communication portables. Ces conditions sont difficiles à remplir, étant donné la complexité et le grand nombre de fonctions à implanter, auxquels s'ajoutent les contraintes liées au temps réel et à la large dynamique des signaux d'entrée. Pour satisfaire ces exigences, une optimisation soignée doit être menée à tous les niveaux, depuis l'algorithme, jusqu'au circuit et à la conception de la librairie de cellules, en passant par l'architecture du système et du circuit. Les aspects majeurs de l'optimisation concernent notamment le choix des algorithmes, les modifications nécessaires pour réduire le coût de calcul, le choix d'une unité arithmétique à virgule fixe, la minimisation du nombre de bits nécessaires pour chaque valeur dans l'algorithme ainsi que l'adéquation minutieuse entre algorithmes et architecture.

La méthode d'optimisation est détaillée puis illustrée dans le cas de deux applications types du traitement de la parole : la réduction de bruit pour les aides auditives digitales ainsi que l'analyse spectrale et la quantification du codeur CELP FS1016.

Blank page

Acknowledgements

This work was made possible by the contribution of many people.

First, I would like to thank my supervisor Professor Fausto Pellandini for putting his trust in me, giving me the opportunity to work in his research group, and for supervising the writing of this thesis. I also would like to thank Professor Murat Kunt, Professor Heinz Hügli and Mr. Michael Ansorge for co-examining this thesis.

This work was partially financed by the Swiss Foundation for Research in Microtechnology (FSRM), under the project FSRM 91/07 and the Federal Commission for the Encouragement of Scientific Research (CERS), under the project CERS 2747.1.

Mr. Michael Ansorge also participated directly in the research described in this report. He contributed to the organization of my work, bringing useful ideas and carefully examining and proof-reading our publications, contributing to their improvement with his comments and corrections. Mr. Alain Dufaux also worked with me during part of this research and we had a lot of interesting discussions.

I am also very grateful to the following students, who did their undergraduate thesis or their practical semester work with me: Joerg Troger, Stéphane Kuenzi, Olivier Huesser, Beat Rupp, François Seguin and Giuseppina Biundo. I have learned a lot from them.

My husband, Andreas Pauletti, was most helpful during the writing of this report. Besides his love, continuous moral support and encouragement, he proof-read (more than once!) the whole manuscript and drew several of the figures.

Vincent Moser kindly provided the document style definitions from his own Ph.D. report, as well as a lot of typographical advice.

Catherine Marselli did the French translation, and supported me endlessly before and during the writing of this thesis.

My colleagues were always available and helpful during my time at IMT. In particular our secretary Catherine Lehnherr and our System Manager Heinz Burri.

Some of my colleagues were particularly friendly and supportive. Among them Vincent Moser, Alexis Boegli, Dominique Daudet, Dequn Sun, Christian Robert and Javier Bracamonte.

Finally, I would like to thank all my family and friends, who make my life worth living.

Table of Contents

Chapter 1
Introduction

The research presented in this Ph.D. report addresses the optimized implementation of some functional blocks which are found frequently in digital speech processing applications.

1.1. Motivation

The principal means of human communication is speech. This fact is reflected in modern technology, as machines are used to transmit, store, manipulate, recognize, and create speech, as well as for recognizing the identity of the speaker. For these tasks, the speech signal is usually represented digitally.

The development of VLSI and DSP chips has paved the way for the implementation of highly complex digital speech processing algorithms. As a result, speech processing technology is now being used in telecommunications and business, for applications like voice mail, personal communications systems, automated operators, information retrieval systems, and voice activated security.

On the other hand, some applications of digital speech processing, such as personal communications systems and hearing aids, require the use of portable, battery operated devices. Their implementation is thus characterized by tight constraints in power consumption and size. For high volume applications, low cost is also a priority.

The choice of a fixed-point arithmetic is a key point to decrease cost, size and power consumption in ASIC implementations. Furthermore, commercial fixed-point DSP chips are cheaper and have a smaller power consumption than floating-point DSPs. Therefore, the analysis of fixed-point quantization effects is of great importance in carefully optimized implementations.

Optimization at the algorithmic level (algorithm choice and simplification) is the basis for a low power implementation as it allows savings of orders of magnitude in power consumption. Another aspect is the determination of the optimum scaling and minimum wordlength needed at every node of the algorithm.

In order to reduce the number of iterations in the design phase, it is desirable to be able to predict some aspects of the performance of the hardware before actually implementing it. In Chapter 3, a practical method for evaluating the effects resulting from the use of fixed-point arithmetic is presented, as part of a methodology aimed to optimize the implementation of speech processing algorithms for low power applications.

This methodology was applied to the implementation of a noise reduction algorithm for digital hearing aids, and to the implementation of spectral analysis and quantization for speech coding.

1.2. Scope of the Research

In the research presented in this report, only digital speech processing algorithms were considered. In particular, the study was restricted to two areas of speech processing: speech enhancement with application to digital hearing aids, and speech coding with application to portable communications devices. Both applications are characterized by very tight constraints in cost, power consumption and size.

As the choice of a fixed-point arithmetic is a key point to decrease cost, size and power consumption, in both programmable DSP and ASIC implementations, only fixed-point implementations were considered. This implies a higher development effort, as the designer has to determine the

dynamic range and precision needs of the algorithms before implementation, either analytically, or through simulation. The practical and simple method for evaluating fixed-point quantization effects on DSP algorithms, presented in Chapter 3, aims to help the designer in this task. The proposed method allows a simulation of the system in final working conditions and at the same time benefit of the flexibility of using a high level language, independently of the hardware.

Of all the possible optimization strategies at different implementation levels, only optimization at the algorithmic level allows power consumption savings of orders of magnitude. Thus, in the research described in this report, the optimization effort is restricted to algorithmic optimization. Algorithmic optimization comprises the following strategies:

(1) Choice of the algorithms.

(2) Simplification of the algorithms in order to reduce the complexity and decrease the dynamic range needs.

(3) Study of the fixed-point quantization effects, to determine the optimum scaling and minimum wordlength required at every node of the algorithm.

(4) Simplification of the interactions among the different algorithms inside the whole system.

(5) Good interrelation between the algorithms and the target architecture.

These optimization strategies were used in the implementation of a noise reduction algorithm for digital hearing aids on a fixed-point commercial DSP and on a low power VLSI architecture, as described in Chapter 4. They were also used in the implementation of the spectral analysis block of the CELP FS1016 speech coder, as described in Chapter 7.

1.3. Organization of the Report

In Chapter 2, a brief introduction to the field of digital speech processing and its applications is given. The purpose of this chapter is to give some of the basic definitions and to show the importance of optimization in speech applications.

An optimization methodology, which is based on algorithmic optimization and the study of fixed-point quantization effects, is proposed in Chapter 3. This methodology was used in the implementation of a noise reduction algorithm for digital hearing aids, as explained in Chapter 4.

The theoretical fundamentals for understanding the LSP representation of LPC coefficients, with application to speech coding, are given in Chapter 5. The CELP FS1016 speech coder, in particular its spectral analysis block, is also explained. These concepts are used in Chapter 6, in which two novel efficient algorithms for LPC to LSP conversion are presented. In Chapter 7, the DSP56001 optimized implementation of the CELP FS1016 spectral analysis block is given.

Finally, the general conclusions are given in Chapter 8.

1.4. Main Contributions

The main contributions of the Ph.D. work described in this report are:

(1) The optimization methodology for speech processing algorithms presented in Chapter 3, together with a simple and practical method for evaluating the behavior of digital signal processing algorithms in the case of 2's complement fixed-point implementations.

(2) Two novel efficient algorithms for LSP calculation from LPC coefficients, named Mixed-LSP and "quantized-search Kabal", presented in Chapter 6.

(3) The unified comparison among three existing LSP calculation algorithms, and the two proposed methods, given in Chapter 6. This comparison is done using the same conditions (same speech database and target speech coder).

1.5. Publications

Part of the work described in this report has already been the subject of some publications. The paper presented at the Seventh European Signal Processing Conference in Scotland, in

October 94 [Gras94], describes the methodology of optimization, simulation of quantization effects, and its application to a noise reduction/speech enhancement algorithm for digital hearing aids. The optimization methodology and the application to the noise reduction algorithm are explained with more details in Chapter 3 and 4 of this report. A companion paper, presented by A. Heubi at the same conference [Heub94], describes the low power architecture used for the VLSI implementation (see § 4.7).

The paper presented at the IEEE International Conference on Acoustics, Speech, and Signal Processing in Munich, in April 97 [Gras97a], describes the new efficient method for LPC to LSP conversion, called Mixed-LSP, which is explained in Chapter 6.

Two internal IMT reports, covering some parts of Chapter 3 and 4 [Gras95], and Chapter 5, 6 and 7 [Gras97b] were also written. In particular, the listings for the C, Matlab, and DSP56000 assembly programs, used in the work described in this thesis are given in these two reports.

1.6. References

[Gras94] S. Grassi, A. Heubi, M. Ansorge, and F. Pellandini, “Study of a VLSI Implementation of a Noise Reduction Algorithm for Digital Hearing Aids”, *Proc. EUSIPCO'94*, Vol.3, pp. 1661-1664, 1994.

[Gras95] S. Grassi, *Simulation of Fixed-point Quantization Effects on DSP Algorithms*, IMT Report No 375 PE 03/95, University of Neuchâtel, IMT, 1995.

[Gras97a] S. Grassi, A. Dufaux, M. Ansorge, and F. Pellandini, "Efficient Algorithm to Compute LSP Parameters from 10-th order LPC Coefficients", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol. 3, pp. 1707-1710, 1997.

[Gras97b] S. Grassi, *DSP56001 Implementation of the Spectral Analysis and Quantization for the CELP FS1016 Speech Coder*, IMT Report No 421 PE 10/97, University of Neuchâtel, IMT, 1997.

[Heub94] A. Heubi, S. Grassi, M. Ansorge, and F. Pellandini, "A Low Power VLSI Architecture for Digital Signal Processing with an Application to Adaptive Algorithms for Digital Hearing Aids", *Proc. EUSIPCO'94*, Vol. 3, pp. 1875-1878, 1994.

Chapter 2
Digital Speech Processing

In this chapter a brief introduction to the field of digital speech processing and its applications is given. The purpose is to mention some concepts and give some definitions that are used throughout this report, and to show the importance of the optimization of speech processing functional blocks for some particular applications.

Theoretical fundamentals which are more specific to the work done are given at the beginning of Chapter 3 and 4, and in Chapter 5.

2.1. The Speech Signal

To communicate information to a listener a speaker produces a speech signal in the form of pressure waves that travel from the speaker’s head to the listener’s ears [Osha87]. These pressure waves are converted to an analog electrical speech signal through the use of transducers (e.g., microphones). This speech signal contains frequencies ranging from 100 Hz up to perhaps 8 kHz, and has amplitudes between 30 to 90 dB.

To digitally process speech signals which are in analog form, they are converted into a digital form (i.e., a sequence of numbers). This is done in two steps. The signal is first periodically sampled, obtaining a discrete-time, continuous-amplitude signal which is then quantized in amplitude.

The rate at which the analog signal is sampled is known as the *sampling frequency*, F_s. The Nyquist theorem requires that F_s be greater than twice the bandwidth of the signal to avoid aliasing distortion. Thus the analog signal is low-pass filtered before sampling. As the low-pass filter is not ideal, the sampling frequency is chosen to be higher than twice the bandwidth. In telecommunication networks, the analog speech signal is band-limited to 300-3400 Hz and sampled at 8 kHz. Hereafter, the term *speech coding* (see § 2.9) will refer to the coding of this type of signal. For higher quality, speech is band-limited to 0-7000 Hz and sampled at 16 kHz. The resulting signal is referred to as *wideband speech*.

The sampled signal is quantized in amplitude via an analog-to-digital converter, which represents each real sample by a number selected from a finite set of L possible amplitudes (where $B = \log_2 L$ is the number of bits used to digitally code the values). This quantization process adds a distortion called *quantization noise*, which is inversely proportional to L. In practice 12 bits are needed to guarantee an SNR higher than 35 dB over typical speech ranges [Osha87].

2.2. Model of Speech Production

Speech production can be viewed as a filtering operation, in which a sound source excites a vocal tract filter. The source may be either periodic, resulting in *voiced speech*, or noisy and aperiodic, causing *unvoiced speech*. There are also some parts of speech which are neither voiced nor unvoiced but a mixture of the two, called the transition regions. Amplitude versus time plots of typical voiced and unvoiced speech are shown in Figure 2.1.

In this speech production model the effects of the excitation source and the vocal tract are considered independently. While the source and tract interact acoustically, their independence causes only secondary effects.

The voicing source occurs at the larynx at the base of the vocal tract, where the airflow from the lungs is interrupted periodically by the vocal folds generating periodic puffs of air.

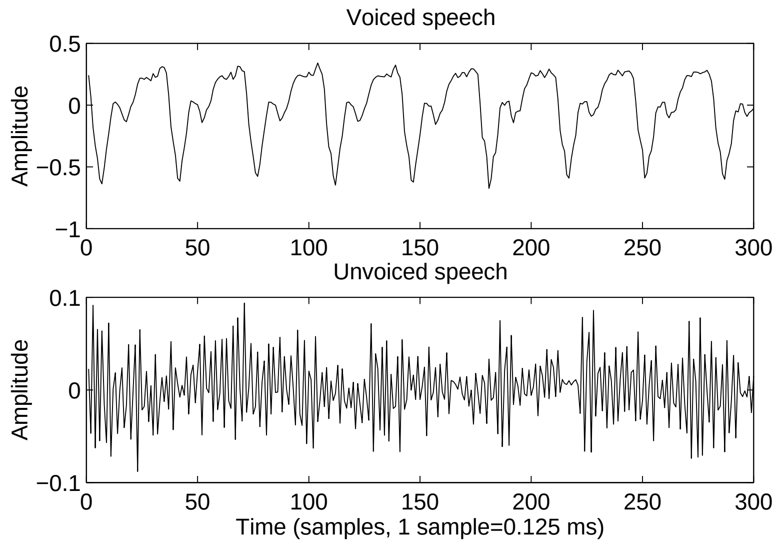


Figure 2.1: Typical voiced and unvoiced speech waveforms.

The rate of this excitation is the fundamental frequency F_0, also known as *pitch*. Voiced speech has thus a spectra consisting of harmonics of F_0. Typical speech has an F_0 of 80-160 Hz for males. Average F_0 values for males and females are respectively 132 Hz and 223 Hz.

Unvoiced speech is noisy due to the random nature of the excitation signal generated at a narrow constriction in the vocal tract.

The vocal tract is the most important component in the speech production process. For both, voiced and unvoiced excitation, the vocal tract acts as a filter, amplifying certain sound frequencies while attenuating others. The vocal tract can be modeled as an acoustic tube with resonances, called *formants*, and antiresonances (or spectral valleys). These formants are denoted as F_i, where F_1 is the formant with the lowest center frequency). The formants correspond to poles of the vocal tract frequency response, whereas some spectral nulls are due to the zeros. Moving the articulations of the vocal tract alters the shape of the acoustic tube, changing its frequency response.

Thus, the produced speech signal is non-stationary (time-varying) changing characteristics as the muscles of the vocal tract contract and relax. Whether or not the speech signal is voiced, its characteristics (e.g., spectral amplitudes) are often relatively fixed or quasi-stationary over short periods of time (10-30 ms), but the signal changes substantially over intervals greater than the duration of a given sound (typically 80 ms).

2.3. Frequency-domain Analysis of the Speech Signal

Most useful parameters in speech processing are found in the spectral domain. The speech signal is more consistently and easily analyzed spectrally than in the time domain and the common model of speech production (see § 2.2) corresponds well to separate spectral models for the excitation and the vocal tract. The hearing mechanism appears to pay much more attention to spectral amplitude of speech than to phase or timing aspects (see § 2.6). For these reasons, spectral analysis is used primarily to extract relevant parameters of the speech signal. One form of spectral analysis is the short-time Fourier transform, which is defined, for the signal s(n), as:

$$S_k(e^{j\omega}) = \sum_{n=-\infty}^{\infty} w(k-n)\cdot s(n)\cdot e^{-j\omega n} \tag{2.1}$$

Due to the non-stationary nature of speech, the signal is windowed, by multiplication with $w(k-n)$, to isolate a quasi-stationary portion for spectral analysis.

The choice of duration and shape of the window w(n), as well as the degree of overlap between successive windows, reflects a compromise in time and frequency resolution. Tapered cosine windows such as the Hamming window are typically used, and the length of the window is usually 10 to 30 ms for speech sampled at 8 kHz.

In Equation (2.1), the variable ω is the angular frequency, which is related to the real frequency Ω (in Hz) by the equation:

$$\omega = 2\pi\Omega/F_s \tag{2.2}$$

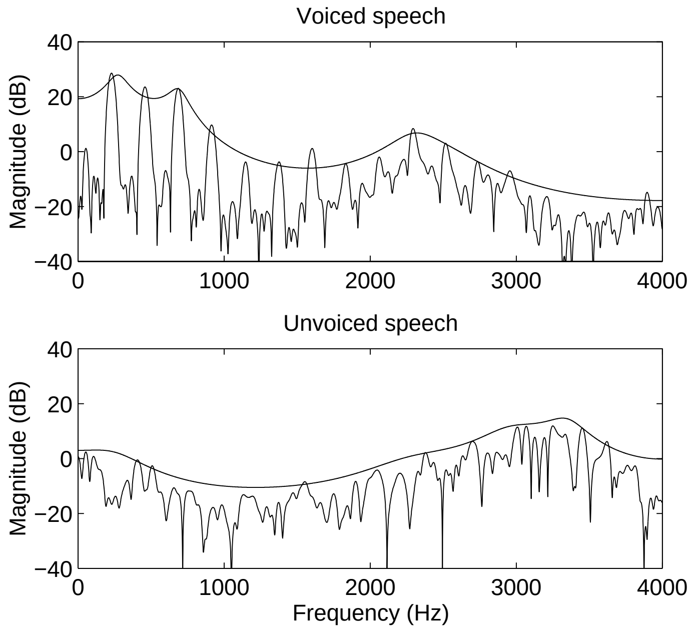


Figure 2.2: Spectra of the voiced and unvoiced speech waveforms shown in Figure 2.1, and 10-th order LPC envelope.

Another variable which is sometimes used is the normalized frequency f, related to ω and Ω by:

$$f = \Omega/F_s\,, \quad f = \omega/2\pi \tag{2.3}$$

As the spectrum of a digital signal is periodic in ω, the useful range for the frequency, corresponding to one period of the spectrum is given by: $0 \le \omega \le 2\pi$, $0 \le f \le 1$, and $0 \le \Omega \le F_s$. Furthermore, as the speech signal is real, the spectrum is symmetric and the interesting frequency range is: $0 \le \omega \le \pi$, $0 \le f \le 0.5$, and $0 \le \Omega \le F_s/2$.

The short-time power spectra of the voiced and unvoiced speech waveforms of Figure 2.1, as well as their 10-th order LPC envelope (see § 2.4) are shown in Figure 2.2.

The discrete Fourier transform (DFT) is used for computation of Equation (2.1) so that the frequency variable ω takes N discrete values (N corresponding to the window duration). Since the Fourier transform is invertible, no information is lost in this representation. A more economical representation of speech parameters is achieved by the use of linear predictive analysis.

2.4. Linear Predictive Modeling of the Speech Signal

Spectral magnitude is a relevant aspect of speech which is widely used in speech processing. One source of spectral magnitude is the short-time Fourier transform. Alternatively, linear predictive coding (LPC) provides an accurate and economical representation of the envelope of the short-time power spectrum of speech.

In LPC analysis, the short-term correlation between speech samples (formants) is modeled and removed. This technique is based on the model of speech production explained in Section 2.2. A simplified block diagram of this model is shown in Figure 2.3.

In this model, the excitation signal, e(n), is either an impulse train (for voiced speech) or white noise (for unvoiced speech). The combined spectral contributions of the glottal flow, the vocal tract, and the radiation of the lips are represented by a time varying digital filter. This filter is called the *LPC synthesis filter*. Its transfer function has both poles and zeros, but to minimize analysis complexity, the filter is assumed to be all-pole, with a transfer function given by:

$$H_p(z) = \frac{S(z)}{E(z)} = \frac{1}{1+\sum_{k=1}^{p} a_p(k)\cdot z^{-k}} = \frac{1}{A_p(z)} \qquad (2.4)$$

where $\{a_p(1),\ldots,a_p(p)\}$ are the LPC coefficients and p is the order of the filter (or LPC order). An order of 10 is typically used for telephone bandwidth (300-3400 Hz) speech sampled at 8 kHz. Using this LPC order, formant resonances and general spectral shape (envelope) are modeled accurately. The 10-th order LPC spectra for the voiced and unvoiced speech waveforms of Figure 2.1 (superposed to its corresponding short-time power spectra), is shown in Figure 2.2.

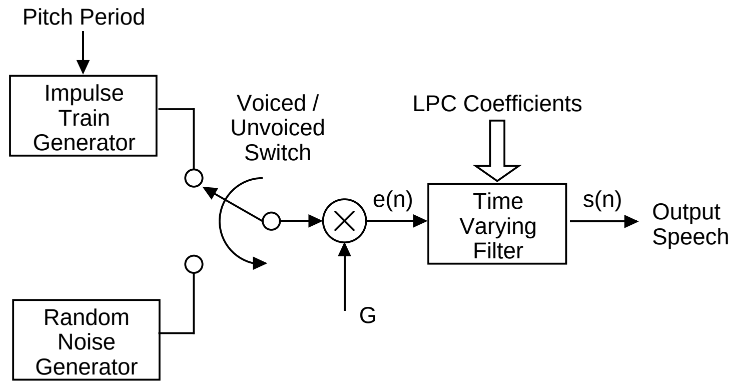


Figure 2.3: Block diagram of the simplified source filter model of speech production.

The *LPC analysis filter* is given by:

$$A_p(z) = 1+\sum_{k=1}^{p} a_p(k)\cdot z^{-k} \qquad (2.5)$$

Transforming Equation (2.4) to the time domain results in:

$$e(n) = s(n)+\sum_{k=1}^{p} a_p(k)\cdot s(n-k) = s(n)-\hat{s}(n) \qquad (2.6)$$

It is seen that the current speech sample s(n) is predicted by a linear combination of p past samples, $\hat{s}(n)$. Thus the signal e(n) is the prediction *error* or *residual* signal. Hence, the *p*-th order LPC analysis problem is stated as follows: given measurements of the signal s(n), determine the parameters $\{a_p(1),\ldots,a_p(p)\}$ so as to minimize the error signal e(n).

2.5. Calculation of the LPC Coefficients

Using the least-squares method, the LPC coefficients are determined by minimizing the mean energy of the error signal, given by:

$$\varepsilon_p = \sum_{-\infty}^{\infty} e^2(n) = \sum_{-\infty}^{\infty}\left[s(n)+\sum_{k=1}^{p} a_p(k)\cdot s(n-k)\right]^2 \qquad (2.7)$$

The summation range is limited by windowing either the speech or the error signal, leading to the autocorrelation or covariance method, respectively. The autocorrelation method is computationally more efficient than the covariance method and the resulting synthesis filter is always stable. In the autocorrelation method, the LPC coefficients are calculated by using the efficient Levinson-Durbin recursion (see § 5.2). This method is very popular in speech coders such as the CELP FS1016 (see § 5.11).

An alternative representation of the LPC coefficients (see § 5.4), which corresponds to the multipliers of a *lattice filter* realization of the LPC synthesis filter, are the Parcor (partial correlation) or reflection coefficients, $\{k_1,\ldots,k_p\}$. The LPC coefficients can be transformed to reflection coefficients and vice versa, using the recursions given in Equations (5.18) and (5.19). There are some LPC calculation methods, which give directly the reflection coefficients, without calculating the LPC coefficients. Two of these methods are Burg's method [Kond94] and LeRoux-Gueguen method (see § B.1).

Instantaneous (sample by sample) adaptation of the reflection coefficients is obtained by using a gradient least-mean-square (LMS) adaptive algorithm [Widr85]. In this case, the LPC calculation algorithm is called the gradient adaptive lattice (GAL) predictor. This LPC calculation algorithm is used in the noise reduction/speech enhancement algorithm for digital hearing aids described in Chapter 4.

2.6. Hearing and Speech Perception

The speech signal entering the listener's ear is converted into a linguistic message [Osha87]. The ear is especially responsive to those frequencies in the speech signal that contain the most information relevant to communication (i.e., frequencies approximately in the 200-5600 Hz range). The listener is able to discriminate small differences in time and frequency found in speech sounds within this frequency range.

Key perceptual aspects of a speech signal are more evident when represented spectrally than in the time domain. Spectral amplitude is much more important than phase for speech perception and whether a sound can be heard depends on its spectral amplitude. The minimum intensity at which a sound

can be heard is known as the *hearing threshold*, which rises sharply with decreasing frequency below 1 kHz and with increasing frequency above 5 kHz. An upper limit is given by the intensity at which a sound causes discomfort or pain, known as the *threshold of pain*. The range between the thresholds of hearing and pain is known as the *auditory field*. Speech normally occupies only a portion of the auditory field.

Formant frequencies and amplitudes (see § 2.2) play an important role in speech perception. Vowels are distinguished primarily by the location of their three formant frequencies, while formant transitions provide acoustic cues to the perception of consonants. Formant bandwidths are poorly discriminated and their changes appear to affect perception primarily through their effects on formant amplitudes. The valleys between formants are less perceptually important than formant peaks and humans have relatively poor perceptual resolution for spectral nulls.

2.7. Speech Processing and DSP Systems

A digital signal processing (DSP) system is an electronic system applying mathematical operations to digitally represented signals such as digitized speech [Laps97].

DSP enjoys several advantages over analog signal processing. The most significant is that DSP systems are able to accomplish tasks which would be very difficult, or even impossible with analog electronics. Besides, DSP systems have other advantages over analog systems such as flexibility and programmability, greater precision, and insensitivity to component tolerances. Analog signal processing requires specific equipment, rewiring, and calibration for each new application, while digital techniques may be implemented, tested and easily modified on general purpose computers.

These advantages, coupled with the rapidly increasing density of digital IC manufacturing processes make DSP the solution of choice for speech processing.

2.8. Digital Speech Processing Areas and Applications

In the previous sections, some aspects of speech signals that are important in the communication process were described.

Some areas of speech processing, such as speech coding, encryption, synthesis, recognition and enhancement, as well as speaker verification, utilize the properties of the speech signal to accomplish their goals [Rabi94], [Lim83]. In Table 2.1, typical system applications of these speech processing areas are given [Laps97]. It is seen that several of these applications are characterized by tight constraints in power consumption and size. Among them we can mention: hearing aids, digital cellular telephones, vocal pagers and portable multimedia terminals with speech i/o [Chan95].

Digital speech processing area	***System applications***
Speech coding and decoding	Digital cellular telephones, digital cordless telephones, vocal pager, multimedia computers and terminals, secure communications.
Speech encryption and decryption	Digital cellular telephones, digital cordless telephones, multimedia computers and terminals, secure communications.
Speech recognition	Advanced user interfaces, multimedia computers and terminals, robotics, automotive, digital cellular telephones, digital cordless telephones.
Speech synthesis	Multimedia computers, advanced user interfaces, robotics.
Speaker verification	Security, multimedia computers, advanced user interfaces.
Speech enhancement (e.g., noise reduction, echo cancellation, equalization)	Hearing aids, hands-free telephone, telephone switches, automotive, digital cellular telephones, industrial applications.

Table 2.1: Typical system applications of different speech processing areas.

2.9. Speech Coding

Speech coding is the process of compressing the information in a speech signal either for economical storage or for transmission over a channel whose bandwidth is significantly smaller than that of the uncompressed signal.

The ideal coder has low bit rate, high perceived quality, low signal delay, low complexity and high robustness to transmission errors. In practice, a trade-off among these factors is done, depending on the requirements of the application.

The term *speech coding* (or narrowband speech coding) refers to the coding of telephone bandwidth (300-3400 Hz) speech sampled at 8 kHz, whereas the term *wideband speech coding* refers to the coding of speech band-limited to 0-7000 Hz and sampled at 16 kHz.

The speech research community has given different names to different qualities of speech found in a telecommunication network [Osha87]:

(1) *Toll* quality describes speech as heard over the switched telephone network. The frequency range is 300-3400 Hz, with signal-to-noise ratio of more than 30 dB and less than 2-3 % of harmonic distortion.

(2) *Communication*s quality speech is highly intelligible but has noticeable distortion compared with toll quality.

(3) *Synthetic* quality speech is 80-90 % intelligible but has substantial degradation, sounding “buzzy” or “machinelike” and suffering from lack of speaker identifiability.

Some nuances in this characterization are found in speech research, where sometimes a coder is described as having “near toll quality”, or “good communications quality”.

The bit rate of a coder is expressed in bits per seconds (bps) or kilobits per second (kbps) and is given by:

$$T_c(\text{kbps}) = B\ (\text{No. of bits}) \cdot F_s(\text{kHz}) \tag{2.8}$$

Toll quality corresponds to (300-3400 Hz) band-limited speech sampled at 8 kHz and represented with 12 bits (uniform quantization). The bit rate is thus 96 kbps. Using μ-law or A-law logarithmic compression, the number of bits is reduced to 8, and thus the bit rate to 64 kbps. This logarithmic coding was standardized as the ITU-T G.711 and is used as a reference for toll quality in speech coding research.

In communications systems such as satellite communications, digital mobile radio, and private networks, the bandwidth and power available are severely restricted, hence reduction of the bit rate is vital. This is done at the expense of decreased quality and higher complexity.

Toll quality is found in coders ranging from 64 kbps to 10 kbps, near toll and good communications quality is found in the range of 10 to 2.4 kbps, and communications to synthetic quality below 4.0 kbps.

Vector Quantization

Vector quantization (VQ) is the process of quantizing a set of k values jointly as a single vector. If the vector elements are correlated, the number of bits to represent them is reduced with respect to scalar quantization.

The block diagram of a simple vector quantizer is shown in Figure 2.4. The *codebook* $\mathbf{Y}$ contains a number L of *codevectors* $\mathbf{y}_i$ of dimension N: $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{iN}]^T$. The subindex i is the address or *index* of the codevector $\mathbf{y}_i$. Each codevector is uniquely represented by its index. The length of the codebook L, and the number of bits of the index B are related by: $B = \log_2 L$.

The N dimensional input vector $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$ is vector quantized by first finding its “closest” vector in the codebook, and then representing $\mathbf{x}$ by the index of this closest vector. The closest vector is the one that minimizes some *distortion measure*. Typical distortion measures are the mean squared error and the weighted mean squared error. The codebook design process is known as *training* or populating the codebook. One popular method for codebook design is the k-means algorithm [Kond94].

The number of codevectors L, should be large enough that for each possible input vector, substitution by its closest codevector does not introduce excessive error. However, L must be limited to limit the computational complexity of the search and because the bit rate is proportional to $B = \log_2 L$.

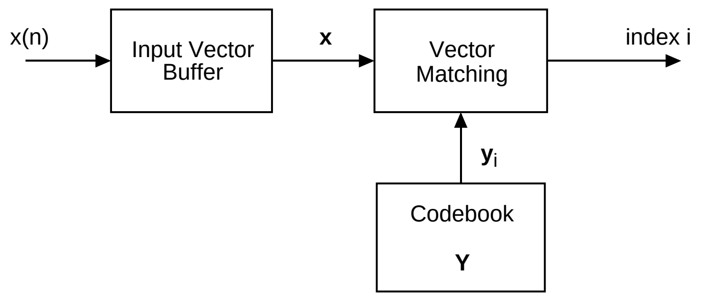


Figure 2.4: Block diagram of a simple vector quantizer.

The main drawback of vector quantization is its high computational and storage cost. Compared to scalar quantization, the major additional complexity of VQ lies in the codebook search. In a *full codebook search*, the input vector is compared with each of the L vectors of the codebook, requiring L computationally expensive distance calculations.

The codebook size is also a problem for codebook training. As an example, if a 20-bit representation is needed, the codebook should contain 2^{20} codevectors of dimension N. This would require a prohibitively large amount of training data, and the training process would need too much time. Besides, as the codebook is stored at both the receiver and the transmitter, the storage requirement would be prohibitively high.

Practical VQ systems use suboptimal search techniques that reduce search time and sometimes codebook memory while sacrificing performance. Among these techniques there are tree-searched VQ, multistage VQ, classified VQ and split VQ [Gers94].

In CELP coders, VQ is used for quantization of the excitation signal, and sometimes also to model the long term correlation of the speech signal (pitch) by means of an adaptive codebook search.

Additionally, VQ is successfully used to quantize spectral parameters (i.e., any representation of the LPC coefficients) as explained in Section 5.8.

CELP coding

Most notable and most popular for speech coding is code excited linear prediction (CELP). These coders had a great impact in the field of speech coding and had found their way in several regional and international standards. While newer coding techniques have been developed, none clearly outperforms CELP in the range of bit rates from 4 to 16 kbps [CELP97]. The obtained quality ranges from toll to good communications quality. Furthermore, several reduced complexity methods for CELP were studied in speech coding research. As a result, more than one full-duplex CELP coder can nowadays be implemented on a state-of-the-art DSP processor.

Current research goes in the direction of reducing complexity and enhancing performance. Another current trend is the use of speech classification, notably voice activity detection (VAD) and voice/non voice classification for bit rate reduction. The obtained coders are variable bit rate coders, with an average bit rate lower than 3 kbps and the same quality of fixed rate coders at 4.8 kbps.

CELP coding refers to a family of speech coding algorithms which combine LPC-based analysis-by-synthesis (AbS-LPC) and vector quantization (VQ) [Gers94]. The general diagram of a CELP coder is shown in Figure 2.5.

In AbS-LPC systems, the LPC model is used (see § 2.4), in which an excitation signal, e(n), is input to a synthesis filter, $H_p(z)$, to yield the synthetic speech output s(n).

There are two synthesis filters. The LPC synthesis filter models the short-term correlation between speech samples (formants) whereas the pitch synthesis filter models the long-term correlation (pitch).

The coefficients of the LPC synthesis filter are determined from a frame of the speech signal, using an open-loop technique such as the autocorrelation method (see § 2.5). The coefficients of the pitch synthesis filter are also determined by open loop techniques [Kond94].

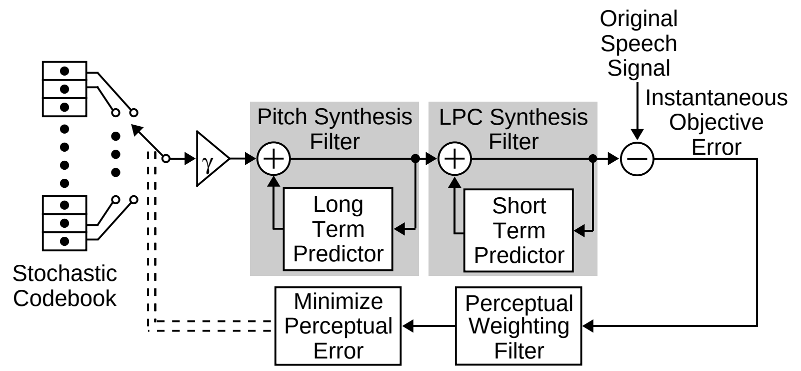


Figure 2.5: Block diagram of a general CELP coder.

Once the parameters of the LPC and pitch synthesis filters are determined, an appropriate excitation signal is found by a closed-loop search. The input of the synthesis filters is varied systematically, to find the excitation signal that produces the synthesized output that best matches the speech signal, from a perceptual point of view.

Vector quantization (VQ) is combined with AbS-LPC in CELP coders [Gers94]. The optimum excitation signal is selected from a stochastic codebook of possible excitation signals (codevectors). Each codevector is passed through the LPC and pitch synthesis filters. The codevector which produces the output that best matches the speech signal is selected.

In some CELP coders, such as the FS1016 (see § 5.11) the pitch synthesis filter is substituted by a search on an adaptive codebook, which models long term correlation.

Parametric Coders

Fixed rate CELP coders do not perform well with bit rates below 4 kbps. Using parametric coders [LOWB97], good communications and near toll quality is obtained at 2.4 kbps. These speech coders are based on algorithmic approaches such as sinusoidal coders, in particular sinusoidal transform coding (STC) and multiband excitation (MBE). Another widely used approach is prototype waveform interpolation (PWI), which is a technique to efficiently model voiced excitation. Combining parametric coders with frame classification schemes, variable bit rate coders with average bit rate of 1.3 kbps are obtained. The main disadvantage of parametric coders is their high complexity, and lower quality when compared with CELP coders.

2.10. Speech Enhancement

Speech enhancement involves processing speech signals for human listening or as preparation for further processing before listening [Lim83]. The main objective of speech enhancement is to improve one or more perceptual aspects of speech, such as overall quality or intelligibility.

Speech enhancement is desirable in a variety of contexts. For example, in environments in which interfering background noise (e.g., office, streets and motor vehicles) results in degradation of quality and intelligibility of speech. Other applications of speech enhancement include correcting for room reverberation, correcting for the distortion of speech due to pathological difficulties of the speaker, postfiltering to improve quality of speech coders, and improvement of normal undegraded speech for hearing impaired people.

An example of speech enhancement is the algorithm described in Chapter 4, which was studied and optimized for implementation. In this algorithm, spectral sharpening is used for both noise reduction and to compensate for the loss in frequency selectivity encountered among hearing impaired people.

Digital Hearing Aids

Analog electroacoustic hearing aids are the primary treatment for most people with a moderate-to-severe sensorineural hearing impairment [Work91]. These conventional hearing aids contain the basic functions of amplification, frequency shaping, and limiting of the speech signal. The conventional hearing aids provide different amounts of amplification at different

frequencies so as to fit as much of the speech signal as possible in the reduced auditory field (see § 2.6) of the hearing impaired person.

Digital hearing aids promise many advantages over conventional analog hearing aids. The first advantage is the increased precision and programmability in the realization of the basic functions. The frequency response can be tailored to the needs of the patient and also change according to different acoustic situations. Another advantage is the possibility of adding new functions such as noise reduction, spectral sharpening and feedback cancellation, which are impossible or very difficult using analog circuits.

Furthermore, external computers can be used to simulate and study new algorithms to be included in the hearing aid and for new and improved methods of prescriptive fitting and evaluation.

On the other hand, the physical implementation of digital hearing aids is characterized by very tight requirements [Lunn91]:

(1) Size: the small physical dimensions of analog hearing aids contribute to the acceptance by the user. The smallest devices (in-the-channel hearing aids) have just some cm^3 to accommodate microphone, receiver, signal processing chip and power supply.

(2) Power supply: for keeping a small dimension, only one 1.5 battery cell should be used.

(3) Power consumption: typical values of 1-2 mW, to allow a battery life of several weeks.

These requirements are very difficult to fulfill given the complexity and number of functions to be implemented, the real time requirement and the large dynamic range of the input signals. Therefore, the physical implementation of digital hearing aids can only be achieved by a careful optimization that ranges from algorithm level, through system and circuit architecture to layout and design of the cell library.

In Chapter 4, the optimization of the implementation of a noise reduction/speech enhancement algorithm for digital hearing aids is presented.

The sampling frequency for digital hearing aids is a controversial issue. In [Lunn91] an F_s of 12 kHz is used, whereas in several algorithms proposed in literature, an F_s of 8 kHz is used. Higher sampling rates may be unnecessary due to the reduced auditory field of the hearing impaired person.

2.11. Speech Processing Functional Blocks

Some functional blocks which are typically used in the different speech processing areas, and which were optimized for implementation in the work described in this report, are explained as follows.

Lattice FIR, IIR and GAL Predictor

Lattice filters and lattice linear predictors are used in many areas of digital speech processing such as coding, synthesis and recognition, as well as in the implementation of adaptive filters [Proa89], [Osha87]. The lattice structure offers significant advantages over the transversal filter realization. The lattice filter performance using finite word-length implementation is much superior to that exhibited by the direct implementation. Also, the lattice adaptive linear predictor presents faster convergence than the direct form when the stochastic gradient algorithm (LMS) is used [Honi88]. In commercial speech synthesis chips the lattice filter is prevalently used because of its guaranteed stability and suitability for fixed-point arithmetic [Osha87], [Wigg78], [Iked84]. Furthermore, lattice filter structures are particularly suitable for VLSI implementation due to their modular structure, local interconnections, and rhythmic data flow [Kail85].

The noise reduction/speech enhancement algorithm described and optimized in Chapter 4 is based on lattice filter structures (GAL LPC predictor, and modified FIR and IIR lattice filters). These functional blocks find also application in other speech processing systems (see § 4.8). The GAL predictor is used in backward predictive speech coders and other systems where instantaneous update of spectral information is needed.

The modified FIR and IIR filters studied in Chapter 4 are the basis for the postfiltering algorithm found in several speech coders and vocoders to improve quality of the synthesized speech. These modified FIR and IIR filters are also used in CELP coders for perceptual weighting of the error between the original and synthesized speech. Finally the IIR lattice filter is ideal for the implementation of the LPC synthesis filter found in most speech coding and synthesis systems.

LPC Calculation

LPC provides an accurate and economic representation of the speech spectral envelope (see § 2.4). This representation is used in speech coding to model and remove short-term correlation of the input signal. The LPC coefficients are used in the synthesis filter found in speech synthesis systems. Due to its representation of perceptually important speech parameters it is also used in speech recognition and speaker verification systems.

An interesting aspect of LPC analysis is that it is not just applied to speech processing, but also to a wide range of other fields such as control and radar [Osha87].

Two types of LPC calculation algorithms were optimized for implementation in the work described in this report. One is the LPC calculation on a frame-by-frame basis using the autocorrelation method and the Levinson-Durbin recursion. This algorithm was optimized for implementation on a fixed-point commercial DSP as part of the DSP56001 implementation of the CELP FS1016 spectral analysis and quantization described in Chapter 7. The second is the sample-by-sample calculation of the reflection coefficients done with the GAL predictor, which was optimized for both implementation on a DSP56001 and on a low power VLSI architecture, as described in Chapter 4.

LSP Representation of LPC Parameters

Line spectrum pair (LSP) parameters are a one to one representation of the LPC coefficients. This representation allows more efficient encoding (quantization) of spectral information, and is very popular in low bit rate coding (see § 5.6, 5.7 and 5.8).

LSP parameters are not only used to encode speech spectral information more efficiently than using other representations, but also provide good performance in speech recognition [Pali88], and speaker recognition [Liu90].

On the other hand, the calculation of LSP parameters from LPC coefficients is a computationally intensive task, as it involves the resolution of polynomials by numerical root search.

In Chapter 5, a survey of existing algorithms for LSP calculation is given (see § 5.9). Three algorithms which are found promising for efficient real time implementation are retained for further study and comparison.

In Chapter 6, two new efficient algorithms for LSP calculation are presented, and then compared with existing algorithms from the point of view of accuracy, reliability and computational complexity. The efficient implementation of these algorithms on a DSP56001 is given in Chapter 7.

Efficient implementation of LSP to LPC conversion is also addressed in Chapter 5, 6, and 7.

2.12. Implementation Issues

The goal of speech coding is reducing bit rate, without degrading speech quality, whereas hearing aids are aimed to improve speech intelligibility and perceived quality. However, in the implementation of these algorithms, other factors apart from the their functionality are of importance. Some of these factors are discussed as follows.

Real-time Constraints

A real-time process is a task which needs to be performed within a specified time limit. Most speech processing systems must meet rigorous speed goals, since they operate on segments of real-word signals in real-time.

While some systems (like databases) are required to meet performance goals *on average*, speech processing algorithms must meet goals at defined instants of time. In such systems, failure to maintain the necessary processing rates is considered a serious malfunction. These systems are said to be subject to *hard real-time constraints*.

In digital speech processing, the processing needs to be performed within 125 µs for sample-by-sample processes (with F_s = 8 kHz). The allowed time is higher for processes performed on a frame-by-frame basis, such as LPC calculation with autocorrelation method (see § 2.5), with typical block lengths of 20-30 ms, and subframe lengths of 5-10 ms.

Processing Delay

In some speech processing applications, such as digital hearing aids and telecommunications, the total delay has to be kept within specified limits. The processing time usually adds to other components of the total delay (e.g., algorithmic delay and transmission delay). Thus, in some cases the processing speed has to be increased beyond the speed required for real-time operation, to keep up with the delay requirement.

Programmable DSP versus Custom Hardware

The designer needs to decide whether to use a programmable DSP chip or to build custom hardware. These two options are discussed next.

Programmable DSP Implementation

Programmable digital signal processors (often called DSPs) are microprocessors that are specialized to support the repetitive, numerically intensive tasks found in DSP processing [Laps97].

Dozen of families of DSPs are available on the market today. The first task in selecting a DSP processor is to weight the relative importance of performance, cost, integration, ease and cost of development, and size and power consumption for the desired application.

Algorithmic optimization is very important from the cost point of view. Any speech processing algorithm can be implemented using commercially available DSP processors, but the cost will increase rapidly with the number of DSP chips used. Another important consideration is the power consumption of the final product, especially if it is a portable, battery operated device.

A key issue is the choice of a fixed-point or floating-point device. Floating-point DSPs are costlier and have a higher power consumption than fixed-point DSPs. Floating-point operations require more complex circuitry and larger word sizes (which imply wider buses and memory) increasing chip cost. Also, the wider off-chip buses and memories required increase the overall system cost and power consumption.

On the other hand, floating-point DSPs are easier to program, as, usually, the programmer does not have to be concerned by dynamic range and precision considerations.

Most high volume applications use fixed-point processors because the priority is low cost. For applications that are less cost sensitive, or that have extremely demanding dynamic range and precision requirements, or were ease of programming is important, floating-point processors are the choice.

Note also that the implementation on a commercial fixed-point DSP can be seen as an intermediate step before the actual implementation using custom hardware (see § 4.4 and 4.6). This implementation allows real time evaluation, optimization of the scheduling, and helps in the study and optimization of the fixed-point behavior.

Custom Hardware and ASIC

There are two important reasons why custom-developed hardware is sometimes a better choice than a commercial DSP implementation: performance and production cost.

In virtually any application, custom hardware can be designed which provides a better performance than a programmable DSP. Furthermore, in some applications such as digital hearing aids, the tight constraints in size and power can only be met by using custom hardware.

For high volume products, custom hardware is less expensive than a DSP processor. Due to its specialized nature, custom hardware has the potential to be more cost effective than commercial DSP chips. This is because a custom implementation places in the hardware only those functions needed by the application, whereas a DSP processor requires every application to pay for the full functionality of the processor, even if it uses only a small subset of its capabilities.

Custom hardware can take many forms, such as printed circuit boards using off-the shelf components, but this form is falling out of favor as the performance of DSP processors increases. In case a very high performance is needed, or very low power and size are required, the solution is the use of application specific integrated circuits (ASIC).

Designing a custom chip provides the ultimate flexibility, since the chip can be tailored to the needs of the application. On the other hand, the development cost is high, and the development time can be long.

A key point for an optimized ASIC DSP implementation is the choice of a fixed-point arithmetic, and minimization of the number of bits needed for the representation of constants and variables (see § 3.1).

2.13. Fixed-point versus Floating-point Arithmetic

The choice of a fixed-point arithmetic is a key point to decrease cost, size, and power consumption in both programmable DSP and ASIC implementations. As in speech processing applications such as hearing aids and portable communications devices, minimization of cost, size, and power consumption is essential, a fixed-point arithmetic is chosen. This implies a higher development effort. The designer has to determine the dynamic range and precision needs of the algorithms before implementation, either analytically, or through simulation.

2.14. Algorithmic Optimization

Algorithmic optimization is essential to obtain a low power ASIC implementation. This is seen in Table 2.2, where the expected power saving at different implementation levels is given [Raba97]. An explanation of all the possible optimization strategies listed in this table is out of the scope of this report.

Implementation level	***Optimization strategy***	***Expected saving***
Algorithm	Algorithmic selection	Orders of magnitude
Behavioral	Concurrency memory	Several times
Power Management	Clock control	10-90%
Register Transfer Level	Structural transformation	10-15%
Technology independent	Extraction/ decomposition	15%
Technology dependent	Technology mapping Gate sizing	20%
Layout	Placement	20%

Table 2.2: Expected power saving by optimization carried out at different implementation levels.

2.15. Summary of the Chapter

In this chapter a brief introduction to the field of digital speech processing and its applications was given.

It was shown that algorithmic optimization and the choice of a fixed-point arithmetic are essential in speech processing applications such as hearing aids or portable communications devices.

In Chapter 3, a methodology for optimization of speech processing algorithms is presented. The emphasis is placed in algorithmic optimization and the study of fixed-point quantization effects.

2.16. References

[CELP97] ICASSP'97 session: "CELP Coding", 12 different papers, *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol.2, pp. 731-778, 1997.

[Chan95] A. Chandrakasan and R. Brodersen, “Minimizing Power Consumption in Digital CMOS Circuits”, *Proc. of the IEEE*, Vol. 83, No. 4, pp. 498-523, 1995.

[Gers94] A. Gersho, “Advances in Speech and Audio Compression”, *Proc. of the IEEE*, Vol. 82, No. 6, 1994.

[Honi88] M. Honig and D. Messerschmitt, *Adaptive Filters: Structures, Algorithms, and Applications*, Kluwer Academic Publisher, Boston, USA, 1988.

[Iked84] O. Ikeda, "Speech Synthesis LSI LC8100", *Proc. of Speech Technology*, New York, pp. 188-191, 1984.

[Kail85] T. Kailath, "Signal Processing in the VLSI Era" in *VLSI and Modern Signal Processing*, ed. by S. Kung, H. Whitehouse, and T. Kailath, Prentice Hall, Englewood Cliffs, NJ, 1985.

[Kond94] A. M. Kondoz, *Digital Speech: Coding for Low Bit Rate Communication Systems* (Chapter 3, 11), Wiley, Chichester, 1994.

[Laps97] P. Lapsley et al., *DSP Processor Fundamentals: Architectures and Features*, IEEE Press Series on Signal Processing, Piscataway, NJ, 1997.

[Lim83] J. Lim (Editor), *Speech Enhancement*, Prentice-Hall Signal Processing Series, Englewood Cliffs, New Jersey, 1983.

[Liu90] Chi-Shi Liu et al., "A Study of Line Spectrum Pair Frequencies for Speaker Recognition", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'90, Vol. 1, pp. 277-280, 1990.

[LOWB97] ICASSP'97 session: "Speech Coding at Low Bit Rates", 14 different papers, *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol.2, pp. 1555-1610, 1997.

[Lunn91] T. Lunner and J. Hellgren, "A Digital Filterbank Hearing Aid Design, Implementation and Evaluation", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'91, Vol. 5, pp. 3661-3664, 1991.

[Neuv93] Y. Neuvo, "Digital Filter Implementation Considerations" in *Handbook for Digital Signal Processing*, ed. by S. Mitra and J. Kaiser, Wiley, New York, 1993.

[Osha87] D. O'Shaughnessy, *Speech Communication: Human and Machine* (Chapter 3, 4, 5, 6 and 7), Addison-Wesley, Reading, 1987.

[Pali88] K. Paliwal, "A Study of Line Spectrum Frequencies for Speech Recognition", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'88, Vol. 1, pp. 485-488, 1988.

[Proa89] J. Proakis and D. Manolakis, *Introduction to Digital Signal Processing,* Macmillan, New York, 1989.

[Raba97] J. Rabaey, *Cad Tools for Low Power*, Electronics Laboratories Advanced Engineering Course on: Architectural and Circuit Design for Portable Electronic Systems, EPFL, Lausanne, 1997.

[Rabi94] L. Rabiner, “Applications of Voice Processing to Telecommunications”, *Proc. of the IEEE*, Vol. 82, No. 2, pp. 199-228, 1994.

[Thon93] T. Thong and Y. Jenq, "Hardware and Architecture" in *Handbook for Digital Signal Processing*, ed. by S. Mitra and J. Kaiser, Wiley, New York, 1993.

[Widr85] B. Widrow and S. Stearns, *Adaptive Signal Processing,* Prentice-Hall, Englewood Cliffs - N.J, 1985.

[Wigg78] R. Wiggins and L. Brantingham, "Three Chip System Synthesizes Human Speech", *Electronics*, Vol. 51, No. 18, pp. 109-116, 1978.

[Work91] Working-group on Communication and Aids for the Hearing-impaired People, "Speech-perception Aids for Hearing-impaired People: Current Status and Needed Research", *J. of the Acoustical Society of America*, Vol. 90, No.2, pp. 637-685, 1991.

Chapter 3
Methodology of Optimization

In this chapter, a methodology for optimization of speech processing algorithms is presented. The emphasis is placed in algorithmic optimization (algorithm choice and simplification) and the study of fixed-point quantization effects.

A practical method for evaluating the behavior of digital signal processing (DSP) algorithms in the case of an implementation using fixed-point 2's complement arithmetic is proposed. A theoretical study of quantization effects is out of the scope of this report and can be found among others in [Jack89] and [Vaid87].

3.1. Methodology of Optimization

Some speech processing applications such as digital hearing aids and portable telecommunications devices, are characterized by very tight requirements in chip size and power consumption as well as the complexity and number of functions to be implemented. The proposed methodology of optimization (Figure 3.1) is aimed to efficient implementation of these devices. A good interrelation between algorithmic level and target architecture is essential in this optimization process.

The system is simulated using a double-precision C program. This program is first used to evaluate the performance of the system and to tune its parameters and then is used as reference system for further optimization and simplification.

For each functional block of the system, a survey of different algorithmic options for its realization is done. Only algorithms that are promising for efficient implementation are chosen, taking into account computational complexity, influence on the performance of the whole system, and the suitability for a fixed-point implementation. These algorithms are modified to reduce computational complexity, improve overall performance, allow better implementation on the target architecture, or improve their fixed-point implementation.

A simulation of fixed-point quantization effects is done to minimize the number of bits required at every node of the algorithm while keeping a good performance.

Implementation on a commercial low-cost fixed-point DSP, such as the DSP56001 is done for real time evaluation and to observe which blocks are computationally more expensive.

Custom VLSI is done using either standard cell approach and automatic CAD tools or the low power architecture and the tool for optimal scheduling of DSP algorithms proposed in [Heub94].

An example of the application of this methodology in the optimal implementation of a noise reduction/ speech enhancement algorithm for digital hearing aids is given in Chapter 4.

3.2. Quantization Effects in Digital Signal Processing

Finite-precision effects are inherent of any digital realization whether it be hardware or software. There are two common forms to represent numbers in a digital computer, fixed-point and floating-point notation. In practice fixed-point implementation leads to more efficient solutions on custom hardware, in terms of area and power consumption. Also, most popular low-cost commercial DSP chips are based on fixed-point arithmetic. Floating-point arithmetic is briefly described in [Proa89] and [Vaid87]. Hereafter, only fixed-point representation is considered.

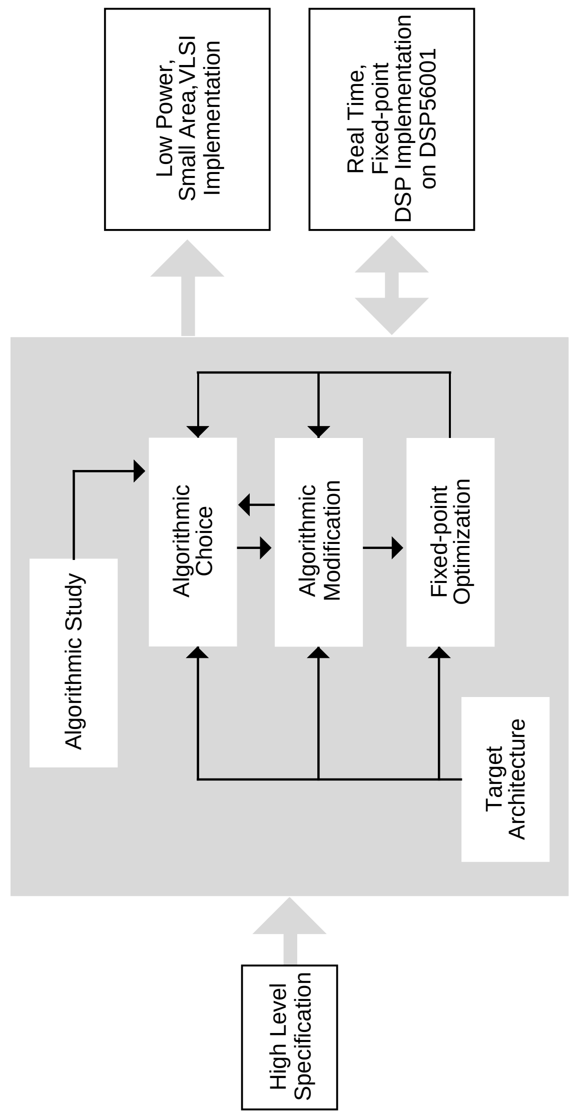


Figure 3.1: Methodology for algorithmic optimization.

3.3. Binary Fixed-point Representation of Numbers

A binary representation of a number is a means of writing the number in terms of powers of two. For example the decimal number 6.375 can be represented as 110.011, an abbreviation for: $1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}$.

A binary number comes with a "binary point". The portion to the left represents an integer (e.g., $110 = 6$) and the portion to the right represents fractions less than one (e.g., $0.011 = 0.375$). In fixed-point notation the binary point is constrained to lie at a fixed position in the bit pattern, as shown in Figure 3.2.

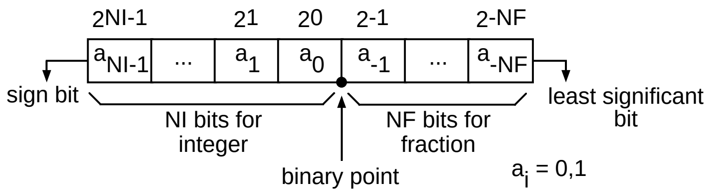


Figure 3.2: Allocation of bits in a word for fixed-point implementation.

The first bit to the left is called the sign bit. The precision of the number system is defined as the increment between two consecutive numbers and is determined by the value of the least significant bit (LSB).

Within the subclass of fixed-point representations, there are three commonly used methods to represent bipolar numbers: sign-magnitude, one's complement, and two's complement representation [Vaid87]. They are all based on the natural binary code, but differ in the way they handle negative numbers.

In the sign-magnitude representation, the term:

$$\sum_{k=-NF}^{NI-2} a_k \cdot 2^k \tag{3.1}$$

always represents the magnitude and the sign is kept in the sign bit.

In the 2's complement notation a positive number is the same as in sign-magnitude representation, and a negative number is given by:

$$x = -a_{NI-1} \cdot 2^{NI-1} + \sum_{k=-NF}^{NI-2} a_k \cdot 2^k \tag{3.2}$$

The negative of a number is obtained by subtracting the corresponding positive number from 2^{NI}. In the case of pure fractional arithmetic (NI=1) a negative number is obtained by subtracting its positive number from 2 (from there the name of 2's complement).

The 1's complement representation is identical to sign-magnitude and 2's complement for positive numbers. A negative number is formed by complementing its corresponding positive number representation. Note that zero is now represented by $00\ldots0$ or $11\ldots1$, which is an undesired ambiguity.

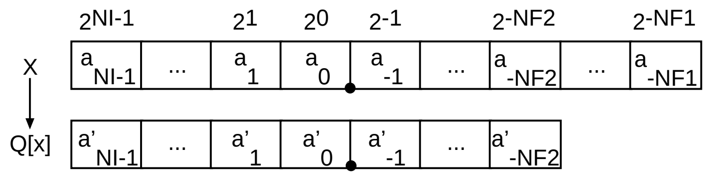
Two's complement arithmetic is easy to implement for both additions and multiplications, and elegantly handles negative numbers. In the work described in this report, only the case of two's complement fixed-point arithmetic is studied.

3.4. Rounding and Truncation

In performing fixed-point computations such as multiplications it is often necessary to quantize a binary fixed-point number x, to another number Q[x], reducing the precision (number of fractional bits) from NF1 to NF2 as shown in Figure 3.3. This can be done via truncation or rounding.

The effect of this reduction of precision is to introduce an error whose power depends on the values of NF1 and NF2 and whose statistical behavior depends on the type of truncation or rounding used. This error is referred to by the generic name of round-off error (whether rounding or truncation is actually employed) and is given by:

$$e = Q[x] - x \tag{3.3}$$



2^{NI-1}
2^1
2^0
2^{-1}
2^{-NF2}
2^{-NF1}
X
Q[x]
a_{NI-1} ... a_1 a_0 a_{-1} ... a_{-NF2} ... a_{-NF1}
a'_{NI-1} ... a'_1 a'_0 a'_{-1} ... a'_{-NF2}

Figure 3.3: Truncation or rounding of a fixed-point binary number.

Regardless of the actual binary representation used, several types of rounding or truncation can be implemented, among them, 2's complement truncation, sign-magnitude truncation, rounding and convergent rounding.

Truncation

In truncation, the least significant bits (NF2 to NF1) are simply dropped, regardless of the sign and the convention to represent negative numbers. If x is positive then:

$$-(2^{-NF2} - 2^{-NF1}) \le e \le 0 \tag{3.4}$$

If x is negative the error depends on which binary representation is used:

Sign-magnitude truncation and 1's complement truncation:

$$0 \le e \le (2^{-NF2} - 2^{-NF1}) \quad \text{and} \quad |Q[x]| \le |x| \tag{3.5}$$

2's complement truncation:

$$-(2^{-NF2} - 2^{-NF1}) \le e \le 0, \quad \text{and} \quad |Q[x]| \ge |x| \tag{3.6}$$

Rounding

In rounding, the value Q[x] is taken to be the nearest possible number to x, thus the error is limited by:

$$-\frac{(2^{-NF2} - 2^{-NF1})}{2} \le e \le \frac{(2^{-NF2} - 2^{-NF1})}{2} \tag{3.7}$$

Rounding is more accurate than truncation but requires more effort in its implementation. The quantization curve and

statistical behavior of the error for rounding and truncation is given in Figure 3.4.

Convergent Rounding

The conventional rounding rounds up any value above one-half and rounds down any value below one-half. The question arises as to which way one-half should be rounded. If it is always rounded one way, the result will eventually be biased in one direction. Convergent rounding solves the problem by rounding down if the number is odd and rounding up if the number is even.

3.5. Dynamic Range, Overflow, and Saturation Arithmetic

In general, a fixed-point DSP implementation is an elaborated interconnection of multipliers, adders and delays in which all the signals involved (including inputs, outputs and internal signals) are represented with fixed-point arithmetic. An overflow occurs when the value of a signal exceeds the dynamic range available. The dynamic range is the range of numbers which can be represented within the arithmetic used. In 2's complement fixed-point arithmetic, this range is given by ($MIN = -2^{NI-1}$, $MAX = 2^{NI-1} - 2^{-NF}$).

Scaling is the process of readjusting some internal gain parameters to avoid overflows. In [Jack89] it is shown how scaling should be performed, depending on the class of inputs. For a fixed total number of bits there is a trade-off between decreasing the probability of overflows by scaling and increasing the round-off error. Therefore scaling is usually applied only to minimize the probability of overflow to a reasonable extent and not to preclude it entirely.

When an overflow actually occurs the resulting distortion is minimized by using clipping or saturation arithmetic. The overflowed result is substituted by the values MIN or MAX, according to its sign.

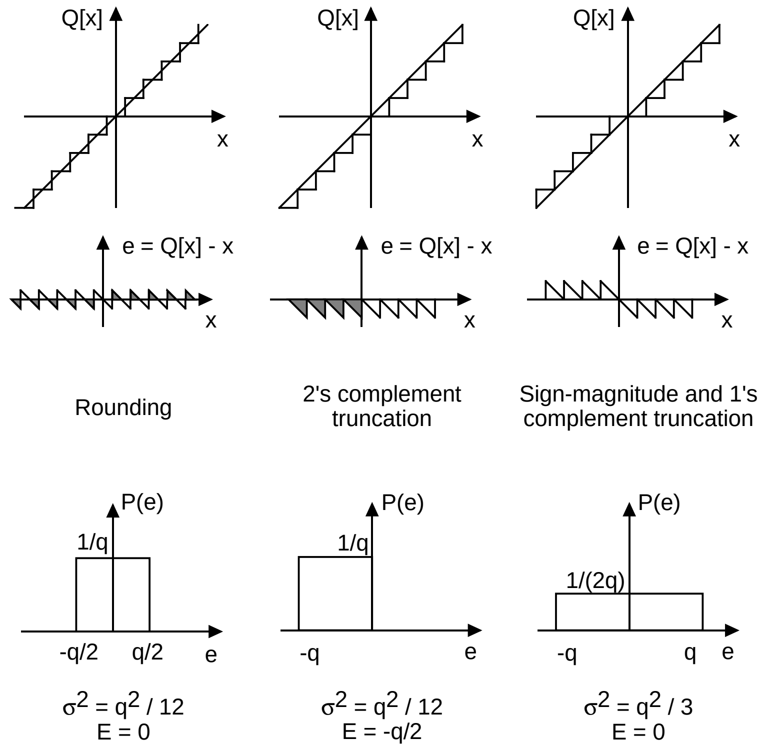


Figure 3.4: Quantization curve, error and statistical behavior of the error (P(e) = pdf, E = mean, σ2= variance and q = 2^{-NF2}–2^{-NF1}).

3.6. Fixed-point Quantization Effects

The implementation of a DSP algorithm using fixed-point arithmetic involves quantization of the signals and parameters of the system. As a result, the overall input-output behavior is not ideal. Quantization is the process of transforming a value into its closest representation in the number system by means of truncation or rounding, and clipping [Jack89].

Two basic types of quantization effects should be distinguished. The first is due to parameter quantization, where the term parameter refers to the fixed values in the algorithm, usually filter coefficients. The second is due to quantization of the input, output and internal signals of the system.

Parameter Quantization

The filter coefficients are only quantized once (in the design process) and those values remain constant in the filter implementation. The effect of coefficient quantization is to deviate the filter response from its ideal (designed) form in a deterministic manner. The quantized design can be checked and, if no longer meets the specifications, can be optimized, redesigned, restructured, and/or more bit could be allocated to satisfy the specifications.

The structure of the digital filter network influences directly its sensitivity to coefficient quantization.

Signal Quantization

The effect of signal quantization is to add an error or noise signal e(n) to the ideal output y(n) of the digital filter. This noise is a composite of the errors from one or more of the following sources, as applicable:

(1) The quantization error of the analog-to-digital converter at the filter input.

(2) The accumulated errors resulting from rounding or truncation within the filter (round-off noise).

(3) The quantization of the output y(n) to fewer bits for input to a digital-to-analog converter or to another digital system.

Source (3) is sometimes overlooked, but because of the accumulation of the noise in (2), more bits are usually allocated to the internal arithmetic of the filter than are required at the output. Hence, the output is usually requantized to fewer bits. It is often reasonable for the input and output quantization to employ the same number of bits, in which case their power noise levels are the same.

3.7. Round-off Noise and Limit Cycles

The internal signals in a digital filter are invariably subject to quantization causing errors in the computed output. Such quantization is a non-linear phenomenon and can be further subdivided into two types of effects called limit-cycles and round-off noise.

Round-off noise affects the filter output in the form of a random disturbance, and can be analyzed by suitable noise modeling and by the use of linear system theory [Jack89].

Limit-cycle oscillations, which contribute to undesirable periodic components at the filter output are due to the fact that quantization is a non linear operation. When such non-linearities exist in a feedback path, they can lead to oscillations [Jack89].

In the case of a zero or constant input, ideally the output of a stable discrete-time filter would asymptotically approach zero or a constant. However, with quantization, it is often found that relatively small limit-cycle oscillations occur.

A different limit-cycle mode, called rolling-pin limit-cycle, has larger amplitude and is rarely encountered [Jack89]. This rolling-pin limit cycles cannot be predicted theoretically and their occurrence is better checked by simulation.

Usually, limit cycles can be reduced to acceptable levels by giving a sufficient number of bits to the signal representation. Another possible type of oscillation, due to overflows, is avoided by using saturation arithmetic.

3.8. Adaptive and Non-linear Algorithms

Adaptive filters are extensively used in signal processing applications. The least-mean-square (LMS) algorithm is the most attractive adaptation scheme because of its computational simplicity. Adaptive algorithms are non-linear in nature, therefore a theoretical analysis of their finite-precision behavior is very difficult and can be performed only under very simplified conditions [Frie92], [Cara84], assuming a stationary input. The well-known theory of finite precision filters with fixed coefficients is inapplicable for adaptive filters.

For instance the representation of coefficients in adaptive filters requires a much longer word-length than in fixed filters.

In the case of an adaptive or non-linear algorithm with non-stationary input such as speech, theoretical analysis becomes untractable. Nevertheless the practical issue of how many bits are required for proper functioning on a fixed-point implementation remains. In this case the only solution is to perform simulations of the quantized algorithms under final working conditions with the appropriate input signals.

3.9. Simulation of Quantization Effects in DSP Algorithms

In the study by simulation of finite word-length effects on a DSP algorithm the main goal is to determine the minimum number of integer and fractional bits (NI,NF) required at every node of the algorithm, keeping the degradation of performance due to quantization effects within acceptable levels. Additionally, this study can also include optimization and modification of the algorithm to simplify its implementation on the target architecture and to improve the use of the dynamic range available.

The Environment Used

The functional blocks of the DSP algorithm are coded in C language, as Matlab functions, and interfaced under Matlab [MATL93]. Inputs and outputs of a Matlab function are matrices, vectors or scalars. When coding a particular functional block, any value that the designer may wish to modify iteratively is set as input while internal variables of particular interest are returned together with the output signals of the functional block.

Different functions were written to allow speech playback, to load a speech file with two different formats, and to run a program on a DSP56001 card from Matlab. The calling syntax and description of this functions is given in Table 3.1.

Within Matlab, the algorithms can be run with different input parameters and the signals involved can be analyzed, displayed and listened to have an immediate feed-back after introducing any change in the system.

da (x, Fs)	Play the sound vector x on the high quality audio I/O card, with sampling frequency Fs.
[x, Fs] = rd_timit ('filename')	Load a speech file that is in TIMIT format.
[x, Fs] = rd_hsw ('filename')	Load a speech file that is in Hypersignal format.
[y0, y1] = run (x0,x1,'prog_name')	Load and run the DSP56001 program 'prog_name' on the DSP56001 card with input x0, x1 and retrieve outputs y0, y1.

Table 3.1: Matlab functions used in fixed-point characterization of digital speech processing algorithms.

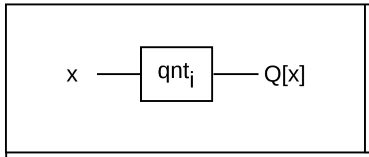
Programs to Simulate Quantization Effects

The first step in the study of the quantization effects on a DSP algorithm is implementing the algorithm in C code using double precision floating-point arithmetic. This in order to determine the optimal parameters that control the behavior of the algorithm and to characterize the "infinite precision" performance of the algorithm. Later this implementation is used as “reference system” to evaluate the degradation in performance of the quantized and optimized system.

A quantized version of the algorithm is obtained from the reference system by placing quantizer operators at different points of the system. A quantizer is an operator that transforms a value into its closest representation in 2's complement fixed-point arithmetic by means of clipping and rounding or truncation. Each placed quantizer is described by its rounding type and its number of integer bits and fractional bits (NI,NF). The place of the quantizers as well as their rounding type is set at the moment of compilation, whereas the number of bits of each quantizer (NI,NF) is given at run time together with other

parameters of the algorithms and the input signal. The value of an overflow counter for each quantizer is returned at the end of the simulation.

An example of the C code for a quantizer with sign-magnitude truncation is given in Figure 3.5. The general expression for a Matlab function and its quantized version can be observed in Table 3.2. The macros and functions to simulate different types of truncation or rounding are given in Appendix A.1.



$\max[i] = -2^{NIi-1} - 2^{-NFi}$
$\min[i] = -2^{NIi-1}$
$\mathrm{con}[i] = 2^{NFi}$

```
/* define rounding policy, in this case is
truncation */
#define ROUND(a) ( (a) < 0 ?  ceil(a)  :  floor(a) )

double qnt( double a, int i)
{
     /* Clipping */
     if        (a>max[i]){a=max[i];ov[i]++;}
     else if   (a<min[i]){a=min[i];ov[i]++;}
     /* round  */
     else      a=(ROUND(con[i]*a))/con[i];
     return a;
}
```

Figure 3.5: The quantizer operator.

General Matlab function (m inputs, n outputs)
[y1, y2, … , yn] = function_name (x1, x2, … , xm)
General quantized Matlab function (m inputs, n outputs)
[y1, y2, … , yn, ov] = qfunction_name (x1, x2, … , xm, prec)

Table 3.2: General expression for a Matlab function and its quantized version.

In Table 3.2, the input precision matrix (prec) and the returned overflow vector (ov) are given by:

$$\text{prec} = \begin{bmatrix} NI_1 & NI_2 & \dots & NI_k \\ NF_1 & NF_2 & \dots & NF_k \end{bmatrix}$$

$$\text{ov} = \begin{bmatrix} ov_1 & ov_2 & \dots & ov_k \end{bmatrix} \qquad (3.8)$$

where k is the number of placed quantizers.

The Input Signals

The input signals used during the simulation must be representative of the kind of input that will be presented to the system in operating conditions.

In the case of speech processing algorithms a collection of speech recordings of good quality, at the appropriate sampling rate, and from a sufficiently large number of speakers should be used. The precision should be greater or equal to the precision of the AD converter in the final implementation.

An existing digitized speech database on CD-ROM, called the TIMIT database [Garo90], was used extensively in the work described in this report.

Measures of Performance

To measure the performance of a quantized (or simplified) system, its output is compared with the output of the reference system using SNR measures. In this context, the output of the reference system is the "non-noisy" signal and the "noise" is the difference between the output of the modified system and the output of the reference system.

Extensive listening tests should be done to determine a threshold of SNR above which it can be assured that the two compared signals cannot be distinguished. This SNR measure should be used as an indicator to locate worst cases, specially when a big amount of different inputs is processed, but should not substitute completely a detailed observation of the interesting cases. This detailed analysis is done by displaying the two compared signals and their difference, and by listening both signals.

3.10. Simulation of DSP56001 Quantization Effects

Producing an optimal real time implementation using DSP56001 assembler is in most cases a time consuming task which is preferably done only once. Arithmetic quantization effects should be studied by simulation before investing much time in assembler coding and in speed and resources optimization. In particular, it should be checked that the dynamic range available in the different registers of the DSP56001 ALU can accommodate the requirements of the algorithm, and the optimum scaling to be applied at each node of the algorithm should be found.

The DSP56001

The DSP56001 is a 24-bit fixed-point, general purpose DSP fabricated by MOTOROLA [MOTO90]. The heart of the processor consists of three execution units operating in parallel: the data arithmetic logic unit (ALU), the address generation unit (AGU) and the program controller. A block diagram of the DSP56001 is given in Appendix A.2.

The data ALU (Figure 3.6) performs all arithmetic and logical operations on data operands. It consists of four 24-bit input registers, two 48-bit accumulator registers with 8-bit extension registers, an accumulator/shifter, two data bus shifter/limiter circuits, and a parallel, single-cycle non-pipelined multiply-accumulate unit (MAC).

Data ALU operations use fractional two's complement arithmetic. Data ALU registers may be read or written, over the X data bus and the Y data bus, as 24- or 48-bit operands. The data ALU is capable of performing any of the following operations in a single instruction cycle: multiplication, multiply-accumulate with positive or negative accumulation, convergent rounding, multiply-accumulate with positive or negative accumulation and convergent rounding, addition, subtraction, a divide iteration, a normalization iteration, shifting and logical operations.

Data ALU source operands, which may be 24, 48, or, in some cases, 56 bits, always originate from data ALU registers.

Arithmetic operations always have a 56-bit result stored in an accumulator. Saturation arithmetic is implemented when an overflow occurs. The arithmetic instructions of the DSP56001 are given in Appendix A.3.

Simulation of DSP56001 Arithmetic

As explained in Section 3.9, finite arithmetic effects are simulated by including quantizer operators in the high-level code of the algorithm. The number of integer and fractional bits of the DSP56001 ALU registers are given in Table 3.3.

Name	***Number of bits <NI, NF>***	***Element***
Word	<1,23>	24-bit X0, X1, Y0, Y1 registers and memory.
Long word	<1,47>	Concatenated X,Y registers and memory.
Accumulator	<9,47>	Accumulator A, B.
Rounded accumulator	<9,23>	Accumulator after rounding.

Table 3.3: Number of integer and fractional bits, NI and NF, for the DSP56001 ALU registers.

The quantization model for some of the most used DSP56001 operations is shown in Figure 3.7.

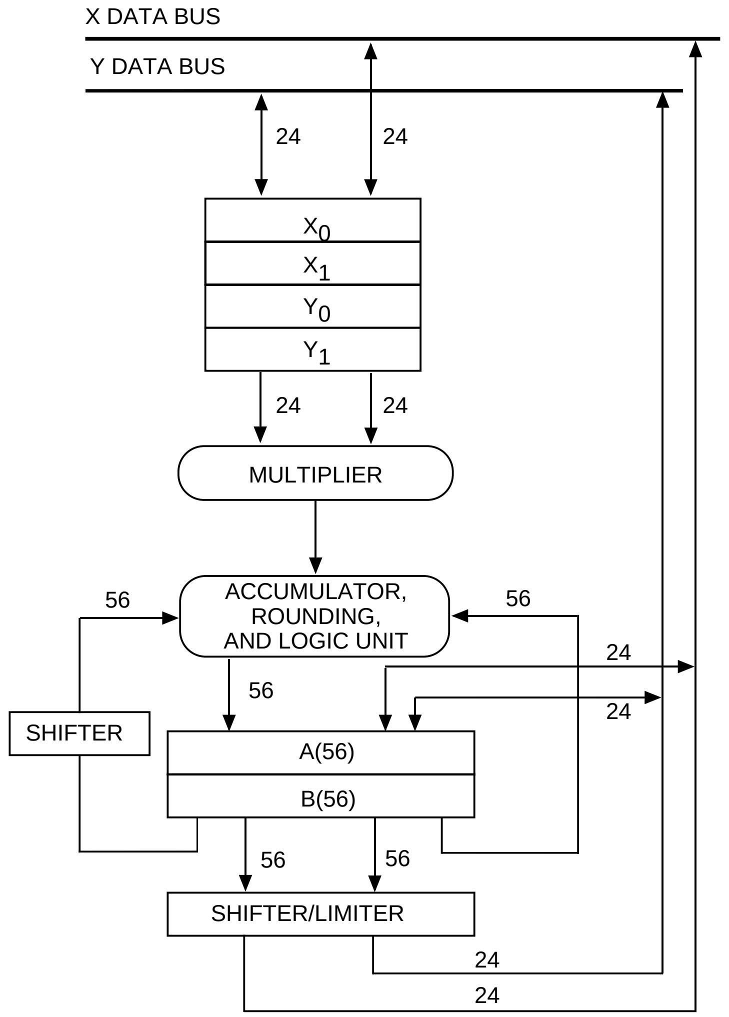


Figure 3.6: Data ALU of the DSP56001.

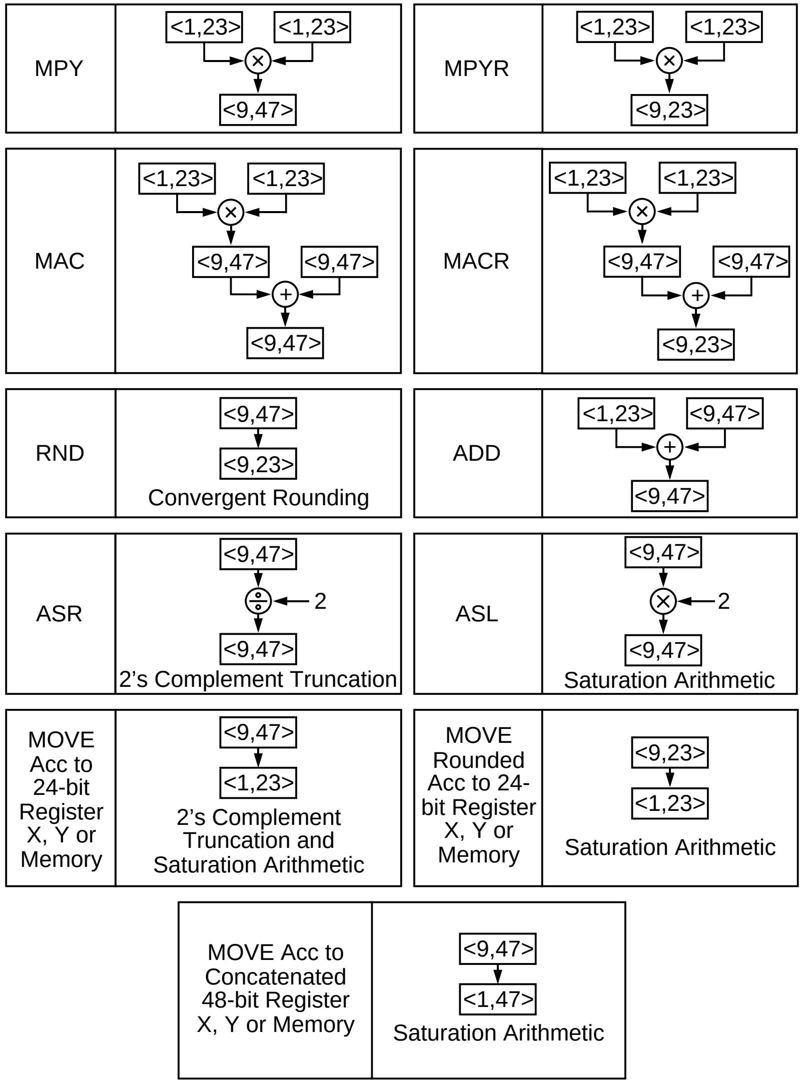


Figure 3.7: Quantization effects of DSP56001 operations.

3.11. Conclusions and Summary of the Chapter

A methodology for optimization of speech processing algorithms was proposed in this chapter, as well as a practical and simple method for evaluating fixed-point quantization effects on these algorithms. Although the application is restricted to digital speech processing algorithms, the method presented is general enough to be easily extended to other classes of DSP algorithms.

The proposed method allows a simulation of the system in final working conditions and at the same time benefit of the flexibility of using high level language, independently of the hardware. In this way, different implementation possibilities can be easily tried out, before doing the actual implementation. Even if the simulation is "independent of the hardware" in the sense that is not running on the hardware itself, many choices such as placing of the quantizers and their rounding strategy are determined by the target architecture.

The characterization of fixed-point arithmetic effects plays an essential role in the optimization of VLSI implementations with tight constraints in size and power consumption such as digital hearing aids and portable devices for telecommunications. In the next chapter, the proposed optimization methodology is used in the implementation of a noise reduction/speech enhancement algorithm for digital hearing aids on both a fixed-point commercial DSP and a low power VLSI architecture.

3.12. References

[Cara84] C. Caraiscos and B. Liu, "A Roundoff Error Analysis of the LMS Adaptive Algorithm", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 32, No. 1, pp. 34-41, 1984.

[Frie92] B. Friedrichs, "Analysis of Finite-precision Adaptive Filters Part I and II", *Frequenz*, Vol. 46, No. 9-10, pp. 219-223 and 262-267, 1992.

[Garo90] J. Garofolo et al., "Darpa TIMIT, Acoustic-phonetic Continuous Speech Corpus CD-ROM", National Institute of Standards and Technology, NISTIR 493, Oct. 1990.

[Gras94] S. Grassi, A. Heubi, M. Ansorge, and F. Pellandini, "Study of a VLSI Implementation of a Noise Reduction Algorithm for Digital Hearing Aids", *Proc. EUSIPCO'94*, Vol.3, pp. 1661-1664, 1994.

[Heub94] A. Heubi, S. Grassi, M. Ansorge, and F. Pellandini, "A Low Power VLSI Architecture for Digital Signal Processing with an Application to Adaptive Algorithms for Digital Hearing Aids", *Proc. EUSIPCO'94*, Vol. 3, pp. 1875-1878, 1994.

[Jack89] L. Jackson, *Digital Filters and Signal Processing* (Chapter 11), Kluwer Academic Publishers, Boston, 1989.

[MATL93] *Matlab User's Guide*, The Math Works Inc., 1993.

[MOTO90] *DSP56000/DSP56001 Digital Signal Processor User's Manual*, DSP56000UM/AD Rev.2, Motorola Inc., 1990.

[MOTO93] A. Chrysafis and S. Lansdowne, “Fractional and Integer Arithmetic Using the DSP56000 Family of General-purpose Digital Signal Processors”, APR3/D Rev. 1, Motorola Inc., 1993.

[Pepe87] R. Pepe and J. Rogers, "Simulation of Fixed-point Operations with High Level Languages", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 35, No. 1, pp. 116-118, 1987.

[Proa89] J. Proakis and D. Manolakis, *Introduction to Digital Signal Processing*, Macmillan, New York, 1989.

[Vaid87] P. Vaidyanathan, "Low-noise and Low-sensitivity Digital Filters" (Chapter 5), in *Handbook of Digital Signal Processing*, ed. by D. Elliott, Academic Press, San Diego, 1987.

Chapter 4
Noise Reduction / Speech Enhancement for Digital Hearing Aids

In this chapter, the optimization methodology explained in Chapter 3 is used for meeting the tight constraints in the physical realization of a noise reduction/speech enhancement algorithm for digital hearing aids.

4.1. Digital Hearing Aids

Analog electroacoustic hearing aids are the primary treatment for most hearing impaired people. They contain the basic functions of amplification, frequency shaping, and limiting of the output signal [Work91]. Digital hearing aids promise many advantages over conventional analog hearing aids, among them the increased precision and programmability of DSP techniques and the possibility of adding new functions such as noise reduction, spectral sharpening and feedback cancellation [Levi87].

On the other hand the physical implementation of digital hearing aids is characterized by very tight requirements in chip size, voltage supply and power consumption, which are very difficult to fulfill given the complexity and number of functions to be implemented together with the real time requirement and large dynamic range of the input signals.

Several algorithms have been proposed to perform the functions of frequency shaping [Lunn91], feedback cancellation [Kate90] and noise reduction [Scha91]. However, the ultimate problem remains the feasibility of a physical implementation of these algorithms, in particular for meeting the constraints of chip size and power consumption. This could be achieved by a careful optimization that ranges from algorithm level, through system and circuit architecture to layout and design of the cell library. The key points in this optimization are among others the choice of a fixed-point arithmetic unit, the optimization of the algorithm minimizing the number of operations and the number of bits required at every node of the algorithm, and a careful match between algorithms and architecture.

4.2. Noise Reduction / Speech Enhancement Algorithms

In the algorithms proposed in [Scha91], spectral sharpening is used for both noise reduction and compensation of the reduced frequency selectivity encountered among many hearing impaired people. The spectral sharpening technique is based on a combination of a gradient adaptive lattice (GAL) linear predictor and two, IIR and FIR, modified lattice filters (synthesis and analysis filters). These algorithms are particularly suitable for a fixed-point VLSI implementation. This is due to the good quantization properties of lattice filters, their modular structure, local interconnections, and rhythmic data flow [Kail85].

The block diagrams of the noise reduction and the speech enhancement algorithms are given in Figure 4.1 and Figure 4.2. The GAL predictor extracts spectral information from the input signal at every sampling instant. This spectral information is encoded in the Parcor coefficients $\{k_m\}$ and used by the analysis and synthesis filters to perform a signal dependent (adaptive) filtering of the input.

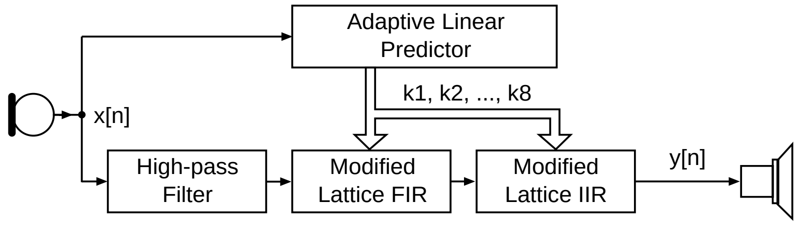


Figure 4.1: Spectral sharpening for noise reduction.

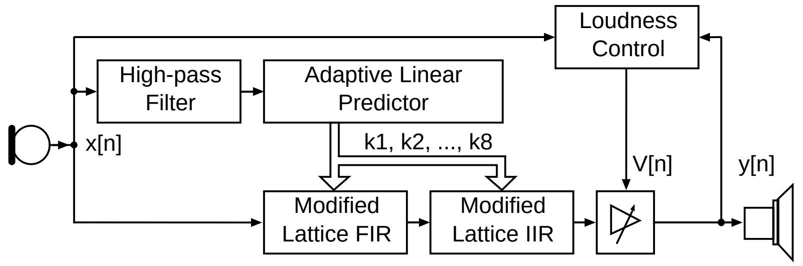


Figure 4.2: Spectral sharpening for speech enhancement.

The sharpening effect applied to a synthesized vowel is shown in Figure 4.3. The noise reduction effect applied to a short phrase is shown in Figure 4.4.

The first cell of the GAL predictor and the analysis and synthesis filters can be observed in Figure 4.5, Figure 4.6 and Figure 4.7, respectively.

To obtain the speech enhancement system, the high pass filter is placed at the input of the GAL and a gain control unit is added at the output of the synthesis filter [Scha91].

Only the noise reduction algorithm was implemented and all the optimization effort was done in the implementation of the computationally expensive core of this algorithm (which contains the GAL predictor and the analysis and synthesis filters).

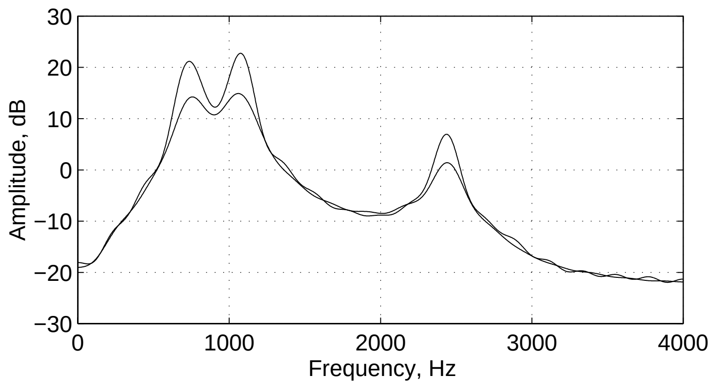


Figure 4.3: The sharpening effect applied to the synthesized english vowel /α/, synthesized using the average formant frequencies given in [Osha87].

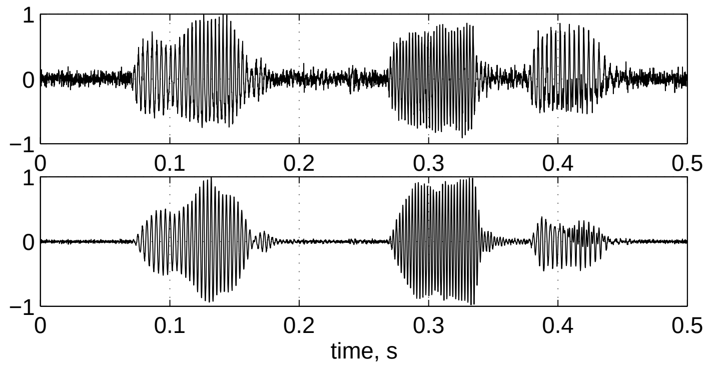


Figure 4.4: Noise reduction applied to the french phrase: “Le pot de...”.

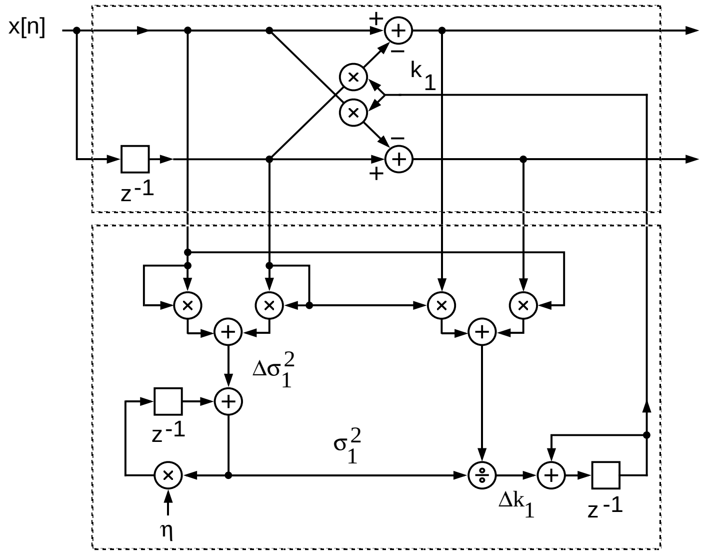


Figure 4.5: GAL predictor (first cell).

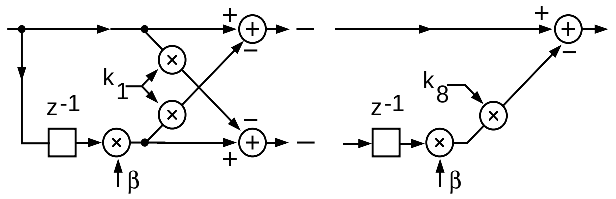


Figure 4.6: Analysis filter.

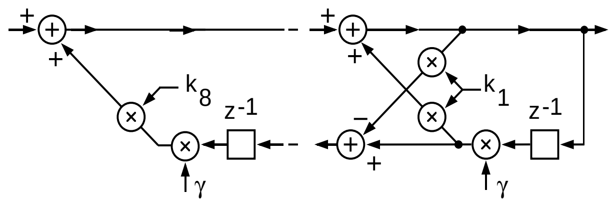


Figure 4.7: Synthesis filter.

4.3. High Level Simulation

A high level simulation of the noise reduction algorithm was done. This simulation was first used to determine the optimal parameters (γ, β and η) and then used as reference system for further simplification and optimization. All the functional blocks were coded in C language as Matlab functions. The listings for the C code of the GAL predictor and the analysis and synthesis filters are given in [Gras95].

A set of 15 files recorded at 8 kHz with 16 bit precision, and the 6300 files from the TIMIT database [Garo90] downsampled to 8 kHz were used as input data. These files were scaled to the range [–1, +1) and their precision was reduced to simulate a 12 bit AD converter. The set of 15 files was used to define the first choice of the number of bits of each quantizer. This choice was tuned by processing some (usually 100) files chosen at random from the TIMIT database. Systematic testing on all the available files was done in some cases such as overflow search.

Testing the Algorithms and Choice of the Parameters

All the functional blocks of the algorithms were implemented and studied separately, in order to understand the influence of the different parameters on each block and on the overall system. These simulations were performed with synthetically produced and digitized speech files.

In particular, the different trade-offs controlling the choice of the parameters were found as well as useful ranges for these values. Subjective and objective measures were used for testing the algorithms. These measures were first used to fine-tune the choice of the parameters obtained with theoretical analysis and then used to characterize the performance of the algorithms [Trog93].

In the case of the noise reduction system, the signal and noise contribution at the output of the system were separated in order to measure the SNR and segmental SNR at the output [Trog93]. This measure was used together with measures of subjectively perceived quality and observations of the time-domain signals. It was found that there is a trade off between

subjective quality and noise reduction. Three different sets of parameters, corresponding to different levels of noise reduction and quality, were found. The improvements in SNR measured using these sets of parameters were 4-5 dB, 5-7 dB, and 7-11 dB, ordered by decreasing quality [Trog93].

The speech enhancement algorithm was evaluated using cepstrum spectral envelope extraction for observing the sharpening effect in dB. This was based on the assumption that efficient speech enhancement requires that the various formants are uniformly emphasized, regardless of their relative power level [Scha91]. Only approximately 6 dB of uniform speech enhancement could be obtained using data synthetically produced. Using real speech as input data, it was not possible to obtain at the same time a reasonable level of speech enhancement and a uniform distribution of the gain at the formants. This could be improved using a higher order high-pass pre-processing filter [Trog93], [Scha91].

The performance of the noise reduction system was also evaluated using informal listening tests with non-impaired subjects for evaluating subjective quality. In the case of non-impaired listeners the sharpening effect is probably perceived as distortion, a "price to pay" for achieving a given noise reduction, while for a hearing impaired individual, it is expected to help in improving intelligibility. Therefore, it is not necessary that the results obtained with normal listeners can be extrapolated to hearing impaired individuals.

Measure of Performance

The high level implementation of the algorithm using double precision floating-point arithmetic is used as reference system.

To measure the performance of a modified system, its output is compared with the output of the reference system using SNR measures, as explained in Section 3.9.

It was found that when the SNR was 15 dB or more, the transfer function of the analysis-synthesis filter of both systems did not differ significantly and their outputs could not be distinguished in listening tests.

4.4. Real Time Implementation on DSP56001

A first implementation of the noise reduction algorithm was done on a DSP56001 for checking real time feasibility and identifying which functional blocks are more time-consuming in view of a possible simplification. These observations are summarized in Table 4.1 for a system of order 8 with a sampling frequency of 8 kHz and clock frequency of 20 MHz. A uniform precision of 24 bit was used throughout the whole system. The serious degradation observed in the performance motivated a more detailed study of the quantization effects.

The computational load of the division was 63% of the time available, therefore a particular effort has to be done in the simplification of this operation. The listings for the assembler implementation and its corresponding fixed-point simulation on C code are given in [Gras95].

Function	***#Cycles***	***%***	***Remarks***
GAL	682	82	Division: 63%
Synthesis	76	9	
Analysis	78	9	
Total	836	100	33% of time available

Table 4.1: Computational load of the first DSP56001 implementation.

4.5. Simplified Division

There is no divide operation in the low power architecture previewed for the final implementation [Heub93]. Also, since the division is a less frequent operation (1 division per 12 multiplications), it is not efficient to implement a full division in a special unit.

GAL algorithms in which the division is replaced with a multiplication by a small constant do not yield the fast convergence independently of the input signal level required for good speech enhancement results. A reasonable compromise is the approximation of the divisor by a power of two.

The measured SNR between the output of a system with simplified division (but otherwise no other change with respect to the reference system) and the reference system was more

than 20 dB. This result shows that the simplified division is feasible.

4.6. Quantization Effects

A simulation of the fixed-point quantization effects was done, following the methodology explained in Chapter 3, in order to determine the minimum word-length and the scaling required at every node of the algorithms [Gras94], [Gras95]. The results obtained in the study of the quantization effects were used for both an efficient real time implementation on a DSP56001 and an implementation on the low power architecture described in [Heub93].

Parameters of the System

The interesting values of γ and β are in the ranges $(0.90, 0.99)$ and $(0.10, 0.50)$ respectively. These parameters were quantized to 12 bit. The effect of this quantization is negligible and was included in the reference system for the remaining of the study. All the simulations were done using values in these ranges and the results obtained hold under these conditions. It was noticed that the system was more sensitive to modifications when $\gamma = 0.99$.

When η is in the range $(0.98, 0.985)$ the GAL algorithm performed well for all input signals. A value of 0.9805 yields to an efficient implementation as explained in Section 4.7.

The Optimized System

The target architecture for the final VLSI implementation is a low power architecture described in [Heub93], [Heub94]. The placing of the quantizers and their rounding strategy is given by the characteristics of the arithmetic unit of the target VLSI architecture. The placing of the quantizers, q_i, is shown (with shadowed boxes) in Figure 4.8 to Figure 4.10. The used rounding strategy is sign-magnitude truncation.

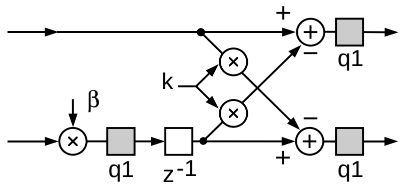


Figure 4.8: Quantizers in the analysis filter.

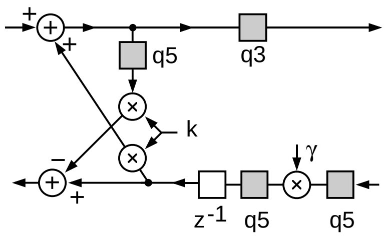


Figure 4.9: Quantizers in the synthesis filter.

The number of bits of each quantizer is also influenced by the final target architecture: the word-length of the multiplier must be a multiple of four and the word-length of the accumulator is twice the word-length of the multiplier.

The minimum number of bits for the quantizers is given in Table 4.2. Using this specification, the measured SNR between the output of the quantized system and the reference system was more than 20 dB. A system with both quantization and simplified division gives more than 15 dB of SNR when compared to the reference system.

No. bits	***q1***	***q2***	***q3***	***q4***	***q5***
N_I	2	2	7	1	7
N_F	14	25	25	15	9

Table 4.2: Number of bits for each quantizer.

The listings for the C code of the optimized system, including the analysis and synthesis filters and both versions of the GAL predictor (with and without simplified division) are given in [Gras95].

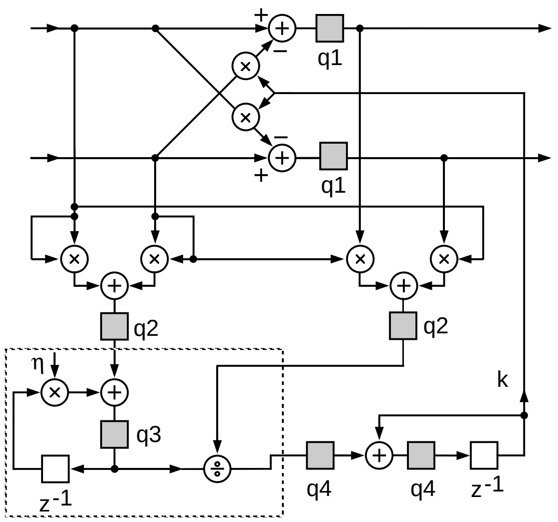


Figure 4.10: Quantizers in the GAL predictor.

Implications for the VLSI Implementation

From the point of view of the word-length requirements we could divide the circuit into two sections. These two sections will be implemented using two different hardware units.

The first section corresponds to the dashed box in the GAL predictor on Figure 4.10 and contains the power estimation recursion and the division. This section requires a higher dynamic range (although not necessarily a higher precision). It contains the two most critical operations of the system which are the long-word (32 bit) multiplication by η and the computationally expensive long-word division. Using a fixed choice of $\eta = 0.9805 = 1 - 2^{-6} - 2^{-8}$ and the simplified division yields to an efficient implementation on a dedicated unit that contains a 32-bit adder, and the logic for the approximation of the power estimation by a power of two. The multiply accumulate is substituted by two additions and two hard-wired shifts.

The second section corresponds to the rest of the GAL algorithm together with the analysis and synthesis filters. The word-length requirements of this section are met by 16 bit multipliers with 32 bit adder-accumulators.

Implications for the DSP56001 Implementation

In Table 4.2 it is observed that at some nodes of the algorithm the minimum word-length required exceeds the uniform 24 bit word-length used in the first DSP56001 implementation. Seven steps of normalization and denormalization were added at these nodes to extend the dynamic range of the temporal registers of the DSP56001. The computational load of the GAL almost doubled as seen in Table 4.3, but the real time constraint is still met. The analysis and synthesis blocks were left unchanged but a 6-bit scaling was included at the input of the synthesis filter to compensate the amplification introduced by this block. Each functional block was coded separately on DSP56001 assembler and simulated in C language (including the effects of the arithmetic used in the DSP56001). Both implementations gave exactly the same results allowing the verification of the accurateness of the simulations. The listings for both implementations are given in [Gras95].

The output of this second DSP56001 implementation is virtually equal to the output of the reference system, with a measured SNR of more than 80 dB. Systematic search on the TIMIT database showed no overflows. From a practical point of view, the DSP implementation was a good verification of the results obtained in the study of the quantization effects.

Function	***#Cycles***	***%***	***Remarks***
Decorrelator	1530	91	Division: 31%
Synthesis	76	4.5	
Analysis	78	4.5	
Total	1684	100	67% of time available

Table 4.3: Computational load of the second DSP56001 implementation.

4.7. VLSI Implementation

An implementation of the optimized system was done using the target low power VLSI architecture described in [Heub93]. The architecture takes advantage of the regularity of the algorithm to simplify the scheduling and the hardware implementation. The processor architecture and modules are organized in a way to limit the overall data transfer to the strict minimum, local data traffic being preferred versus global traffic.

Larger memories are split into a set of smaller memories where a single one is activated at a time. A sequential dynamic memory is used for storing the Parcor coefficients, another for storing the state variables of the analysis and synthesis filters, and two remaining ones for the variables of the linear predictor.

The arithmetic unit is a serial-parallel unit optimized for performing scalar products. The number of relevant partial products occurring in the multiplications is reduced at least by a factor of two using Booth's recoding scheme. Two arithmetic units of this kind are used to achieve a sufficient computational throughput. This for the implementation of the analysis and synthesis filters, and the portion of the GAL outside the dashed box in Figure 4.10.

The dedicated unit for the implementation of the portion of the GAL inside the dashed boxed in Figure 4.10 was realized separately using a low power standard cell library.

The scheduling was hierarchically organized to limit the processing rate of each module to the strict minimum using an adapted version of the "TABU search" optimization technique, which is particularly suitable for the scheduling of DSP algorithms.
The details of this implementation are given in [Heub94]. The floor plan and the layout are shown in Figures 4.11 and 4.12. The resulting silicon area was approximately 4 mm^2 using VLSI Technology's CMN12 1.2 µm CMOS process. The estimated power consumption is 0.65 mW at 2V.

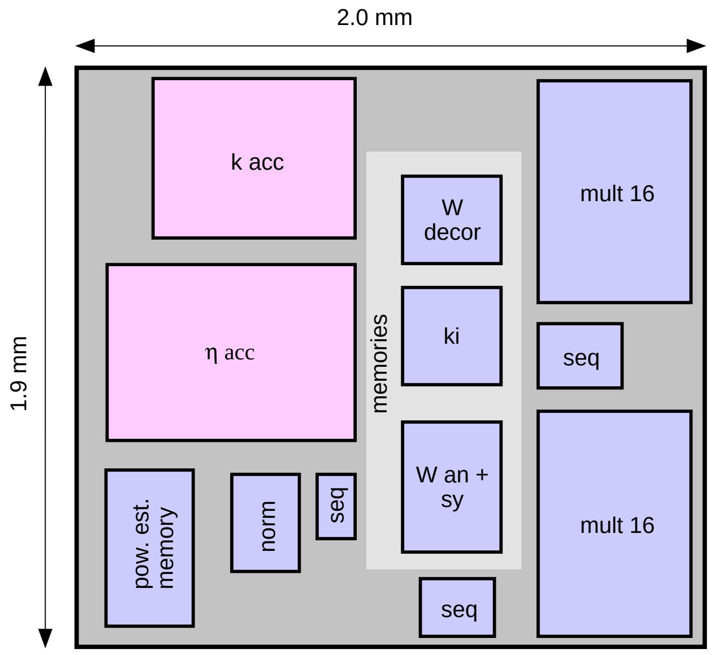


Figure 4.11: Floor plan for the VLSI implementation.

4.8. Further Work

Some possible extensions, and applications of the work described in this chapter are given next.

Given that lattice filters and lattice linear predictors are used in many areas of speech processing such as coding, synthesis and recognition [Osha87], the experience obtained in studying and implementing the algorithms proposed in [Scha91] can be reused in other different applications.

Speech Coding

The GAL linear predictor finds application in backward predictive speech coders such as the 16 kbps ADPCM coder proposed in [Scha90].

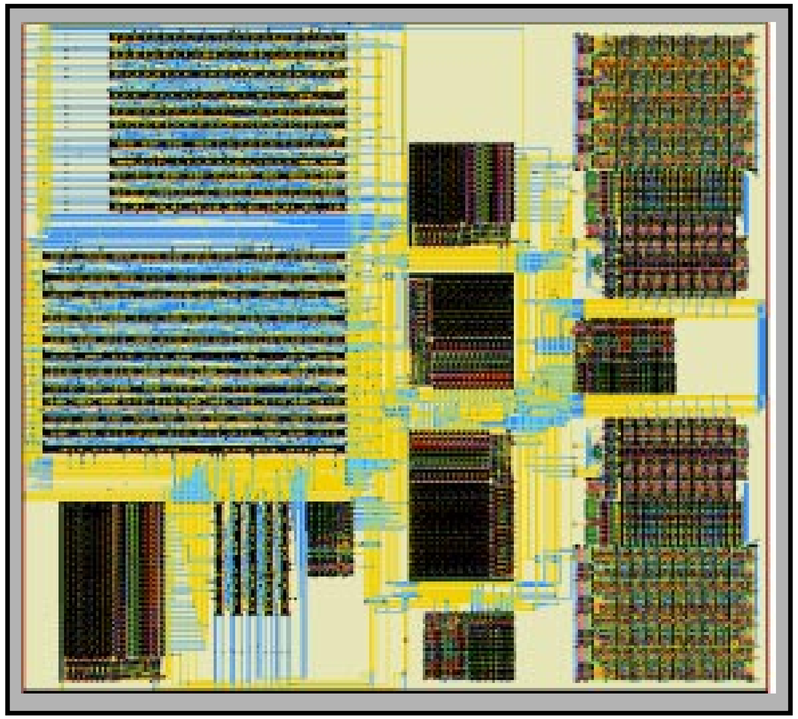


Figure 4.12: Layout for the VLSI implementation.

The high level implementation and testing of this coder was done within a student project [Kunz94]. As the more computationally expensive block of this system is the GAL predictor, the optimized VLSI implementation of this functional block can be reused in an efficient implementation of this speech coder with application in portable devices.

The analysis and synthesis filters studied in this chapter are the basis for the postfiltering algorithm proposed in [Chen87]. The use of this postfiltering technique is now very popular in CELP coders, such as the CELP FS1016 (see § 5.11). These analysis and synthesis filters are also used in CELP coders for perceptual weighting of the error between the original and synthesized speech. Thus the optimization of these filters can be reused for efficient implementation of the CELP FS1016 speech coder.

Frequency Shaping for Digital Hearing Aids

The conventional analog hearing aid always contains the basic function of frequency shaping which provides different levels of amplification for different frequency ranges so as to fit as much of the speech signal as possible between the threshold of audible sound and the ceiling of a too-loud sound [Work91]. A filterbank suitable for frequency shaping in a digital hearing aid is proposed in [Lunn91]. In this algorithm, interpolated half-band FIR filters are used to minimize the number of multiplications per sample. This algorithm was implemented and studied as part of a student work [Hues94]. This filterbank was used as an application example in another student work [Henn94] for estimating the size and power consumption in the case of a VLSI implementation. This work was based on the low power architecture described in [Heub93] and used a standard cells approach (low power library Csel_Lib from CSEM). The resulting silicon area was approximately 3 mm^2 and the estimated power consumption was 0.3 mW at 2V (without sequencing unit).

As the results obtained are promising, it would be interesting to apply the proposed methodology of optimization to the implementation of this functional block.

4.9. Conclusions and Summary of the Chapter

The optimization methodology explained in Chapter 3 was used for meeting the tight constraints in the physical realization of a noise reduction/speech enhancement algorithm for digital hearing aids. The emphasis was placed in the study of the quantization effects and algorithmic optimization.

The interrelation between the target VLSI architecture and the algorithmic level plays an essential role in the optimization process. The resources available in the architecture influenced some choices at the algorithmic level, whereas some constraints and particular needs of the algorithm forced some choices in the VLSI implementation.

The proposed simplification of the gradient adaptive lattice algorithm improves the efficiency in the implementation while keeping good convergence properties.

From a practical point of view, the approach using real input signals is an appropriate means for the characterization of the systems in final operating conditions. The implementation on a commercial fixed-point DSP is an important intermediate step which allows real time evaluation and gives further information on the behavior of the final implementation.

Given that lattice filters and lattice linear predictors are used in many areas of speech processing, the results obtained can be used in other applications where the needs of reduced size and power consumption plays an important role such as portable devices for telecommunications.

4.10. References

[Chen87] J. Chen and A. Gersho, "Real-time Vector APC Speech Coding at 4800 bps with Adaptive Postfiltering", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'87, Vol. 3, pp. 2185-2188, 1987.

[Garo90] J. Garofolo et al., "Darpa TIMIT, Acoustic-phonetic Continuous Speech Corpus CD-ROM", National Institute of Standards and Technology, NISTIR 493, Oct. 1990.

[Gras94] S. Grassi, A. Heubi, M. Ansorge, and F. Pellandini, “Study of a VLSI Implementation of a Noise Reduction Algorithm for Digital Hearing Aids”, *Proc. EUSIPCO’94*, Vol. 3, pp. 1661-1664, 1994.

[Gras95] S. Grassi, *Simulation of Fixed-point Quantization Effects on DSP Algorithms*, IMT Report No 375 PE 03/95, University of Neuchâtel, IMT, 1995.

[Henn94] C. Henny, *Unité Arithmétique Faible Consommation Dédiée au Calcul de Produits Scalaires*, (in French), practical semester project, IMT Uni-NE, winter semester 1993/94, Neuchâtel, 1994.

[Heub93] A. Heubi, M. Ansorge, and F. Pellandini, ("Low Power VLSI Architecture for Digital Signal Processing") "Architecture VLSI Faible Consommation pour le Traitement Numérique du Signal”, *Proc. GRETSI'93*, Vol. 2, pp. 3661-3664, 1993.

[Heub94] A. Heubi, S. Grassi, M. Ansorge, and F. Pellandini, "A Low Power VLSI Architecture for Digital Signal Processing with an Application to Adaptive Algorithms for Digital Hearing Aids", *Proc. EUSIPCO'94*, Vol. 3, pp. 1875-1878, 1994.

[Hues94] O. Huesser, *Filtres de Compensation Spectrale pour Prothèses Auditives Numériques,* (in French), practical semester project, IMT Uni-NE, winter semester 1993/94, Neuchâtel, 1994.

[Kail85] T. Kailath, "Signal Processing in the VLSI Era" in *VLSI and Modern Signal Processing,* ed. by S. Kung, H. Whitehouse, and T. Kailath, Prentice-Hall, Englewood Cliffs, N.J., 1985.

[Kate90] J. Kates, "Feedback Cancellation in Hearing Aids", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'90, Vol. 2, pp. 1125-1128, 1990.

[Kunz94] S. Kunzi, *Etude et Implémentation d'un Codec ADPCM Basé sur un Prédicteur Adaptatif à Gradient,* (in French), practical semester project, winter semester 1993/94, Ecole polytechnique fédérale de Lausanne, Laboratoire de microtechnique EPFL - UNI NE, Neuchâtel, 1994.

[Levi87] H. Levitt, "Digital Hearing Aids: A Tutorial Review", *J. of Rehabilitation, Research and Development*, Vol. 24, No. 4, pp. 7-20, 1987.

[Lunn91] T. Lunner and J. Hellgren, "A Digital Filterbank Hearing Aid Design, Implementation and Evaluation", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'91, Vol. 5, pp. 3661-3664, 1991.

[Osha87] D. O'Shaughnessy, *Speech Communication: Human and Machine* (Chapter 3), Addison-Wesley, Reading, 1987.

[Rutt85] M. Rutter, "An Adaptive Lattice Filter" in *VLSI Signal Processing: a Bit-serial Approach*, ed. by P. Denyer and D. Renshaw, Addison-Wesley, 1985.

[Scha90] A. Schaub, "Backward-adaptive Predictive DPCM", *ASCOM Technical Review*, No.1, pp. 12-19, 1990.

[Scha91] A. Schaub and P. Straub, "Spectral Sharpening for Speech Enhancement/Noise Reduction", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'91, Vol. 2, pp. 993-996, 1991.

[Trog93] J. Troger, *Filtrage Adaptatif pour la Réduction du Bruit Appliqué au Traitement de la Parole,* (in French), diploma work, winter semester 1992/93, Ecole polytechnique fédérale de Lausanne, Laboratoire de microtechnique EPFL - UNI NE, Neuchâtel, 1993.

[Work91] Working-group on Communication and Aids for the Hearing-impaired People, "Speech-perception Aids for Hearing-impaired People: Current Status and Needed Research", *J. of the Acoustical Society of America*, Vol. 90, No.2, pp. 637-685, 1991.

Blank page

Chapter 5
Line Spectrum Pairs and the CELP FS1016 Speech Coder

This chapter gives the theoretical fundamentals for understanding the line spectrum pair (LSP) representation of linear predictive coding (LPC) coefficients with application to narrowband speech coding. It also explains the structure of the CELP FS1016 speech coder, in particular the spectral analysis block, in which LPC analysis with LSP representation is used.

These concepts are used in Chapter 6, in which two novel efficient algorithms for LPC to LSP conversion are presented and in Chapter 7, in which the DSP56001 optimized implementation of the CELP FS1016 spectral analysis block is given.

5.1. LPC Analysis

Linear predictive coding (LPC) is widely used in different speech processing applications for representing the envelope of the short-term power spectrum of speech.

In LPC analysis of order p, the current speech sample s(n) is predicted by a linear combination of p past samples, $\hat{s}(n)$:

$$\hat{s}(n) = -\sum_{k=1}^{p} a_p(k) \cdot s(n-k) \tag{5.1}$$

where $\hat{s}(n)$ is the predictor signal and $\{a_p(1),\ldots,a_p(p)\}$ are the LPC coefficients. The calculation of these coefficients is given in section 5.2. The value $\hat{s}(n)$ is subtracted from s(n), giving the residual signal e(n), with reduced variance:

$$e(n) = s(n) - \hat{s}(n) = s(n) + \sum_{k=1}^{p} a_p(k) \cdot s(n-k) \tag{5.2}$$

Taking the z transform of Equation (5.2) gives:

$$E(z) = A_p(z) \cdot S(z) \tag{5.3}$$

where S(z) and E(z) are the transforms of the speech signal and the residual signal respectively, and $A_p(z)$ is the LPC analysis filter of order p:

$$A_p(z) = 1 + \sum_{k=1}^{p} a_p(k) \cdot z^{-k} \tag{5.4}$$

This filter is used to remove the short term correlation of the input speech signal, giving an output E(z) with approximately flat spectrum. The short-term power spectral envelope of the speech signal can therefore be modeled by the all-pole synthesis filter:

$$H_p(z) = \frac{1}{A_p(z)} = \frac{1}{1 + \sum_{k=1}^{p} a_p(k) \cdot z^{-k}} \tag{5.5}$$

Equation (5.3) is the basis for the LPC analysis model. Conversely, the LPC synthesis model (see § 2.4) consists of an excitation source E(z), providing input to the spectral shaping filter $H_p(z)$, to yield the synthesized output speech S(z):

$$S(z) = H_p(z) \cdot E(z) \tag{5.6}$$

E(z) and $H_p(z)$ are chosen following certain constraints, so that S(z) is as close as possible in some sense to the original speech.

5.2. Calculation of the LPC Coefficients

In the classical least-squares method, the LPC coefficients are determined by minimizing the mean energy of the residual signal, given by:

$$\varepsilon_p = \sum_{-\infty}^{\infty} e^2(n) = \sum_{-\infty}^{\infty} \left[s(n) + \sum_{k=1}^{p} a_p(k) \cdot s(n-k) \right]^2 \tag{5.7}$$

the summation range is limited by windowing either the speech or the residual signal, leading to the autocorrelation or covariance method, respectively. The autocorrelation method is computationally more efficient than the covariance method and the resulting synthesis filter is always stable.

Autocorrelation Method and Durbin's Recursion

As speech is non-stationary, a frame of N samples of the speech signal, $\{s_1,\ldots,s_N\}$ is windowed using Hamming or other tapered cosine windows. The length of the frame is usually 20 to 30 ms for speech sampled at 8 kHz. Minimization of ε_p with respect to the LPC coefficients leads to the Yule-Walker equations:

$$\mathbf{R}_p \cdot \mathbf{a}_p = -\mathbf{r}_p \tag{5.8}$$

where:

$$\mathbf{R}_p = \begin{bmatrix} r_0 & r_1 & r_2 & \cdots & r_{p-1} \\ r_1 & r_0 & r_1 & \cdots & r_{p-2} \\ r_2 & r_1 & r_0 & \cdots & r_{p-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & r_{p-3} & \cdots & r_0 \end{bmatrix} \quad , \mathbf{a}_p = \left[a_p(1) \ \cdots \ a_p(p) \right]^T \quad , \mathbf{r}_p = \left[r_1 \ r_2 \ \cdots \ r_p \right]^T \tag{5.9}$$

and r_k is the *k*-th autocorrelation coefficient of the windowed speech signal:

$$r_k = \sum_{n=k}^{N-1} w(n) \cdot s(n) \cdot w(n-k) \cdot s(n-k) \tag{5.10}$$

here $\{w(n)\}$ is the window function of N samples. The LPC coefficients are given by:

$$\mathbf{a}_p = -\mathbf{R}_p^{-1} \cdot \mathbf{r}_p \tag{5.11}$$

The autocorrelation matrix $\mathbf{R}_p$ has a Toepliz structure, leading to the solution of Equation (5.11) through the very efficient Levinson-Durbin recursion, which is described as follows:

$$\begin{aligned}
&\varepsilon_0 = r_0 \\
&\text{for } 1 \le m \le p \quad : \\
&a_m(0) = 1 \\
&a_m(m) = k_m = \frac{\left[-r_m - \sum_{i=1}^{m-1} a_{m-1}(i).r_{m-i} \right]}{\varepsilon_{m-1}} \\
&a_m(j) = a_{m-1}(j) + k_m . a_{m-1}(m-j) \qquad \text{for} \quad 1 \le j \le m-1 \\
&\varepsilon_m = \varepsilon_{m-1}(1-k_m^2)
\end{aligned} \tag{5.12}$$

The values $\{k_m\}$ are known as the Parcor (partial correlation) or reflection coefficients. In the case of an order $p = 10$, the computational cost of the Levinson-Durbin recursion is 110 multiplications, 100 additions, and 10 divisions.

An order of $p = 10$ is typically used for narrowband or telephone bandwidth (300-3400 Hz) speech sampled at 8 kHz. Hereafter, an order of $p = 10$ is assumed.

The position of the zeros of the 10-th order LPC analysis filter for a 30 ms segment of the vowel /æ/ is shown in Figure 5.1 (on page 84). These zeros correspond to the poles of the LPC synthesis filter, whose power spectrum is shown in Figure 5.3. The formants are the resonances or sharp peaks in the power spectrum, and are due to poles close to the unit circle. The bandwidth of the formants is narrower as the poles are closer to the unit circle. The LPC coefficients are an attractive description of the spectral envelope since they describe the perceptually important spectral peaks more accurately than the spectral valleys [Kond94].

5.3. Bandwidth Expansion

LPC analysis does not estimate accurately the spectral envelope for high pitch voiced speech due to the harmonic spacing, which is too large to provide adequate sampling of the spectral envelope [Pali95a]. Such inaccuracy occurs mainly in formant

bandwidths, which are underestimated by a large amount, resulting in metallic sounding synthesized speech.

One method to overcome this problem is bandwidth expansion, in which each LPC coefficient $a_p(k)$ is multiplied by a factor γ^k ($\gamma < 1$), moving the poles of $H_p(z)$ inward by a factor of γ and expanding the bandwidths of all the poles by the same amount ΔB, given by:

$$\Delta B = -\frac{F_s}{\pi} \cdot \ln(\gamma) \tag{5.13}$$

where F_s is the sampling frequency. The resulting bandwidth expanded LPC synthesis filter is given by:

$$H'_p(z) = \frac{1}{A'_p(z)} = \frac{1}{1 + \sum_{k=1}^{p} \left[a_p(k) \cdot \gamma^k \right] \cdot z^{-k}} \tag{5.14}$$

As bandwidth expansion decreases spectral sensitivity around the spectral peaks, it is also beneficial for quantization of LPC coefficients (see § 5.4).

Bandwidth expansion is commonly used in speech coders, with typical values of γ between 0.996 and 0.988, at a sampling frequency of 8 kHz, corresponding to 10 to 30 Hz of expansion.

5.4. Quantization of the LPC Coefficients

In low-bit rate speech coding, the LPC coefficients are widely used to encode spectral envelope. In forward LPC-based coders, the LPC coefficients are calculated from the original speech input, quantized and transmitted frame-wise. The transmission of these coefficients has a major contribution to the overall bit rate. Thus, it is important to quantize the LPC coefficients using as few bits as possible without introducing excessive spectral distortion and with reasonable complexity. A very important requirement is that the all-pole synthesis filter $H_p(z)$ remains stable after quantization.

Objective Measure of LPC Quantization Performance

The root-mean-square spectral distortion, which is commonly used for evaluating the performance of LPC quantization [Pali95a], is defined, for a frame n, as follows:

$$SD_n = \sqrt{\frac{1}{(\Omega_2 - \Omega_1)} \int_{\Omega_1}^{\Omega_2} \left[10 \log_{10} \frac{S_n(\Omega)}{Sq_n(\Omega)} \right]^2 d\Omega} \quad \text{(dB)} \tag{5.15}$$

where Ω is the frequency in Hz, and the frequency range is given by Ω_1 and Ω_2. A frequency range of 125-3400 Hz is used in [Rama95], for speech sampled at 8 kHz, while a range of 0-3000 Hz is used in [Pali95a] and [Atal89]. A most common practice is to use the full band, 0-4000 Hz. This is the frequency range that will be used throughout this report.

In Equation (5.15), $S_n(\Omega)$ and $Sq_n(\Omega)$ are the original and quantized spectrum of the LPC synthesis filter, associated with the *n*-th frame of speech:

$$S_n(\Omega) = \frac{1}{\left| A_n(z = e^{j\frac{2\pi\Omega}{Fs}}) \right|^2}, \qquad Sq_n(\Omega) = \frac{1}{\left| \hat{A}_n(z = e^{j\frac{2\pi\Omega}{Fs}}) \right|^2} \tag{5.16}$$

here $A_n(z)$ and $\hat{A}_n(z)$ are the original and quantized LPC analysis filters. The subindex n refers to the *n*-th frame, and not to the order of the LPC filter. In practice the spectra $S_n(\Omega)$ and $Sq_n(\Omega)$ are evaluated using fast Fourier transform (FFT). Alternatively, an efficient method for estimating the root-mean-square spectral distortion is the cepstral measure [Gray76].

The spectral distortion is evaluated for all the N_f frames in the test data. Its average value $\overline{SD}$ represents the distortion associated with a particular quantizer:

$$\overline{SD} = \frac{1}{N_f} \sum_{n=1}^{N_f} SD_n \tag{5.17}$$

Transparent quantization of LPC information means that the LPC quantization does not introduce any audible distortion in the coded speech. The spectral distortion measure is known

to have a good correspondence with subjective measures and the

following conditions are considered sufficient (but not necessary) to achieve transparent quantization [Pali95a]:

- An average spectral distortion of less than about 1 dB.
- No outlier frames with spectral distortion larger than 4 dB.
- Less than 2% of outlier frames with spectral distortion in the range of 2-4 dB.

Alternative Representations of LPC Coefficients

The LPC coefficients are not suitable for quantization because of their high spectral sensitivity. Small quantization errors in the individual LPC coefficients produce relatively large spectral errors, and can also result in instability of the quantized all-pole synthesis filter. To avoid unacceptable distortion, a large amount of bits (80-100 bits/frame) is needed for scalar quantization of the LPC coefficients. It is therefore necessary to transform the LPC coefficients into a set of equivalent parameters which have less spectral sensitivity and ensure stability of the all-pole filter after quantization. Suitable representations are the reflection coefficients (RC), the log-area ratio (LAR), the inverse sine (IS), and the line spectrum pairs (LSP).

The reflection coefficients $\{k_m\}$ (see § 5.2) are spectrally less sensitive to quantization than the LPC coefficients. They are bounded in magnitude by unity, and the stability of the all pole filter is easily ensured by keeping this bound on the quantized reflection coefficient. These coefficients are also very important in the physical realization of the all-pole synthesis filter, as they are the multipliers of a lattice filter realization, which is suitable for fixed-point implementation. The forward and backward transformations are given below:

LPC to RC transformation:

$$\text{for } m = p, p-1, \ldots, 1:$$
$$k_m = a_m(m)$$
$$a_{m-1}(j) = \frac{a_m(j) - k_m \cdot a_m(m-j)}{1 - k_m^2} \qquad \text{for} \quad 1 \le j \le m-1 \qquad (5.18)$$

RC to LPC transformation:

$$\text{for } m = 1, \ldots, p:$$
$$a_m(m) = k_m$$
$$a_m(j) = a_{m-1}(j) + k_m \cdot a_{m-1}(m-j) \qquad \text{for} \quad 1 \le j \le m-1 \qquad (5.19)$$

About 50 bits/frame are required for transparent quantization using uniform scalar quantization of the reflection coefficients, and 36 bits/frame using non-uniform scalar quantization [Pali95a].

Although the reflection coefficients are more suitable for quantization than the LPC coefficients, they have a non-flat spectral sensitivity, with absolute values near unity requiring more accuracy than values away from unity. This problem can be overcome expanding the regions near $|k_m| = 1$ by means of non-linear transformations, such as the log-area ratio (LAR) and the inverse sine (IS). The forward and backward transformations are given below:

RC to LAR transformation:

$$g_m = \log\left(\frac{1+k_m}{1-k_m}\right) \qquad \text{for} \quad m = 1, \ldots, p \qquad (5.20)$$

LAR to RC transformation:

$$k_m = \frac{10^{gm} - 1}{10^{gm} + 1} \qquad \text{for} \quad m = 1, \ldots, p \qquad (5.21)$$

RC to IS transformation:

$$s_m = \sin^{-1}(k_m) \qquad \text{for} \quad m = 1, \ldots, p \qquad (5.22)$$

IS to RC transformation:

$$k_m = \sin(s_m) \qquad \text{for} \quad m = 1, \ldots, p \qquad (5.23)$$

Non-uniform scalar quantization using the IS and LAR representation requires 34 bits/frame for transparent quantization [Pali95a]. The major drawback of these representations is that the frame to frame correlation of LPC parameters is not highlighted [Kond94].

A widely used representation of LPC coefficients is line spectrum pair (LSP) parameters. For scalar quantization, it performs only slightly better than LAR, but this representation has several properties which are desirable for quantization as will be explained in Section 5.7. Although this representation is also referred to as line spectrum frequencies (LSF), the term LSP is adopted hereafter.

5.5. Interpolation of the LPC Coefficients

In speech coding systems, LPC analysis is generally carried out on a frame-by-frame basis with a new set of parameters computed, quantized and transmitted at frame intervals of 20 to 30 ms. This slow update of frames can lead to large changes in LPC parameters in adjacent frames, which may introduce undesired transients or clicks in the reconstructed signal. To overcome this problem, interpolation of LPC parameters is used at the receiver to get smooth variations in their values. Usually, interpolation is done linearly, at equally spaced time instants called sub-frames. Four sub-frames are generally used.

The interpolation is not done directly on the LPC coefficients since the interpolated all-pole synthesis filter can become unstable [Atal89]. In fact, stability issues in the interpolation are very similar to those encountered in quantization (see § 5.4). Interpolation on the reflection coefficients, log area ratios, inverse sine coefficients and LSP parameters always produce stable filters. Thus, it is natural to use for interpolation the same LPC representation that was used for quantization. In [Pali95b] it is shown that LSP representation has the best interpolation performance.

5.6. Line Spectrum Pairs

The LSP representation of LPC coefficients was first introduced by Itakura in [Itak75]. This representation is widely used in the domain of speech coding due to its desirable quantization properties, such as bounded range, intra-frame and inter-frame correlation, and simple check of filter stability [Kond94]. Additionally, LSP representation allows frame to frame interpolation with smooth spectral changes (see § 5.5).

Use of LSP Representation in Speech Coding

LSP representation of 10-th order LPC coefficients is used in nearly all narrowband speech coder standards, with bit rates of less than 16 kbps, such as:

- The ITU-T G.729 CS-ACELP coder, at 8 kbps [Kata95].
- The ITU-T G.723.1, dual rate speech coder for multimedia, at 5.3 /6.3 kbps [ITUT96].
- The GSM 6.60, enhanced full rate coder, at 13 kbps [Järv97].
- The TIA IS-96, North-American standard for CDMA cellular telephony, variable rate QCELP [Gard93].
- The TIA IS-641, enhanced full rate coder for North-American TDMA cellular telephony, at 7.4 kbps [Honk97].
- The Japanese half-rate personal digital cellular standard [Ohya94].
- The US DoD Federal Standard for secure telephony, FS1016 CELP coder at 4.8 kbps[Fede91].
- The new US Federal Standard for secure telephony, MELP coder, at 2.4 kbps [Supp97].

Older standard coders, such as the GSM 6.10 [ETSI92] and the IS-54 [Gers91], use reflection coefficients and LAR to quantize spectral information. These coders will be replaced by newer standards which use LSP representation, respectively the GSM 6.60 and the IS-641. In these new standards, LSP representation allows more efficient quantization of the spectral

information, with less bits and better speech quality. The bit rate saving is used to improve speech quality, through a better representation of other coder parameters and allocation of more bits to error protection.

All the CELP speech coders found in recent publications [CELP97] use LSP representation of 10-th order LPC. The fixed rate coders have a bit rate ranging from 4 to 12.2 kbps and the variable rate coders have an average bit-rate ranging from 3 to 7 kbps. Also, LSP representation of 10-th order LPC is used in some emerging very low bit rate coders, at bit rates of about 2.4 kbps (fixed) and 1.2 kbps (variable) [LOWB97].

Definition of LSP Parameters

The starting point for deriving the LSP parameters is the LPC analysis filter of order p, $A_p(z)$, given in Equation (5.4).

A symmetrical polynomial $P_p(z)$ and an antisymmetrical polynomial $Q_p(z)$ are formed by adding and subtracting to $A_p(z)$ its time reversed system function $z^{-(p+1)}A_p(z^{-1})$. If p is even, $P_p(z)$ and $Q_p(z)$ have a zero at z = –1 and z = +1, respectively:

$$\begin{aligned} P_p(z) &= A_p(z) + z^{-(p+1)} A_p(z^{-1}) = (1 + z^{-1}) \cdot P'_p(z) \\ Q_p(z) &= A_p(z) - z^{-(p+1)} A_p(z^{-1}) = (1 - z^{-1}) \cdot Q'_p(z) \end{aligned} \tag{5.24}$$

The polynomials $P'_p(z)$ and $Q'_p(z)$ are symmetrical, and have the following properties, which are proved in [Soon84]:

- If the roots of $A_p(z)$ are inside the unit circle, then the roots of $P'_p(z)$ and $Q'_p(z)$ lie on the unit circle and are interlaced, starting with a root of $P'_p(z)$.
- Conversely, if the roots of $P'_p(z)$ and $Q'_p(z)$ lie on the unit circle and are interlaced starting with a root of $P'_p(z)$, the roots of $A_p(z)$ are inside the unit circle.

The first property is referred to as the analysis theorem or ordering property. The second property is called the synthesis theorem, and is used to ensure stability of the LPC synthesis filter $H_p(z)$ upon quantization.

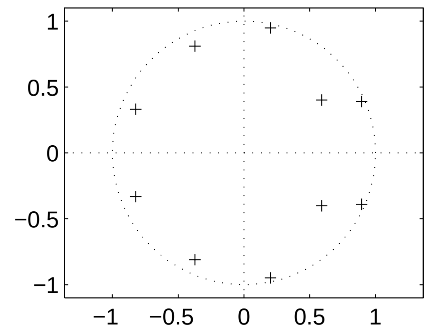


Figure 5.1: Position of the zeros of the 10-th order LPC analysis filter, $A_{10}(z)$, for a 30 ms segment of the vowel /æ/.

Given that the roots of $P'_p(z)$ and $Q'_p(z)$ lie on the unit circle, the polynomials $P'_p(z)$ and $Q'_p(z)$ can be completely specified by the angular positions of their roots. Furthermore, since $P'_p(z)$ and $Q'_p(z)$ have real coefficients, their roots occur in complex conjugate pairs. Hence, only the angular positions of the roots located on the upper semicircle of the z-plane are necessary to completely specify $P'_p(z)$ and $Q'_p(z)$:

The LSPs are defined as the angular positions of the roots of $P'_p(z)$ and $Q'_p(z)$ located on the upper semicircle of the z-plane.

Hereafter, the LSPs are denoted as $\{\omega_i\}$, in the angular frequency domain, and their ordering property is expressed as:

$$0 < \omega_1 < \omega_2 < \ldots < \omega_p < \pi \tag{5.25}$$

The odd-suffixed LSPs correspond to roots of $P'_p(z)$ while the even-suffixed LSPs correspond to roots of $Q'_p(z)$. Other notations that will be used are $\{f_i\}$, in the normalized frequency domain, in which $f_i = \omega_i/(2\pi)$, and $\{x_i\}$, in the “x-domain”, in which $x_i = \cos(\omega_i)$.

For a segment of the vowel /æ/, the location of the zeros of $A_{10}(z)$ is shown in Figure 5.1, and the location of the zeros of the polynomials $P'_{10}(z)$ and $Q'_{10}(z)$ is shown in Figure 5.2. The LPC power spectrum and position of the associated LSP parameters are shown in Figure 5.3.

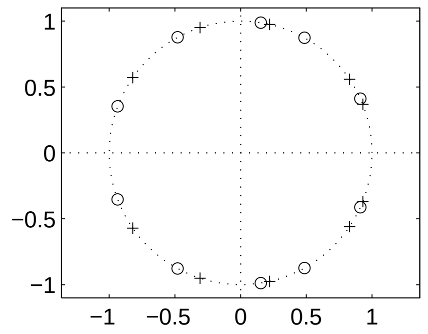


Figure 5.2: Position of the zeros of $P'_{10}(z)$ and $Q'_{10}(z)$, for a 30 ms segment of the vowel /æ/. The zeros of $P'_{10}(z)$ and $Q'_{10}(z)$ are denoted by ‘+’ and ‘o’, respectively.

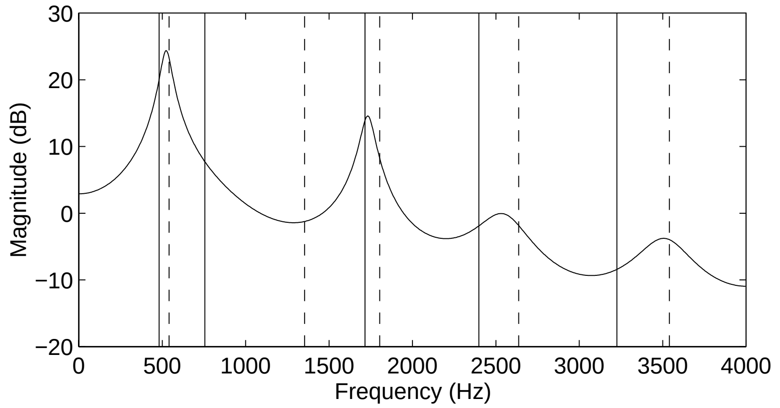


Figure 5.3: LPC power spectrum and position of the corresponding LSP parameters, for a 30 ms segment of the vowel /æ/. Odd-suffixed LSPs, corresponding to zeros of $P'_{10}(z)$, are plotted with a continuous line, while even-suffixed LSPs, corresponding to zeros of $Q'_{10}(z)$, are plotted with a dashed line.

5.7. Characteristics of the LSP Parameters

Some good quantization properties of LSP representation such as bounded range and simple check of filter stability were already mentioned in Section 5.6. Other advantageous properties of LSP parameters are their intra-frame and inter-frame correlation, their localized spectral sensitivity, their close relationship with the perceptually important peaks of the speech spectral envelope, and the fact that LSP parameters constitute a "frequency-domain" representation.

Frequency Domain Representation

LSP representation is a frequency-domain representation of the speech spectral envelope, as opposite to RC, LAR, and IS, which are temporal parameters [Kond94]. Thus, LSP-based quantization schemes can easily incorporate spectral features known to be important for human perception. An example is given in [Pali93], where lower frequency LSPs are quantized more accurately than higher frequency LSPs, as human hear resolve better the differences at lower frequencies.

Intra- and Inter-frame Correlation

A very important property of LSP parameters is their natural ordering, which is given in Equation (5.25) and can be observed in Figure 5.4. This ordering property is not only used to warrant stability of the LPC synthesis filter upon quantization, but also to speed up the calculation of LSP parameters (see § 5.9). The ordering property also indicates that the LSPs within a frame are correlated. This high correlation between neighboring LSPs is shown in [Kond94] and is called *intra-frame correlation*.

Due to the slow changes in the configuration of the vocal tract, there exists also a strong correlation between LSPs of adjacent frames, which is called *inter-frame correlation*.

Both intra- and inter-frame correlation can be successfully exploited for efficient quantization of LSP parameters.

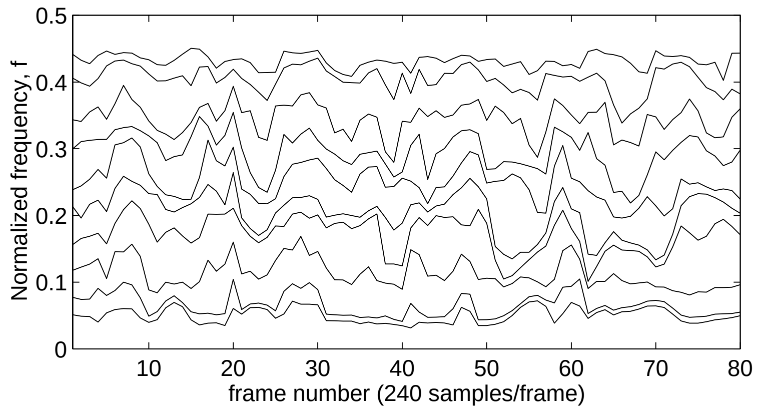


Figure 5.4: LSP trajectories for the sentence "She had your dark suit in greasy wash water all year" output by a male speaker.

Localized Spectral Sensitivity

The spectral sensitivity of each LSP is localized [Pali93], that is, a perturbation in a given LSP produces a change in the LPC power spectrum only in the neighborhood of this LSP frequency. Thus, each LSP can be individually quantized without the leakage of quantization distortion from one spectral region to the other. Note that the other LPC representations such as RC, LAR and IS, do not have this advantage as their spectral sensitivities are not localized.

Close Relationship with Formants of the Spectral Envelope

In [Soon93], it is shown that LSP frequencies display a cluster pattern around the peaks of the spectral envelope. This can also be observed in Figure 5.3. A cluster of (2 to 3) LSPs characterizes a formant frequency and the bandwidth of the given formant depends on the closeness of these LSPs.

As formants are very important for human ear perception, this property can be effectively used in LSP quantization. This is done through the use of an appropriate weighted LSP distance measure, which ensures a better quantization of the LSPs in the formant regions [Pali93]. This property also provides a strong justification to the use of LSP differences in quantization schemes [Soon93].

5.8. Quantization of the LSP Parameters

LSP is the most widely used representation for quantization of spectral information as it can be seen in [SPEC96], [SPEC95a] and [SPEC95b]. In this section, a brief overview of LSP-based spectral quantization techniques is given. The discussion is restricted to narrowband speech sampled at 8 kHz, 10-th order LPC analysis and frame length of 20-30 ms, as these are the conditions that characterize nearly all the systems used in recently proposed spectral quantization methods. The methods are evaluated in terms of spectral distortion measure and number of bits needed to achieve transparent quantization (see § 5.4). As the results reported in the literature do not use the same speech database, the meaningfulness of the comparison among different methods is limited. An attempt to compare several methods using the same database is done in [Pali95a].

Scalar Quantization

The localized spectral sensitivity property of the LSPs makes them ideal for scalar quantization. Each LSP is quantized separately, with a different quantizer. In practice, non uniform bit allocation and quantizers with non-uniform quantization levels are used, since they result in less quantization distortion than uniform quantizers. In [Pali95a], the quantizers are designed using the Lloyd algorithm, and transparent quantization is achieved with 34 bits/frame, using either LSP or LAR representation, but LSP-based quantization has a smaller percentage of outlier frames.

Quantization of the differences between adjacent LSPs (DLSPs) instead of the LSPs themselves, is used to exploit intra-frame correlation. The DLSPs also exhibit less variability across speakers and recording conditions [Soon93]. Using DLSPs, 32 bits/frame are needed for transparent quantization [Pali95a]. A

similar result is obtained with the method proposed in [Soon93], in which DLSPs are quantized with an optimum quantizer, designed taking into account both the statistical distribution and the spectral sensitivity of each DLSP. Although DLSP-based quantization allows a saving of 2 bits/frame, it is more sensitive to channel errors than LSP-based quantization. Thus, in practice, most speech coder systems, such as the FS1016 CELP coder [Fede91], use LSPs instead of DLSPs.

Vector Quantization

Vector quantization can effectively exploit the intra-frame correlation of LSP parameters resulting in smaller quantization distortion than scalar quantization at the same bit rate [Gers92].

In [Pali95a] an informal estimate suggests that the bound for transparent quantization is about 20 bits. In this case, direct full codebook search (see § 2.9) would need a codebook containing more than one million codevectors (2^{20}), of dimension of 10. This would require a prohibitively large amount of training data, and the training process would need too much time. Furthermore, the storage and computational requirements for vector quantization would be prohibitively high. The storage requirement and computational complexity of direct VQ can be reduced, at the cost of reduced performance, using various forms of suboptimal VQ, such as split vector quantization [Pali93] and multistage vector quantization [Lebl93]. The complexity of the search can be reduced even further by using tree-structured [Pham90] or classified VQ [Gers92].

In [Pali95a] some of these suboptimal methods are implemented and tested using the same speech database. Slightly more than 26 bits/frame are needed to achieve transparent quantization using multistage VQ, while 26 bits/frame are sufficient with split VQ. If a weighted LSP distance measure is used instead of euclidean measure, 25 bits/frame are needed with multistage VQ, and 24 bits/frame with split VQ. This weighted LSP distance measure exploits the close relationship between LSPs and formants of the spectral envelope giving more weight to LSPs corresponding to sharp formants than LSPs corresponding to broad formants, and the lowest weight to LSPs corresponding to spectral valleys [Soon93]. Transparent quantization at 23 bits/frame is achieved using linked split VQ as proposed in [Kim96], where the ordering property of the LSP parameters is used to improve the performance of split VQ.

As it was already mentioned, VQ schemes exploit the intra-frame correlation of LSP parameters. To further exploit the inter-frame correlation, predictive VQ can be used, with either moving average (MA) or autoregressive (AR) predictors. In [Skog97], 21 bits/frame are used to achieve transparent quantization, using either first order AR prediction, or third order MA prediction. Although AR prediction performs better than MA, the latter is more robust to channel error conditions.

Transparent quantization can thus be achieved with 21 to 26 bits/frame using VQ, and with at least 32 bits using scalar quantization. Nevertheless scalar quantization is sometimes preferred because is computationally less expensive, more robust against variations of speakers and environments, and can be protected more efficiently against channel errors [Rama95].

Spectral Quantization in the FS1016 CELP Coder

In the FS1016 CELP coder the LSP coefficients are quantized using 34-bit scalar quantization, according to the bit pattern (3,4,4,4,4,3,3,3,3,3) and using the non-uniform quantization levels given in Figure 5.5. This spectral quantizer was tested in [Lebl93] using the TIMIT speech database [Garo90]. The average spectral distortion was 1.48 dB, with 11.4 % of outliers between 2-4 dB. Although the conditions to assure transparent quantization are not fulfilled, the quantizer performed better in subjective evaluation than a 24-bit VQ quantizer with 1.17 dB of average spectral distortion and 2.12 % of outliers between 2-4 dB [Lebl93]. Good communications quality is obtained using this quantizer in the CELP FS1016 speech coder [Fede91].

LSP1

index	value
0	0.0125
1	0.0213
2	0.0281
3	0.0312
4	0.0350
5	0.0425
6	0.0525
7	0.0625

LSP2

index	value
0	0.0262
1	0.0294
2	0.0331
3	0.0369
4	0.0406
5	0.0450
6	0.0500
7	0.0550
8	0.0600
9	0.0650
10	0.0700
11	0.0762
12	0.0838
13	0.0925
14	0.1013
15	0.1100

LSP3

index	value
0	0.0525
1	0.0575
2	0.0625
3	0.0675
4	0.0731
5	0.0800
6	0.0881
7	0.0969
8	0.1062
9	0.1188
10	0.1312
11	0.1438
12	0.1562
13	0.1688
14	0.1812
15	0.1938

LSP4

index	value
0	0.0775
1	0.0825
2	0.0900
3	0.0994
4	0.1100
5	0.1212
6	0.1350
7	0.1462
8	0.1588
9	0.1713
10	0.1838
11	0.1962
12	0.2088
13	0.2212
14	0.2338
15	0.2462

LSP5

index	value
0	0.1250
1	0.1312
2	0.1412
3	0.1512
4	0.1606
5	0.1688
6	0.1788
7	0.1888
8	0.1988
9	**0.2088**
10	0.2188
11	0.2312
12	**0.2438**
13	0.2562
14	**0.2688**
15	0.2812

LSP6

index	value
0	0.1838
1	0.1962
2	0.2112
3	0.2288
4	0.2500
5	0.2750
6	0.3000
7	0.3250

LSP7

index	value
0	0.2250
1	0.2350
2	0.2450
3	0.2625
4	0.2875
5	0.3100
6	0.3375
7	0.3625

LSP8

index	value
0	0.2781
1	0.3000
2	0.3156
3	0.3312
4	0.3500
5	0.3688
6	0.3938
7	0.4188

LSP9

index	value
0	0.3450
1	0.3600
2	0.3750
3	0.3875
4	0.4000
5	0.4138
6	0.4288
7	0.4438

LSP10

index	value
0	0.3988
1	0.4088
2	0.4188
3	0.4275
4	0.4362
5	0.4488
6	0.4638
7	0.4788

Figure 5.5: Non-uniform quantization levels of the 34-bit scalar quantizer used in the CELP FS1016 speech coder.

5.9. Determination of the LSP Parameters

In order to determine the LSP parameters, the roots of $P'_{10}(z)$ and $Q'_{10}(z)$, given in Equation (5.24), have to be found. The direct solution of the equations $P'_{10}(z) = 0$ and $Q'_{10}(z) = 0$, using a numerical method such as Newton-Raphson [Sait85] is computationally expensive, as it involves the solution of two 10-th order polynomials using complex arithmetic.

The methods proposed in [Kaba86], [Saou92] and [Chan91], are more suitable for efficient real-time implementation and are explained in this section. Also, other methods which are based on discrete Fourier or cosine transform, as well as an adaptive method are briefly explained at the end of this section.

Kabal's Method

In Kabal's method [Kaba86], the symmetry of the polynomials $P'_{10}(z)$ and $Q'_{10}(z)$ is used to group their terms:

$$\begin{aligned} P'_{10}(z) &= z^{-5} \cdot \left[(z^{+5} + z^{-5}) + p'_1(z^{+4} + z^{-4}) + \ldots + p'_5\right] \\ Q'_{10}(z) &= z^{-5} \cdot \left[(z^{+5} + z^{-5}) + q'_1(z^{+4} + z^{-4}) + \ldots + q'_5\right] \end{aligned} \qquad (5.26)$$

where:

$$\begin{aligned} p'_1 &= a_{10}(1) + a_{10}(10) - 1, & q'_1 &= a_{10}(1) - a_{10}(10) + 1 \\ p'_2 &= a_{10}(2) + a_{10}(9) - p'_1, & q'_2 &= a_{10}(2) - a_{10}(9) + q'_1 \\ p'_3 &= a_{10}(3) + a_{10}(8) - p'_2, & q'_3 &= a_{10}(3) - a_{10}(8) + q'_2 \\ p'_4 &= a_{10}(4) + a_{10}(7) - p'_3, & q'_4 &= a_{10}(4) - a_{10}(7) + q'_3 \\ p'_5 &= a_{10}(5) + a_{10}(6) - p'_4, & q'_5 &= a_{10}(5) - a_{10}(6) + q'_4 \end{aligned} \qquad (5.27)$$

here $a_{10}(k)$ are the 10-th order LPC coefficients. Polynomial division, which only needs additions and subtractions, was done on the coefficients of $P_{10}(z)$ and $Q_{10}(z)$, to remove the trivial zeros at $z = \pm 1$. Evaluating $P'_{10}(z)$ and $Q'_{10}(z)$ on the unit circle, $z = e^{j\omega}$, and removing the linear phase term, $e^{-5j\omega}$ and the factor of 2, gives:

$$\begin{aligned} P'_{10}(\omega) &= \cos(5\omega) + p'_1\cos(4\omega) + \ldots + p'_4\cos(\omega) + \frac{p'_5}{2} \\ Q'_{10}(\omega) &= \cos(5\omega) + q'_1\cos(4\omega) + \ldots + q'_4\cos(\omega) + \frac{q'_5}{2} \end{aligned} \qquad (5.28)$$

The Chebyshev polynomials of first kind are given by:

$$\begin{aligned} T_n(x) &= \cos(n\omega) = 2xT_{n-1}(x) - T_{n-2}(x) \\ T_0(x) &= \cos(0) = 1 \\ T_1(x) &= \cos(\omega) = x \\ T_2(x) &= \cos(2\omega) = 2x^2 - 1 \\ T_3(x) &= \cos(3\omega) = 4x^3 - 3x \\ T_4(x) &= \cos(4\omega) = 8x^4 - 8x^2 + 1 \\ T_5(x) &= \cos(5\omega) = 16x^5 - 20x^3 + 5x \end{aligned} \qquad (5.29)$$

applying the mapping $x = \cos(\omega)$ to Equation (5.28) and using the Chebyshev polynomials of first kind, two polynomials of 5-th order, $P'_{10}(x)$ and $Q'_{10}(x)$, are obtained:

$$\begin{aligned} P'_{10}(x) = {} & 16x^5 + 8p'_1x^4 + (4p'_2 - 20)x^3 + (2p'_3 - 8p'_1)x^2 + \\ & +(5 - 3p'_2 + p'_4)x + (p'_1 - p'_3 + 0.5\,p'_5) \\ Q'_{10}(x) = {} & 16x^5 + 8q'_1x^4 + (4q'_2 - 20)x^3 + (2q'_3 - 8q'_1)x^2 + \\ & +(5 - 3q'_2 + q'_4)x + (q'_1 - q'_3 + 0.5\,q'_5) \end{aligned} \qquad (5.30)$$

The roots of $P'_{10}(x)$ and $Q'_{10}(x)$ are the LSPs in the "x-domain", $\{x_i\}$, with $x_i = \cos(\omega_i)$. Equation (5.25) gives:

$$+1 > x_1 > x_2 > \ldots > x_{10} > -1 \qquad (5.31)$$

An example of the behavior of the functions $P'_{10}(x)$ and $Q'_{10}(x)$ is shown in Figure 5.6. As $P'_{10}(x)$ and $Q'_{10}(x)$ are 5-th order polynomials, their roots cannot be calculated in a closed form. In the numerical solution proposed in [Kaba86], zero crossings are searched starting at $x = +1$, with decrements of $\Delta = 0.02$. Once an interval containing a zero crossing is found, the position of the root is refined, first by using four successive bisections, and then by doing linear interpolation. Giving the ordering property of the roots, the search is done alternatively on $P'_{10}(x)$ and $Q'_{10}(x)$, starting from the position of the last root

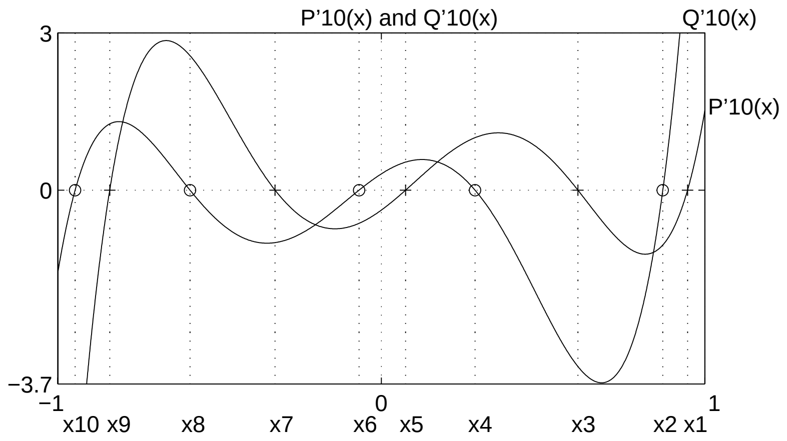


Figure 5.6: Behavior of the functions $P'_{10}(x)$ and $Q'_{10}(x)$ (x_1 to x_{10} are the LSPs in the “x-domain”, in which $x = \cos(\omega)$).

that was found. For a 10-th order LPC system, a maximum of 150 polynomial evaluations is needed [Kaba86].

The grid separation of $\Delta = 0.02$, and the number of bisections of 4, are chosen to avoid missing zero crossings in the search. These values are based on the minimum difference between roots found on 10 s of speech, sampled at 8 kHz [Kaba86].

An efficient recursive evaluation of $P'_{10}(x)$ and $Q'_{10}(x)$ for a given value of x, which uses the coefficients $\{q'_i\}$ and $\{p'_i\}$ defined in Equation (5.27), is also proposed in [Kaba86]:

$$\begin{aligned}
&b_7(x) = 0, \quad b_6(x) = 0, \quad b_5(x) = 1\\
&b_4(x) = 2x + p'_1\\
&b_3(x) = 2xb_4(x) - b_5(x) + p'_2\\
&b_2(x) = 2xb_3(x) - b_4(x) + p'_3\\
&b_1(x) = 2xb_2(x) - b_3(x) + p'_4\\
&P'_{10}(x) = xb_1(x) - b_2(x) + 0.5 \cdot p'_5
\end{aligned} \tag{5.32}$$

The evaluation for $Q'_{10}(x)$ is similar, using the coefficients $\{q'_i\}$ instead of $\{p'_i\}$. Then, the expansion on powers of x given in Equation (5.30) is not necessary. The computational cost of this recursive evaluation is 4 multiplications and 9 additions.

Saoudi’s Method

In Saoudi’s method [Saou92], two new real functions are derived from the symmetrical and the antisymmetrical polynomials, $P_{10}(z)$ and $Q_{10}(z)$, given in Equation (5.24). These functions are shown to obey a three-term recurrence relation, which leads to the following tridiagonal matrices:

$$\mathbf{M}_5 = \begin{bmatrix} 2\alpha_1+\alpha_2-2 & 1 & 0 & 0 & 0\\ \alpha_2\alpha_3 & \alpha_3+\alpha_4-2 & 1 & 0 & 0\\ 0 & \alpha_4\alpha_5 & \alpha_5+\alpha_6-2 & 1 & 0\\ 0 & 0 & \alpha_6\alpha_7 & \alpha_7+\alpha_8-2 & 1\\ 0 & 0 & 0 & \alpha_8\alpha_9 & \alpha_9+\alpha_{10}-2 \end{bmatrix}$$

$$\mathbf{M}_5^* = \begin{bmatrix} \alpha_2^*-2 & 1 & 0 & 0 & 0\\ \alpha_2^*\alpha_3^* & \alpha_3^*-\alpha_4^*-2 & 1 & 0 & 0\\ 0 & \alpha_4^*\alpha_5^* & \alpha_5^*-\alpha_6^*-2 & 1 & 0\\ 0 & 0 & \alpha_6^*\alpha_7^* & \alpha_7^*-\alpha_8^*-2 & 1\\ 0 & 0 & 0 & \alpha_8^*\alpha_9^* & \alpha_9^*-\alpha_{10}^*-2 \end{bmatrix} \tag{5.33}$$

the values α_m^* and α_m are obtained by using the antisymmetric split Levinson recursion, which is described as follows:

$$\begin{aligned}
&P_0^*(z) = 0, \quad P_1^*(z) = 1 - z^{-1}, \quad p^*_{m,0} = 1 \quad \text{for } m \ge 1\\
&\lambda_0^* = 1, \quad \tau_0^* = r_0\\
&\text{for } \ 1 \le m \le 10:\\
&\tau_m^* = \begin{cases} \sum_{i=0}^{t} (r_i - r_{m-i})p^*_{m,i} & \text{for } m = 2t+1\\ \sum_{i=0}^{t-1} (r_i - r_{m-i})p^*_{m,i} + r_t p^*_{m,t} & \text{for } m = 2t \end{cases}\\
&\alpha_m^* = \frac{\tau_m^*}{\tau_{m-1}^*}, \qquad \lambda_m^* = 2 - \frac{\alpha_m^*}{\lambda_{m-1}^*}, \qquad \alpha_m = \lambda_m^*(2 - \lambda_{m-1}^*)\\
&P_{m+1}^*(z) = (1+z^{-1})P_m^*(z) - \alpha_m^* z^{-1} P_{m-1}^*(z)
\end{aligned} \tag{5.34}$$

here r_k is the *k*-th autocorrelation coefficient given in Equation (5.10). As the polynomials $P_m^*(z)$ are antisymmetrical, only half of their coefficients are calculated. For an order of 10, the computational cost of the antisymmetric split Levinson recursion is 58 multiplications, 122 additions, and 20 divisions. This recursion is used instead of the Levinson-Durbin recursion of Equation (5.12).

The eigenvalues of M_5 and M_5^* correspond, respectively, to the odd- and even-suffixed LSPs in the “x-domain”, except for a gain factor of 2, thus, they will be denoted as $\{\lambda_i\}$, with $\lambda_i = 2\cos(\omega_i)$, and the ordering property expressed as:

$$+2 > \lambda_1 > \lambda_2 > \ldots > \lambda_{10} > -2 \tag{5.35}$$

Different methods to compute the eigenvalues of the tridiagonal matrices are compared in [Saou92], and the bisection method is chosen, with a number of 8 bisections, as the one with minimum complexity for the application. This method is briefly explained in the next paragraphs.

The eigenvalues of M_5 and M_5^* are the roots of their characteristic polynomials, which are given by:

$$\begin{aligned} L_5(x) &= \left|M_5 - xI_5\right| = 0 \\ L_5^*(x) &= \left|M_5^* - xI_5\right| = 0 \end{aligned} \tag{5.36}$$

where I_5 is the identity matrix of 5 elements. Due to the tridiagonal form of M_5 and M_5^*, their characteristic polynomials obey the following recursions [Acto90]:

$$\begin{aligned} L_0(x) &= 1 \\ L_1(x) &= (d(0) - x) \\ L_2(x) &= (d(1) - x)L_1(x) - e(1)\cdot L_0(x) \\ L_3(x) &= (d(2) - x)L_2(x) - e(2)\cdot L_1(x) \\ L_4(x) &= (d(3) - x)L_3(x) - e(3)\cdot L_2(x) \\ L_5(x) &= (d(4) - x)L_4(x) - e(4)\cdot L_3(x) \end{aligned} \tag{5.37}$$

$$\begin{aligned} L_0^*(x) &= 1 \\ L_1^*(x) &= (d^*(0) - x) \\ L_2^*(x) &= (d^*(1) - x)L_1^*(x) - e^*(1)\cdot L_0^*(x) \\ L_3^*(x) &= (d^*(2) - x)L_2^*(x) - e^*(2)\cdot L_1^*(x) \\ L_4^*(x) &= (d^*(3) - x)L_3^*(x) - e^*(3)\cdot L_2^*(x) \\ L_5^*(x) &= (d^*(4) - x)L_4^*(x) - e^*(4)\cdot L_3^*(x) \end{aligned} \tag{5.38}$$

where d(k) and d*(k) are respectively the diagonal elements of M_5 and M_5^*, and e(k) and e*(k) are the elements below the diagonal. The sequence of polynomials $L_n(x)$ is a Sturmanian sequence [Acto90], thus, for a given value of $x = \gamma$, the number of sign changes in the numerical sequence $\{L_0(\gamma),\ldots,L_5(\gamma)\}$ gives the number of roots of $L_5(x)$ which are smaller than γ. This property holds also for $L_5^*(x)$, and is used, together with the ordering property of Equation (5.35), to search each LSP independently. If an odd LSP is searched, the evaluation given in Equation (5.37) is used, while the evaluation given in Equation (5.38) is used for even LSPs.

Each LSP is iteratively approximated from below. The approximation value is initialized at –2, which is smaller than the searched LSP. The addition value is initialized at 4, and is halved at every step and added to the current approximation value to obtain the trial value. The trial value is used with either Equation (5.37) or (5.38). The number of sign changes in the obtained sequence corresponds to the number of roots between the trial value and -2. Thus, it is possible to know if the trial value is smaller than the searched LSP, in which case the trial value is accepted, and becomes the current approximation value for the next iteration.

In order to search the 10 LSP parameters, 80 of the evaluations given either in Equation (5.37) or (5.38) are needed. The computational cost of each evaluation is 9 additions and 8 multiplications.

Chan's Method

In Chan's method [Chan91], the LSP parameters $\{\omega_i\}$ are computed directly from the reflection coefficients $\{k_m\}$. Thus, there is no need to calculate the LPC coefficients. The advantage of using reflection coefficients is that they are bounded in magnitude by 1 and thus can be computed entirely on fixed-point arithmetic, using the LeRoux-Gueguen algorithm [Lero77], which is given in Appendix B.

Chan's method is based on the use of the auxiliary function:

$$\psi_m(z) = z^{\frac{m+1}{2}} \cdot A_m(z) \tag{5.39}$$

evaluating this function on the unit circle, $z = e^{j\omega}$, gives:

$$\psi_{10}(e^{j\omega}) = e^{\frac{j\cdot 11\cdot\omega}{2}} A_{10}(e^{j\omega}) = \mathrm{Re}\left[\psi_{10}(e^{j\omega})\right] + j\cdot\mathrm{Im}\left[\psi_{10}(e^{j\omega})\right] \tag{5.40}$$

the symmetrical and antisymmetrical polynomials, $P_p(z)$ and $Q_p(z)$, given in Equation (5.24) can be expressed as:

$$\begin{aligned} P_{10}(e^{j\omega}) &= A_{10}(e^{j\omega}) + e^{-j\cdot 11\cdot\omega} A_{10}(e^{-j\omega}) = e^{-j\frac{11}{2}\omega} \cdot 2\,\mathrm{Re}\left[\psi_{10}(e^{j\omega})\right] \\ Q_{10}(e^{j\omega}) &= A_{10}(e^{j\omega}) - e^{-j\cdot 11\cdot\omega} A_{10}(e^{-j\omega}) = e^{-j\frac{11}{2}\omega} \cdot 2j\,\mathrm{Im}\left[\psi_{10}(e^{j\omega})\right] \end{aligned} \tag{5.41}$$

Thus, the zero crossings of $\mathrm{Re}[\psi_{10}(e^{j\omega})]$ and $\mathrm{Im}[\psi_{10}(e^{j\omega})]$ correspond to the odd- and even-suffixed LSPs, respectively.

From Equation (5.19) it is seen that the polynomial $A_m(z)$ obeys to the following recursion:

$$A_m(z) = A_{m-1}(z) + k_m z^{-m} A_{m-1}(z^{-1}), \quad 1 \le m \le p \tag{5.42}$$

Using Equation (5.39) and (5.42) the following recursive evaluation is obtained:

$$\begin{aligned} \mathbf{Y}_0(\omega) &= \mathbf{r}\left(\frac{\omega}{2}\right)\cdot\begin{pmatrix}1\\0\end{pmatrix} \\ \mathbf{Y}_m(\omega) &= \mathbf{r}\left(\frac{\omega}{2}\right)\cdot\mathbf{K}_m\cdot\mathbf{Y}_{m-1}(\omega) \end{aligned} \tag{5.43}$$

where K_m is given in Equation (5.46) and:

$$\mathbf{Y}_m(\omega) = \begin{bmatrix}\mathrm{Re}\{\psi_m(\omega)\}\\ \mathrm{Im}\{\psi_m(\omega)\}\end{bmatrix}, \quad \mathbf{r}\left(\frac{\omega}{2}\right) = \begin{bmatrix}\cos\left(\frac{\omega}{2}\right) & -\sin\left(\frac{\omega}{2}\right)\\ \sin\left(\frac{\omega}{2}\right) & \cos\left(\frac{\omega}{2}\right)\end{bmatrix} \tag{5.44}$$

this recursion can be rearranged to decrease the computational complexity, obtaining the following formula:

$$\mathbf{Y}_{10}(\omega) = \mathbf{R}_{10}(\omega)\cdot\mathbf{K}_9 \ldots \mathbf{R}_4(\omega)\cdot\mathbf{K}_3\cdot\mathbf{R}_2(\omega)\cdot\mathbf{K}_1\cdot\mathbf{r}\left(\frac{\omega}{2}\right)\cdot\begin{bmatrix}1\\0\end{bmatrix} \tag{5.45}$$

where:

$$\mathbf{R}_m(\omega) = \begin{bmatrix}\cos(\omega)+k_m & -\sin(\omega)\\ \sin(\omega) & \cos(\omega)-k_m\end{bmatrix}, \quad \mathbf{K}_m = \begin{bmatrix}1+k_m & 0\\ 0 & 1-k_m\end{bmatrix} \tag{5.46}$$

thus, 30 multiplications and 20 additions are needed per evaluation, as well as 10 additions per frame to prepare the scaling matrices $\mathbf{K}_m$.

The functions $\mathrm{Re}[\psi_{10}(e^{j\omega})]$ and $\mathrm{Im}[\psi_{10}(e^{j\omega})]$ are searched alternatively for zero-crossings, starting with $\mathrm{Re}[\psi_{10}(e^{j\omega})]$. The evaluation of $\psi_{10}(\omega)$ is done using Equation (5.45). The search is done on the range $(0,\pi)$, using a grid of 128 points, corresponding to a resolution of 0.0078π. If more accuracy is needed, a bisection technique can be used.

The total cost to search a set of LSPs, without using bisection, is 3840 multiplications and 2570 additions. This is computationally too expensive, and this algorithm was only retained for its possible advantages for a fixed-point implementation. Additionally, this algorithm requires the storage (or evaluation) of trigonometric functions.

Spectral Transform Methods

There exists several methods based on spectral transforms, either discrete cosine transform (DCT) or discrete Fourier transform (DFT), or their fast versions, fast Fourier transform (FFT) or fast DCT. One of these methods, which is proposed by Kang and Fransen [Kaba86] uses the all-pass ratio filter:

$$R_p(z) = \frac{z^{-(p+1)} A_p(z^{-1})}{A_p(z)} \qquad (5.47)$$

the phase spectrum of this filter is evaluated, and the LSPs correspond to the frequencies where the phase value is a multiple of π.

In [Soon84], $P'_{10}(\omega)$ and $Q'_{10}(\omega)$, given in Equation (5.28) are evaluated on a fine grid by using DCT. Sign changes at adjacent grid points isolate the intervals which contain a root, and further bisection of these intervals approximates the root positions.

In [Kond94], a DFT is done on the coefficients of $P'_{10}(z)$ and $Q'_{10}(z)$, given in Equation (5.24). The LSPs are the frequency location of the partial minima of the power spectrum. As the coefficients are real and symmetrical, the number of computation is reduced to 6 multiply-adds (MAC) per spectrum point. The suggested DFT size is 1024 points. Similarly, in [Kang87] a single complex fast Fourier transform is used to compute both spectrums of $P_{10}(z)$ and $Q_{10}(z)$ at once, with a transform size of 512, giving a frequency resolution of 15.625 Hz. To improve this frequency resolution, a three-point parabolic approximation is suggested. A zero crossing search using DFT with 64 to 128 points, together with linear interpolation is proposed in [Furu89].

All these proposed spectral transform methods are computationally too expensive, when compared to the methods of Kabal and Saoudi. Besides, they require calculation or storage of trigonometric functions. Thus, they are not further considered in the work described in this report.

Adaptive Methods

All the LSP calculation methods previously described require calculation of the LPC coefficients, or some equivalent parameters such as the reflection coefficients or the values α_m^* and α_m, given in Equation (5.34).

A least-mean-square adaptive method to calculate the LSP parameters directly from the speech samples is proposed in [Chee87]. The initial estimation uses evenly distributed LSP values and a new set of LSP parameters is estimated for each input speech sample.

This method is very attractive because of its low complexity, but as it is a “learning type” algorithm, outlier samples can result in adaptation errors [Kond94]. If this error occurs at the end of the frame, there is no time for correction, before the LSP set is used. Furthermore, the experience done in [Chee87] is limited to an order of p = 4, and uses synthetic speech. With an order of p = 10, and using real speech as input, the convergence behavior of the algorithm excludes its use on speech coders such as the CELP FS1016 [Fede91], therefore, this method is not considered hereafter.

5.10. LSP to LPC Transformation

The conversion of LSP parameters to LPC coefficients is less computationally expensive than the LPC to LSP conversion. The LPC analysis filter can be expressed as:

$$A_{10}(z) = \frac{P_{10}(z) + Q_{10}(z)}{2} \qquad (5.48)$$

where $P_{10}(z)$ and $Q_{10}(z)$ are the symmetrical and antisymmetrical polynomials given in Equation (5.24). These polynomials are obtained from the LSP parameters $\{\omega_i\}$ using the following relations:

$$P_{10}(z) = (1 + z^{-1}) \prod_{i=1,3,5,7,9} \left[1 - 2\cos(\omega_i) \cdot z^{-1} + z^{-2}\right]$$

$$Q_{10}(z) = (1 - z^{-1}) \prod_{i=2,4,6,8,10} \left[1 - 2\cos(\omega_i) \cdot z^{-1} + z^{-2}\right] \qquad (5.49)$$

It is important to notice that, if the LSP parameters are expressed in the "x-domain", where $x_i = \cos(\omega_i)$, as it is done in Kabal's and Saoudi's methods, the LSP to LPC conversion is eased, avoiding the calculation or storage of trigonometric functions.

Direct Expansion Method

The polynomials $P_{10}(z)$ and $Q_{10}(z)$ are found by multiplying the product terms of Equation (5.49). Then the LPC coefficients are calculated by means of Equation (5.48). This calculation is given in Appendix C.1 and has a computational cost of 62 multiplications and 92 additions.

LPC Analysis Filter Method

Equation (5.48) shows that the LPC analysis filter is the parallel combination of the filters $P_{10}(z)$ and $Q_{10}(z)$. Similarly, these filters are each the cascade combination of five second-order sections and one first-order section, corresponding to the factors of Equation (5.49). The resulting structure is shown Figure 5.7 and is used to obtain the LPC coefficients, as explained in Appendix C.2, at the cost of 30 multiplications and 70 additions.

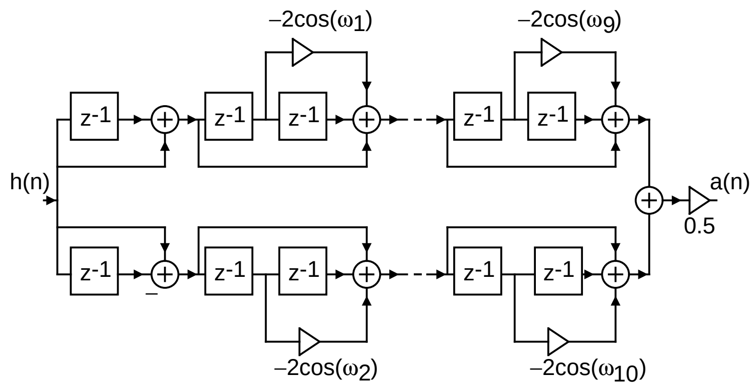


Figure 5.7: Filter used to generate the LPC coefficients, in the LPC analysis filter method. The $\{\omega_i\}$ are the LSP parameters.

Kabal's Method

In [Kaba86], an alternative reconstruction process using Chebyshev series representation is formulated. This leads to an efficient reconstruction process which takes the symmetry of the polynomials into account. This procedure is given in Appendix C.3. The computational cost is 20 multiplications and 59 additions. This is the least expensive of the three algorithms for LSP to LPC conversion described in this section. Besides, Kabal's algorithm is highly regular and numerically stable [Kaba86], which is advantageous for efficient implementation.

5.11. The CELP FS1016 Speech Coder

Code-excited linear predictive (CELP) (see § 2.9) speech coding refers to a family of speech coding algorithms which combine LPC-based analysis-by-synthesis (AbS-LPC) and vector quantization (VQ) [Gers94].

In AbS-LPC systems, the LPC synthesis model is used (see § 5.1), in which an excitation signal, e(n), is input to the LPC synthesis filter, $H_p(z)$, to yield the synthetic speech output $\hat{s}(n)$. The coefficients of the synthesis filter are determined from a frame of the speech signal, using an open-loop technique such as the autocorrelation method (see § 5.2). Once the synthesis filter is determined, an appropriate excitation signal is found by a closed-loop search. The input of the synthesis filter is varied systematically, to find the excitation signal that produces the synthesized output that best matches the speech signal, from a perceptual point of view.

Vector quantization (VQ) is combined with AbS-LPC in CELP coders [Gers94]. The optimum excitation signal is selected from a stochastic codebook of possible excitation signals (codevectors). Each codevector is passed through the synthesis filter, and the vector which produces the output that best matches the speech signal is selected.

The U.S. Federal Standard 1016, is a CELP algorithm operating at 4.8 kbps, intended primary for secure voice transmission. The block diagram of this coder is shown in Figure 5.8. This coder uses an 8 kHz sampling rate and a 30 ms frame size, with four subframes of 7.5 ms each.

Long-term correlation of the speech signal (pitch) is modeled using an adaptive codebook and the excitation signal is formed by the addition of two scaled codevectors, one selected from the stochastic codebook and one selected from the adaptive codebook. The search for the optimum codevectors and gains is done for every subframe.

The encoder generates and transmits one set of LPC coefficients per frame, and four sets of codebook indices and gains per frame (one set per subframe).

The spectral analysis block corresponds to the shadowed region in Figure 5.8. This block works on frames of 30 ms, while the rest of the encoder, containing the dictionary searches and gain selection, works on a subframe of 7.5 ms. The detailed diagram of the spectral analysis block is given in Figure 5.10 and is explained next.

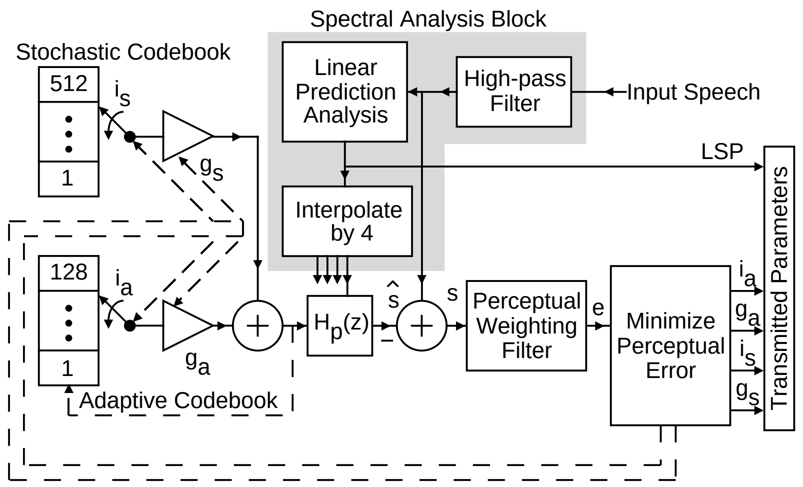


Figure 5.8: Block diagram of the CELP FS1016 speech coder.

Short-term Spectral Analysis in the CELP FS1016 Coder

The short-term linear prediction analysis is performed once per frame by open-loop, 10-th order autocorrelation analysis (see § 5.2) using a 30 ms Hamming window, no pre-emphasis, and 15 Hz bandwidth expansion, with $\gamma = 0.994$ (see § 5.3). The Hamming window is centered at the end of the last frame, as it is shown in Figure 5.9.

Besides improving speech quality, the bandwidth expansion is also beneficial for LSP quantization and for fast LSP calculation (see § 6.3).

The bandwidth expanded LPC coefficients are converted to a set of LSP parameters and quantized using the 34-bit, independent nonuniform scalar quantization tables given in Figure 5.5, as specified in [Fede91].

Two quantized sets of LSP parameters, corresponding to the window positions A and B in Figure 5.9, are used for interpolation with the weights given in Table 5.1, obtaining four sets of LSP parameters, one set for each subframe. Each of these LSP sets is converted to LPC coefficients, and used in the synthesis filter for the dictionary searches and gain selection.

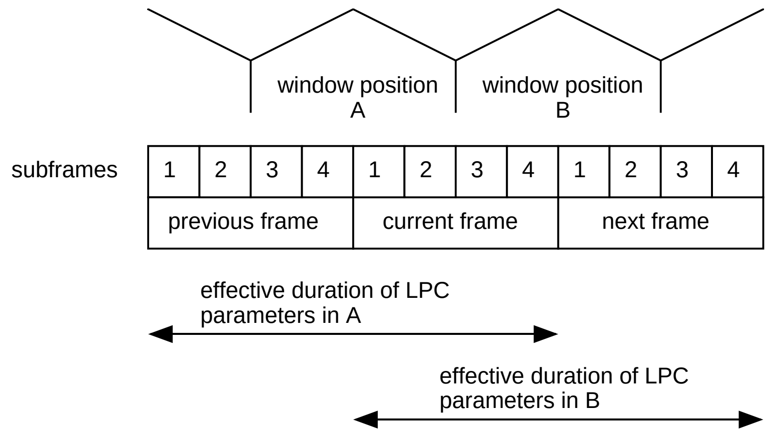


Figure 5.9: Position of the LPC analysis windows for a given frame in the CELP FS1016 speech coder.

Subframe	***LSP set A***	***LSP set B***
1	7/8	1/8
2	5/8	3/8
3	3/8	5/8
4	1/8	7/8

Table 5.1: Interpolation weights used to obtain four sets of LSP parameters from the two quantized LSP sets corresponding to the LPC analysis window position A and B.

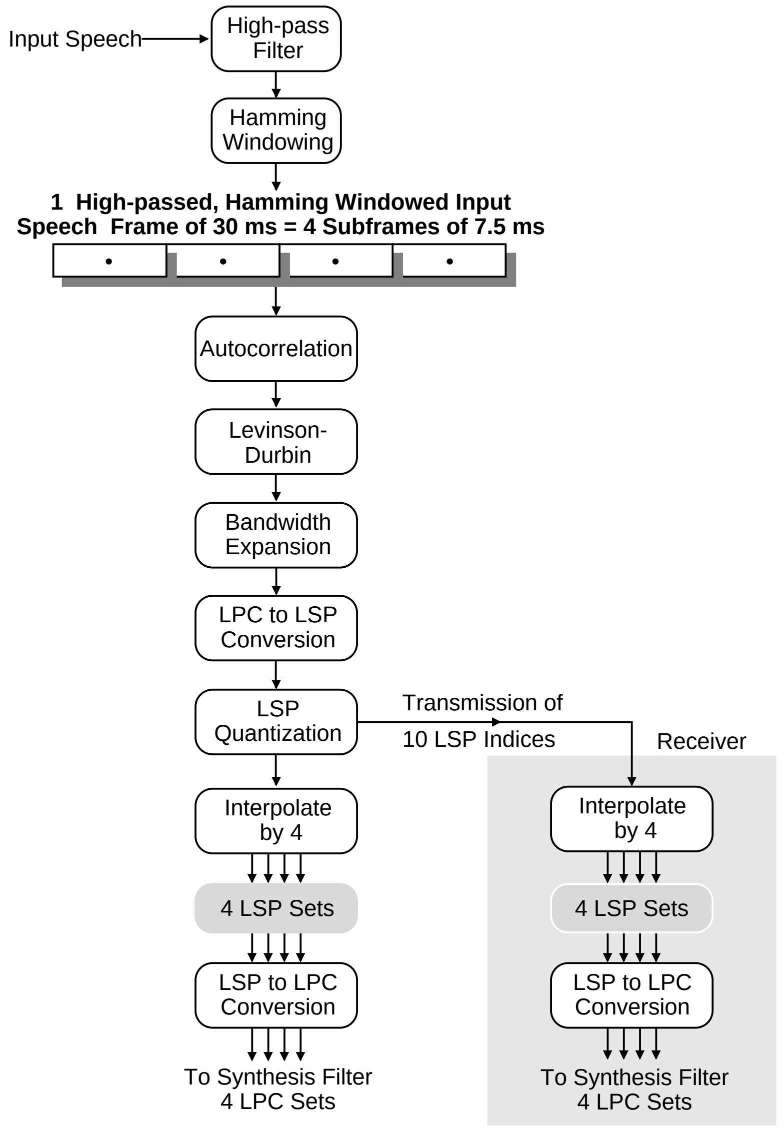


Figure 5.10: Short-term spectral analysis in the CELP FS1016 speech coder.

5.12. Summary of the Chapter

Spectral analysis and quantization for speech coding was introduced in this chapter. In particular, it was shown that LSP is the most used representation for spectral quantization and interpolation. The definition, properties, and characteristics of LSP were discussed, as well as different methods for quantization of LSP parameters.

Particular emphasis was placed on the different existing methods for LSP calculation, and the computational complexity of these methods was discussed. The methods of Kabal, Saoudi and Chan are promising for efficient real time implementation, and will be studied in the next chapter.

Finally, it was shown how spectral analysis and quantization is done in the CELP FS1016 speech coder.

These concepts will be used in Chapter 6, in which two novel algorithms for LSP calculation are presented, and in Chapter 7, where the DSP56001 optimized implementation of the CELP FS1016 spectral analysis block is given.

5.13. References

[Acto90] F. S. Acton, *Numerical Methods that Work*, Mathematical Association of America, Washington, DC, 1990.

[Atal89] B. Atal, R. Cox, and P. Kroon, "Spectral Quantization and Interpolation for CELP Coders", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'89, Vol.1, pp. 69-72, 1989.

[CELP97] ICASSP97 session: "CELP Coding", 12 different papers, *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol.2, pp. 731-778, 1997.

[Chan91] C. Chan and K. Law, "An Algorithm for Computing the LSP Frequencies Directly from the Reflection Coefficients", *Proc. European Conference on Speech Communication and Technology*, EUROSPEECH'91, pp. 913-916, 1991.

[Chee87] B. Cheetham, "Adaptive LSP Filter", *IEE Electronics Letters*, Vol. 23, No. 2, pp. 89-90, 1987.

[ETSI92] European Telecommunication Standard Institute (ETSI), “Full-rate Speech Transcoding”, Recommendation GSM 06.10, 1992.

[Fede91] "Federal Standard 1016, Telecommunications: Analog to Digital Conversion of Radio Voice by 4,800 bit/second Code Excited Linear Prediction (CELP)", National Communications Systems, Office of Technology and Standards, Washington, DC20305-2010, 1991.

[Furu89] S. Furui, *Digital Speech Processing, Synthesis, and Recognition* (Chapter 5), Dekker, New York, 1989.

[Gard93] W. Gardner et al., "QCELP: A Variable Bit Rate Speech Coder for CDMA Digital Cellular" in *Speech and Audio Coding for Wireless and Network Applications*, ed. by B. Atal, V. Cuperman, and A. Gersho, Kluwer Academic Publishers, Boston, MA, USA, 1993.

[Garo90] J. Garofolo et al., "Darpa TIMIT, Acoustic-phonetic Continuous Speech Corpus CD-ROM", National Institute of Standards and Technology, NISTIR 493, 1990.

[Gers91] I. Gerson and M. Jasiuk, "Vector Sum Excited Linear Prediction (VSELP)" in *Advances in Speech Coding*, ed. by B. Atal, V. Cuperman, and A. Gersho, Kluwer Academic Publishers, Boston, MA, USA, 1991.

[Gers92] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, 1992.

[Gers94] A. Gersho, “Advances in Speech and Audio Compression”, *Proc. of the IEEE*, Vol. 82, No. 6, 1994.

[Gray76] A. Gray and J. Markel, "Distance Measures for Speech Processing", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 24, No. 5, pp. 380-391, 1976.

[Honk97] T. Honkanen et al., "Enhanced Full Rate Speech Codec for IS-136 Digital Cellular System", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol. 2, pp. 731-734, 1997.

[Itak75] F. Itakura, "Line Spectrum Representation of Linear Predictive Coefficients of Speech Signals", *J. of the Acoustical Society of America* , Vol. 57, pp. S35, 1975.

[ITUT96] International Telecommunications Union, “Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbps”, Recommendation G.723.1, 1996.

[Järv97] K. Järvinen et al., "GSM Enhanced Full Rate Speech Codec", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol. 2, pp. 771-774, 1997.

[Kaba86] P. Kabal and P. Ramachandran, "The Computation of Line Spectral Frequencies Using Chebyshev Polynomials", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 34, No. 6, pp. 1419-1426, 1986.

[Kang87] G. Kang and L. Fransen, "Experimentation with Synthesized Speech Generated from Line Spectrum Pairs", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 35, No. 4, pp. 568-571, 1987.

[Kata95] A. Kataoka et al., "LSP and Gain Quantization for the Proposed ITU-T 8 kb/s Speech Coding Standard", *IEEE Speech Coding Workshop*, pp. 7-8, 1995.

[Kim96] M. Kim et al., "Linked Split-vector Quantizer of LPC Parameters", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'96, Vol.1, pp. 741-744, 1996.

[Kond94] A. M. Kondoz, *Digital Speech: Coding for Low Bit Rate Communication Systems* (Chapters 3, 4), Wiley, Chichester, 1994.

[Lebl93] W. LeBlanc et al., "Efficient Search and Design Procedures for Robust Multi-stage VQ of LPC Parameters for 4 kbps Speech Coding", *IEEE Trans. on Speech and Audio Processing*, Vol. 1, No. 4, pp. 373-385, 1993.

[Lero77] J. LeRoux and C. Gueguen, "A Fixed Point Computation of Partial Correlation Coefficients", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 25, No. 3, pp. 257-259, 1977.

[LOWB97] ICASSP97 session: "Speech Coding at Low Bit Rates", 14 different papers, *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol.2, pp. 1555-1610, 1997.

[Ohya94] T. Ohya et al., "5.6 kbits/s PSI-CELP of the Half Rate PDC Speech Coding Standard", *Proc. IEEE Vehicular Technology Conference*, Vol. 1, pp. 1680-1684, 1994.

[Pali93] K. Paliwal and B. Atal, "Efficient Vector Quantization of LPC Parameters at 24 bits/frame", *IEEE Trans. on Speech and Audio Processing*, Vol. 1, No. 1, pp. 3-14, 1993.

[Pali95a] K. Paliwal and W. Kleijn, "Quantization of LPC Parameters" (Chapter 12) in *Speech Coding and Synthesis*, ed. by W. Kleijn and K. Paliwal, Elsevier, Amsterdam, 1995.

[Pali95b] K. Paliwal, "Interpolation Properties of Linear Prediction Parametric Representations", *Proc. European Conference on Speech Communication and Technology*, EUROSPEECH'95, Vol. 2, pp. 1029-1032, 1995.

[Pham90] N. Phamdo and N. Farvardin, "Coding of Speech LSP Parameters Using TSVQ with Interblock Noiseless Coding", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'90 , Vol. 1, pp.193-196, 1990.

[Rama95] R. P. Ramachandran et al., "A Two Codebook Format for Robust Quantization of Line Spectral Frequencies", *IEEE Trans. on Speech and Audio Processing*, Vol. 3, No. 3, pp. 157-168, 1995.

[Sait85] S. Saito and K. Nakata, *Fundamentals of Speech Signal Processing* (Chapter 9), Academic Press, New York, 1985.

[Saou92] S. Saoudi and J. Boucher, "A New Efficient Algorithm to Compute the LSP Parameters for Speech Coding", *Signal Processing*, Elsevier, Vol. 28, No. 2 , pp. 201-212, 1992.

[Skog97] J. Skoglund and J. Lindén, "Predictive VQ for Noisy Channel Spectrum Coding: AR or MA", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol. 3, pp. 1351-1354, 1997.

[Soon84] F. Soong and B. Juang, "Line Spectrum Pair (LSP) and Speech Data Compression", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'84, pp. 1.10.1-1.10.4, 1984.

[Soon93] F. Soong and B. Juang, "Optimal Quantization of LSP Parameters", *IEEE Trans. on Speech and Audio Processing*, Vol. 1, No. 1, pp. 15-24, 1993.

[SPEC95a] ICASSP95 session: "Spectral Quantization", 10 different papers, *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'95, USA, Vol.1, pp. 716-755, 1995.

[SPEC95b] EUROSPEECH95 session: "Quantization of Spectral Parameters", 9 different papers, *Proc. European Conference on Speech Communication and Technology*, EUROSPEECH '95, Vol.2, pp. 1029-1064, 1995.

[SPEC96] ICASSP96 session: "Spectral Quantization", 10 different papers, *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'96, Vol.1, pp. 737-776, 1996.

[Supp97] L. Supplee et al., "MELP: The New Federal Standard at 2400 bps", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol.2, pp. 1591-1594, 1997.

Chapter 6 Proposed Algorithms for LSP Calculation

In this chapter two novel efficient algorithms for calculation of LSP parameters from LPC coefficients are presented. These algorithms are referred to as “Mixed LSP” and “quantized-search Kabal”. In the previous chapter, it was found that Kabal’s, Saoudi’s and Chan’s algorithms are the most promising for efficient real time implementation among the existing LSP calculation algorithms.

The proposed LSP calculation algorithms are first explained and then compared with the algorithms of Kabal, Chan and Saoudi from the point of view of accuracy, reliability and computational complexity.

Kabal’s algorithm is found to be the most efficient and accurate of the existing methods. This algorithm, as well as “Mixed LSP” and “quantized-search Kabal”, were implemented on a DSP56001 and their computational complexity in MIPS was compared.

The reader is reminded that, unless stated otherwise, an LPC order of p = 10 is assumed through this chapter.

6.1. First Proposed Method: Mixed-LSP

In order to derive the LSP parameters, the roots of $P'_{10}(z)$ and $Q'_{10}(z)$, given in Equation (5.24), have to be found. In Kabal's method (see § 5.9), the 5-th order polynomials, $P'_{10}(x)$ and $Q'_{10}(x)$, are obtained by evaluating $P'_{10}(z)$ and $Q'_{10}(z)$ on the unit circle, and applying the mapping $x = \cos(\omega)$ together with Chebyshev polynomials of first kind. The LSPs are the roots of $P'_{10}(x)$ and $Q'_{10}(x)$, and are found by a zero-crossing search on a grid of $\Delta = 0.02$, followed by four successive bisections and a final linear interpolation. The precision of the obtained LSPs is higher than required by speech coding applications, but the number of bisections cannot be decreased, or the size of the grid increased, without compromising the zero-crossing search. In this section, it is shown that five intervals, containing each only one zero-crossing of $P'_{10}(x)$ and one zero-crossing of $Q'_{10}(x)$, can be calculated, avoiding the zero-crossing search. This fact allows a trade-off between LSP precision and computational complexity [Gras97a].

Different Derivation of $P'_{10}(x)$ and $Q'_{10}(x)$

A different derivation of the polynomials $P'_{10}(x)$ and $Q'_{10}(x)$ is given in Appendix D.1. This derivation uses the auxiliary function $\Psi_m(z)$, given in Equation (5.39), and the mapping $x = \cos(\omega)$ together with Chebyshev polynomials of first and second kind. Using this derivation the polynomials $P'_{10}(x)$ and $Q'_{10}(x)$ are expressed as:

$$\begin{aligned} P'_{10}(x) &= C_{10}(x) - D_{10}(x) \\ Q'_{10}(x) &= C_{10}(x) + D_{10}(x) \end{aligned} \tag{6.1}$$

where $C_{10}(x)$ is a 5-th order polynomial, and $D_{10}(x)$ is a 4-th order polynomial, whose roots can be calculated in a closed form. The behavior of the functions $P'_{10}(x)$, $Q'_{10}(x)$, and $D_{10}(x)$ is shown in Figure 6.1, where x_1 to x_{10} are the LSPs in the "x-domain", in which $x_i = \cos(\omega_i)$, and r_1 to r_4 are the roots of $D_{10}(x)$.

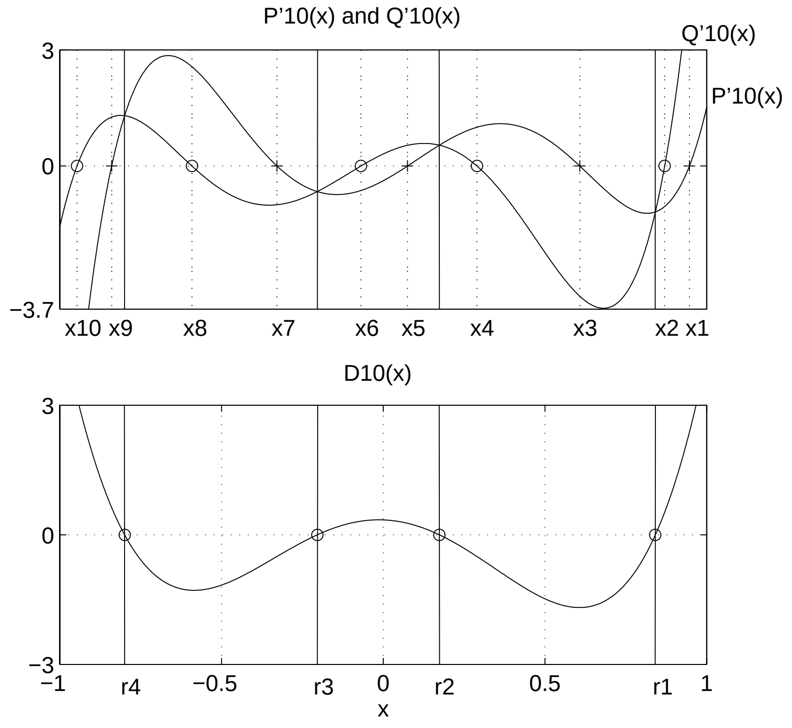


Figure 6.1: Behavior of the functions $P'_{10}(x)$, $Q'_{10}(x)$, and $D_{10}(x)$ (x_1 to x_{10} are the LSPs in the "x-domain", in which $x = \cos(\omega)$, and r_1 to r_4 are roots of $D_{10}(x)$).

In Appendix D.6, it is proved that the roots of $D_{10}(x)$ are real, different, and inside the interval (–1,+1). Furthermore, in Equation (6.1) it is seen that these roots correspond to the intersections of $P'_{10}(x)$ with $Q'_{10}(x)$.

Due to the ordering property of the LSP parameters (see § 5.6 and Equation (5.31)), when going from $x = +1$ to $x = -1$, $P'_{10}(x)$ is crossing the x-axis first at x_1, then $Q'_{10}(x)$ has its first zero-crossing at x_2. As the next LSP is x_3, $P'_{10}(x)$ and $Q'_{10}(x)$ intersect each other before crossing the x-axis at x_3 and x_4, respectively, then they intersect again before x_5, x_6, before x_7, x_8 and before x_9, x_{10}. Thus the roots of $D_{10}(x)$ divide the interval $(-1,+1)$ into five sections, *each section containing only one zero-crossing of $P'_{10}(x)$ and one zero-crossing of $Q'_{10}(x)$.*

Description of the Proposed Algorithm (Mixed-LSP)

The roots of $D_{10}(x)$ are calculated and ordered, to obtain the five intervals containing each only one zero-crossing of $P'_{10}(x)$ and one zero-crossing of $Q'_{10}(x)$. The position of these zero-crossings is refined by five successive bisections, and a final linear interpolation, similarly to Kabal's method. A total of 60 polynomial evaluations is needed. Using the efficient recursive evaluation proposed by Kabal, and given in Equation (5.32), the computational cost of a polynomial evaluation is 4 multiplications and 9 additions.

In Appendices D.4 and D.5, particular attention is paid to the optimization of the calculation and ordering of the roots of $D_{10}(x)$, which finally needs the following operations: 20 multiplications, 34 add/sub, 2 divisions and 5 square roots, for root calculation, as well as 3 comparison/swapping operations for root ordering. The C program for the root calculation and ordering is given in [Gras97b].

In Appendix D.3, it is shown that:

$$D_{10}(x=+1)>0\,,\quad P'_{10}(x=+1)>0\,,\quad Q'_{10}(x=+1)>0 \tag{6.2}$$

Therefore the direction of the sign changes at every zero-crossing is known. This property is used for improving efficiency and reliability of the algorithm. In particular, this property plays an essential role in the algorithm denoted as "quantized-search Kabal" (see § 6.3).

Experimental Evaluation

The Mixed-LSP algorithm was tested using the whole TIMIT database (6300 speech files) [Garo90]. For this experience, as well as for the rest of the experiments reported in this chapter, the speech files were downsampled to 8 kHz and the LPC vectors were calculated as in the CELP FS1016 (see § 5.11), using high-pass filtering of the speech input, 30 ms Hamming windowing, autocorrelation method, and 15 Hz bandwidth expansion ($\gamma = 0.994$). For every speech file, two sets of LSP vectors were calculated, one using the Mixed-LSP algorithm, and the other with a high accuracy method ($\varepsilon < 10^{-16}$).

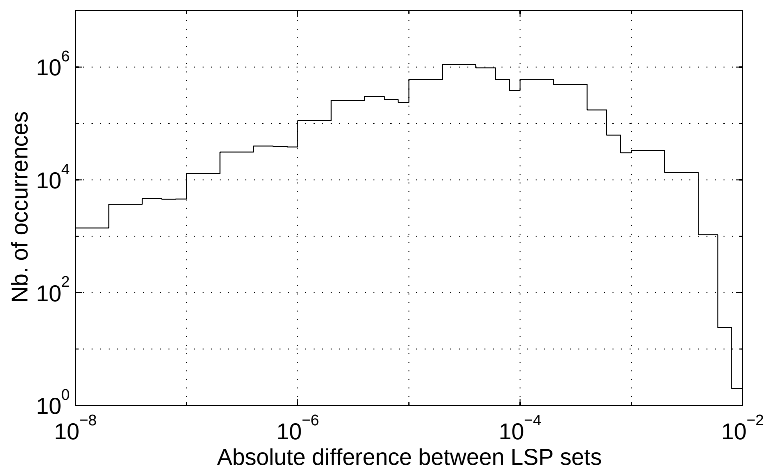


Figure 6.2: Histogram of the absolute difference between LSP sets calculated with Mixed-LSP on one side, and high accuracy on the other side.

The histogram of the absolute differences found on the whole TIMIT database is given in Figure 6.2. The maximum absolute difference found is 0.0092.

6.2. LSP Quantization in the "x-domain" versus LSP Quantization in the "ω-domain"

In the CELP FS1016, the LSP coefficients are quantized using the quantization tables given in Figure 5.5.

As the LSPs obtained with the methods of Kabal, Saoudi and Mixed-LSP are in the "x-domain", it is desirable to perform the quantization in this domain. This is done by applying the mapping $\xi_i = \cos(2\pi f_i)$ to the values f_i of the quantization tables.

Both quantization in the angular frequency ("ω-domain") and in the "x-domain" were evaluated using spectral distortion measure (see § 5.4) on the whole TIMIT database. The LSPs were first calculated with high accuracy and then quantized. The resulting average spectral distortion and percentage of outliers is given in Table 6.1. It is observed that the performance of both quantization in the "ω-domain" and in the "x-domain" are equivalent. Hereafter, the quantization will be done in the "x-domain" for LSPs calculated with the methods of Kabal, Saoudi and Mixed-LSP and in the "ω-domain" for Chan's Method.

Type of quantization	***Average SD (dB)***	***% 2-4 dB outliers***	***% >4 dB outliers***
Quantization in "ω-domain"	1.5326	12.3024	0.1881
Quantization in "x-domain"	1.5329	12.3450	0.1888

Table 6.1: Comparison between quantization performed in the "x-domain" and quantization performed in the "ω-domain".

6.3. Second Proposed Method: Quantized-search Kabal

As mentioned in the previous section, the LSP parameters can be first calculated, using a method such as Kabal's method, and then quantized with the 34-bit non-uniform scalar quantizer of Figure 5.5. To speed up the calculation and quantization processes, a quantized-search technique is used, obtaining the algorithm referred to as "quantized-search Kabal".

The quantized LSPs in the "x-domain" are denoted as $\{qx_i\}$, and the ordering property, which is a necessary condition for stability of the quantized LPC synthesis filter (see § 5.6) is given by:

$$+1 > qx_1 > qx_2 > \ldots > qx_{10} > -1 \tag{6.3}$$

To locate the quantized value, qx_i, of the *i*-th LSP, x_i, the search for the corresponding zero-crossing is done on either $P'_{10}(x)$ (odd-suffixed LSPs) or $Q'_{10}(x)$ (even-suffixed LSPs). This search uses the values of the *i*-th quantization table, converted to the "x-domain" and Kabal's recursive polynomial evaluation given in Equation (5.32). Once the interval containing the zero-crossing, (ξ_{k-1},ξ_k), is found, first the quantized LSP is selected as ξ_k, and then its position is corrected using either the "single-correction" or the "coupled-correction" criterion, which are explained in the next subsections.

Once a quantized LSP, qx_i, is determined, the search for the next quantized LSP, qx_{i+1}, is done using the values of the *i+1*-th quantization table, starting from the first "allowed" value which

would ensure the ordering property of Equation (6.3). As the direction of the sign change at every zero-crossing is known (see § D.3), it is possible to detect if the zero-crossing has already occurred at the first "allowed" value, improving efficiency and reliability of the algorithm.

If the zero-crossing has already occurred at the first "allowed" value, the "coupled-correction" criterion is used to correct the position of the quantized LSP. Otherwise, the "single-correction" criterion is used.

«Single-correction»

The criterion of "single-correction" is explained with the help of Figure 6.3. If an interval (ξ_{k-1},ξ_k) contains the *i*-th zero-crossing, then $qx_i = \xi_k$ is selected. If ξ_k is not the first "allowed" value of the quantization table, qx_i can be "single-corrected", choosing ξ_{k-1} if it is closer to x_i.

The situation is illustrated in Figure 6.3 for the particular case of a zero-crossing of the polynomial $P'_{10}(x)$ from positive to negative (as it is the case for the LSPs x_1, x_5 and x_9), but the discussion that follows is general to all possible cases.

When the LSPs are first calculated and then quantized, x_i is known, and the "horizontal single-correction" (H-SC) criterion is used [CELP3.2a]:

$$\begin{array}{lll} \text{if } H_k \geq H_{k-1} & \Rightarrow qx_i = \xi_{k-1} \\ \text{else} & \Rightarrow qx_i = \xi_k \end{array} \tag{6.4}$$

where H_{k-1} and H_k are the horizontal distances from ξ_{k-1} and ξ_k to the actual LSP value x_i, as shown in Figure 6.3.

In the case of a quantized domain search, only the values of $P'_{10}(\xi_{k-1})$ and $P'_{10}(\xi_k)$ are known, but not x_i. In the direct conversion from predictor coefficients to quantized LSPs proposed by Wolovitz [Camp89], [CELP3.2a], qx_i is selected using a "vertical single-correction" (V-SC) criterion:

$$\begin{array}{lll} \text{if } V_k \geq V_{k-1} & \Rightarrow qx_i = \xi_{k-1} \\ \text{else} & \Rightarrow qx_i = \xi_k \end{array} \tag{6.5}$$

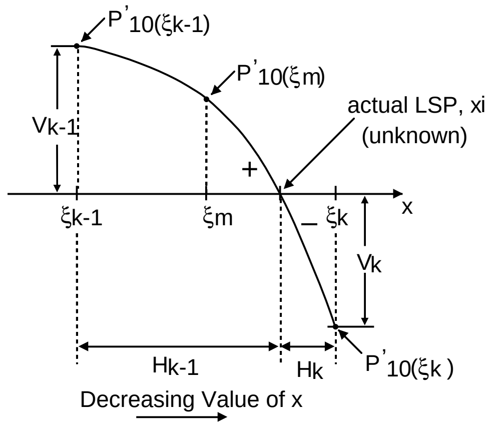


Figure 6.3: Illustration of a zero-crossing of the polynomial $P'_{10}(x)$ from positive to negative, used to explain "single-correction" criterion.

where V_{k-1} and V_k are the vertical distances from $P'_{10}(\xi_{k-1})$ and $P'_{10}(\xi_k)$ to the x-axis, as shown in Figure 6.3.

This "vertical single-correction" criterion does not necessary choose the closest value to the actual LSP, depending on the concavity of the polynomial $P'_{10}(x)$ or $Q'_{10}(x)$ at the zero-crossing. We propose the following criterion that can be used in a quantized domain search, at the cost of 10 extra polynomial evaluations, and is equivalent to the "horizontal single-correction" criterion. The polynomial $P'_{10}(x)$ is evaluated at the center of the interval containing the zero-crossing:

$$\xi_m = \frac{\xi_{k-1} + \xi_k}{2} \tag{6.6}$$

If the zero-crossing is from positive to negative, qx_i is:

$$\begin{array}{ll} \text{if } P'_{10}(\xi_m) \le 0 & \Rightarrow qx_i = \xi_{k-1} \\ \text{else} & \Rightarrow qx_i = \xi_k \end{array} \tag{6.7}$$

Else, if the zero-crossing is from negative to positive qx_i is:

$$\begin{array}{ll} \text{if } P'_{10}(\xi_m) \ge 0 & \Rightarrow qx_i = \xi_{k-1} \\ \text{else} & \Rightarrow qx_i = \xi_k \end{array} \tag{6.8}$$

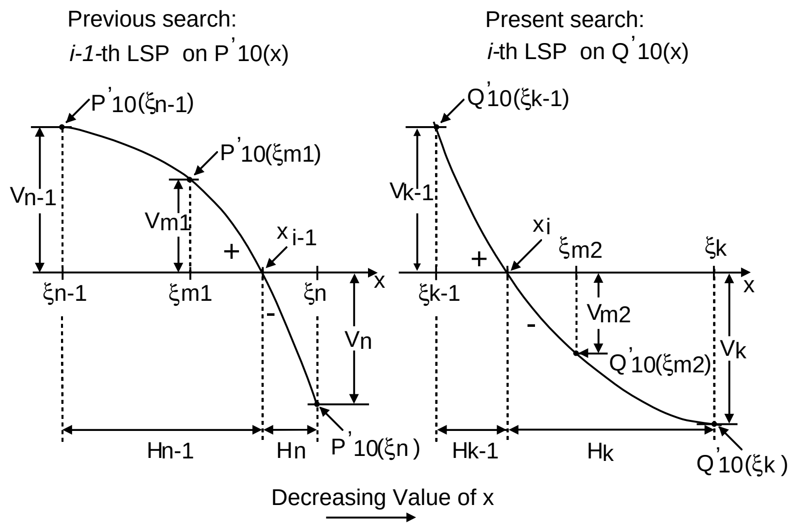


Figure 6.4: Illustration of two successive zero-crossings, from positive to negative, of the polynomials $P'_{10}(x)$ and $Q'_{10}(x)$, used to explain "coupled-correction" criterion.

«Coupled-correction»

The criterion of "coupled-correction" considers the interaction between two consecutive LSPs, and is better explained with the help of Figure 6.4. Here, the interval (ξ_{n-1},ξ_n) contains the *i-1*-th LSP, x_{i-1}, and the interval (ξ_{k-1},ξ_k) contains the *i*-th LSP, x_i. In the previous search, as ξ_n is closer to x_{i-1} than ξ_{n-1}, $qx_{i-1} = \xi_n$ was selected (i.e. qx_{i-1} was not "single-corrected").

If the zero-crossing corresponding to x_i has already occurred at the first "allowed" value of the *i*-th quantization table, the intervals (ξ_{n-1},ξ_n) and (ξ_{k-1},ξ_k) overlap, with $\xi_{k-1} > \xi_n$, and the choice of $qx_{i-1} = \xi_n$ would force the choice $qx_i = \xi_k$, to preserve the ordering property. In this case, the "coupled-correction" criterion is used to decide which choice, $(qx_{i-1},qx_i)=(\xi_n,\xi_k)$ or $(qx_{i-1},qx_i)=(\xi_{n-1},\xi_{k-1})$, is better.

When the LSPs are first calculated and then quantized, x_i and x_{i-1} are known, and the "horizontal coupled-correction" criterion is used [CELP3.2a]:

$$\begin{array}{lll} \text{if} \quad H_n + H_k \geq H_{n-1} + H_{k-1} & \Rightarrow (qx_{i-1}, qx_i) = (\xi_{n-1}, \xi_{k-1}) & \\ \text{else} & \Rightarrow (qx_{i-1}, qx_i) = (\xi_n, \xi_k) & (6.9) \end{array}$$

where H_{n-1} and H_n are the horizontal distances from ξ_{n-1} and ξ_n to x_{i-1}, and H_{k-1} and H_k are the horizontal distances from ξ_{k-1} and ξ_k to x_i, as shown in Figure 6.4.

In the case of a quantized domain search, the values of x_i and x_{i-1} are not known, thus the criterion of Equation (6.9) cannot be used. The direct conversion from predictor coefficients to quantized LSPs proposed by Wolovitz [Camp89], [CELP3.2a], does not use "coupled-correction". A "vertical coupled-correction" criterion analogous to the "vertical single-correction" criterion, could be used:

$$\begin{array}{lll} \text{if} \quad V_n + V_k \geq V_{n-1} + V_{k-1} & \Rightarrow (qx_{i-1}, qx_i) = (\xi_{n-1}, \xi_{k-1}) & \\ \text{else} & \Rightarrow (qx_{i-1}, qx_i) = (\xi_n, \xi_k) & (6.10) \end{array}$$

where V_{n-1} and V_n are the vertical distances from $P'_{10}(\xi_{n-1})$ and $P'_{10}(\xi_n)$ to the x-axis, and V_{k-1} and V_k are the vertical distances from $Q'_{10}(\xi_{k-1})$ and $Q'_{10}(\xi_k)$ to the x-axis, as shown in Figure 6.4.

By simulation, it was found that this criterion differs significantly from the "horizontal coupled-correction" criterion. Thus, the following "enhanced vertical coupled-correction" (EV-CC) criterion, whose performance is very similar to the "horizontal coupled-correction" criterion, is proposed:

If the zero-crossing is from positive to negative:

$$\begin{array}{ll} \text{if} \quad Q'_{10}(\xi_{m2}) \leq 0 \text{ and } V_{m2} > V_{m1} & \Rightarrow (qx_{i-1}, qx_i) = (\xi_{n-1}, \xi_{k-1}) \\ \text{else} & \Rightarrow (qx_{i-1}, qx_i) = (\xi_n, \xi_k) \end{array} \quad (6.11)$$

Else, if the zero-crossing is from negative to positive:

$$\begin{array}{ll} \text{if} \quad Q'_{10}(\xi_{m2}) \geq 0 \text{ and } V_{m2} > V_{m1} & \Rightarrow (qx_{i-1}, qx_i) = (\xi_{n-1}, \xi_{k-1}) \\ \text{else} & \Rightarrow (qx_{i-1}, qx_i) = (\xi_n, \xi_k) \end{array} \quad (6.12)$$

where ξ_{m1} and ξ_{m2} are the center of the intervals (ξ_{n-1},ξ_n) and (ξ_{k-1},ξ_k), respectively, and V_{m1} and V_{m2} are the vertical distances from $P'_{10}(\xi_{m1})$ and $Q'_{10}(\xi_{m2})$ to the x-axis, as shown in Figure 6.4.

In the search for the 6-th quantized LSP, qx_6, if the previous quantized LSP, qx_5, takes one of the three values marked in boldface in the 5-th quantization table of Figure 5.5, a "coupled-correction" would not preserve the ordering property. Thus in these three particular cases, which correspond to $qx_5 = \xi_n = 0.2563$, $qx_5 = \xi_n = 0.0392$, and $qx_5 = \xi_n = -0.1175$, the "coupled-correction" is skipped.

In summary, "coupled-correction" for the *i*-th LSP, using either of the proposed criteria, is considered only if the following conditions are met:

- The zero-crossing corresponding to x_i has already occurred at the first "allowed" value of the *i*-th quantization table.
- The position of the previous quantized LSP, qx_{i-1}, was not corrected (either with single- or coupled-correction), and qx_{i-1} was not the first "allowed" value of its quantization table (or the first value of its quantization table).
- If, in the search for the *6*-th quantized LSP, the previous quantized LSP did not take one of these three particular values: $qx_5 = 0.2563$, $qx_5 = 0.0392$, or $qx_5 = -0.1175$.

Experimental Evaluation

Several versions of the "quantized-search Kabal" algorithm, with different correction criteria, were evaluated using spectral distortion on the whole TIMIT database. Kabal's method followed by quantization was also evaluated for comparison.

The resulting average spectral distortion and percentage of outliers is given in Table 6.2. It is observed that among the "quantized-search Kabal" algorithms, the algorithm using both "horizontal single-correction" (H-SC) and "enhanced vertical coupled-correction" (EV-CC) criteria has the best performance. Furthermore, the performance of this algorithm is very close to the performance of Kabal's algorithm followed by quantization in the "x-domain".

The different versions of the "quantized-search Kabal" algorithm were also compared with the high accuracy method followed by quantization (reference algorithm). The differences between the LSP indices calculated with the algorithm under evaluation and the reference algorithm were counted, and the results are given in Table 6.3. The number of frames containing one, two, three, four and more than four differences of one on the LSP indices are denoted as n1, n2, n3, n4 and n5 respectively. The number of frames containing at least a difference bigger than one on the LSP indices is denoted as nn.

"Quantized-search Kabal" correction criteria	***Ave. SD (dB)***	***% 2-4 dB outliers***	***% >4 dB outliers***
V-SC	1.55552	13.88856	0.22244
V-SC + V-CC	1.55218	13.78761	0.19226
H-SC	1.53495	12.43227	0.19335
H-SC + V-CC	1.53368	12.36321	0.19008
H-SC + EV-CC	1.53295	12.35014	0.18946
Kabal + quant. in the "x-domain"	1.53288	12.34532	0.18884

Table 6.2: Comparison among "quantized-search Kabal" algorithms with different correction criteria, and Kabal's algorithm + quantization, in terms of spectral distortion (V-SC = "vertical single-correction", H-SC = "horizontal single-correction", V-CC = "vertical coupled-correction", EV-CC = "enhanced vertical coupled-correction").

"Quantized-search Kabal" correction criteria	***n_1***	***n_2***	***n_3***	***n_4***	***n_5***	***nn***
V-SC	158620	27920	3376	287	26	10
V-SC + V-CC	159262	26059	2688	185	11	2
H-SC	0	3301	15	7	0	4
H-SC + V-CC	0	2336	10	2	0	1
H-SC + EV-CC	0	706	3	0	0	2

Table 6.3: Comparison among "quantized-search Kabal" algorithm with different correction criteria, and high accuracy method + quantization in the "x-domain", in terms of differences in the obtained indices (Differences of one: n1 = frames with one difference, n2 = frames with two differences, n3 = frames with three differences, n4 = frames with four differences, n5 = frames with more than four differences. Differences bigger than one: nn = frames with at least one difference).

Hereafter, the name "quantized-search Kabal" refers to the version which uses both "horizontal single-correction" and "enhanced vertical coupled-correction" criteria. The differences between LSP indices calculated with this algorithm and the reference algorithm are analyzed in Appendix E.2.

Quantized-search Chan

The LSPs in the "ω-domain" can be first calculated from the reflection coefficients using Chan's method (see § 5.9) and then quantized using the 34-bit quantization tables of Figure 5.5. Similarly to "quantized-search Kabal" algorithm, the LSP calculation and quantization processes are embedded, obtaining the algorithm referred to as "quantized-search Chan".

The quantized LSPs in the "ω-domain" are denoted as $\{q\omega_i\}$, and the ordering property is given by:

$$0 < q\omega_1 < q\omega_2 < \ldots < q\omega_{10} < \pi \qquad (6.13)$$

To locate the *i*-th quantized LSP, $q\omega_i$, the search for the corresponding zero-crossing of $\mathrm{Re}[\psi_{10}(e^{j\omega})]$ or $\mathrm{Im}[\psi_{10}(e^{j\omega})]$ is done using the recursive evaluation for $\psi_{10}(e^{j\omega})$ given in Equation (5.45) and the values of the *i*-th quantization table.

Once the interval containing the zero-crossing is found, the quantized LSP is selected using either the "single-correction" or the "coupled-correction" criterion, explained previously. The best performance among different versions of the "quantized-search Chan" algorithm, is obtained using both "horizontal single-correction" (H-SC) and "enhanced vertical coupled-correction" (EV-CC) criteria. Hereafter, this version will be denoted as "quantized-search Chan".

Computational Complexity

In "quantized-search Kabal" algorithm, polynomial evaluation is done using the efficient Kabal's recursion of Equation (5.32), at the cost of 4 multiplications and 9 additions per evaluation, whereas in "quantized-search Chan" algorithm the evaluation of $\psi_{10}(e^{j\omega})$ is done with the recursion of Equation (5.45), at the cost of 30 multiplications and 20 additions per evaluation. In appendix E.1 it is shown that the maximum possible number of polynomial evaluations in both “quantized-search Kabal” and “quantized-search Chan” is 71. In practice, the maximum number of evaluations found by simulation on the whole TIMIT database was 68.

The flow control of these algorithms is greatly simplified by using flags to store the conditions tested in the correction criteria. To avoid expensive comparisons, the quantization tables of Figure 5.5 are modified to include, with each quantization level, an index (or offset) to the first “allowed” value of the next quantization table. Also, some flags indicating conditions such as “first element of the table”, “last element of the table” and “particular case of qx_5” are stored in the quantization table, to simplify the control flow of the algorithm. More details are given in Chapter 7.

6.4. Program for LSP Quantization

In the CELP FS1016, the LSPs can be first obtained with the methods of Kabal, Saoudi, Chan or Mixed-LSP and then quantized (see § 6.2). In the quantization program distributed with [CELP3.2a] the LSPs are first quantized independently using the equivalent to the “horizontal single-correction” of Equation (6.4), and then the ordering property is checked by expensive comparisons, using the equivalent of the “horizontal coupled-correction” of Equation (6.9) to correct the position if the ordering property was not respected.

We have elaborated a quantization program more suitable for efficient real-time implementation. This program produces the same results as the program distributed with [CELP3.2a] and is very similar to the “quantized-search Kabal” algorithm, except that, as the LSPs are known, the single- and coupled-correction criteria of Equations (6.4), and (6.9) are used.

Efficient real-time implementation is obtained with the use of flags and offsets, similarly to the implementation of “quantized-search Kabal” explained in the previous subsection.

6.5. Bandwidth Expansion and Spectral Smoothing

A drawback in the utilization of the algorithms of Saoudi and Chan in the CELP FS1016 (see § 5.11) is that the 15 Hz bandwidth expansion (see § 5.3) cannot be easily applied, as the LPC coefficients are not calculated in the LeRoux-Gueguen and in the antisymmetric split-Levinson recursions.

An effect similar to bandwidth expansion can be obtained with the spectral smoothing technique described in [Tohk78], in which the autocorrelation coefficients of Equation (5.10) are multiplied by a Gaussian window. This is equivalent to convoluting the LPC power spectrum with the Fourier transform of a Gaussian window, which is itself a Gaussian window. After such a convolution, sharp spectral peaks are smoothed out, and the LPC poles are moved inward the unit circle. Nevertheless, some effort would be necessary to adapt this technique to the CELP FS1016 speech coder.

On the other hand the spectral smoothing technique would not give numerically the same results than the bandwidth expansion. Thus, in order to make meaningful comparisons among the different algorithms (see § 6.6 and § 6.7) the reflection coefficients needed in Chan’s method and the autocorrelation coefficients needed in Saoudi’s method are obtained by transformation from the bandwidth expanded LPC coefficients.

6.6. Accuracy of the Different Algorithms

The accuracy of Kabal’s, Saoudi’s, Chan’s and the proposed Mixed-LSP algorithms was evaluated, by comparison with the high accuracy method, using the whole TIMIT database. For every speech file, two sets of LSP vectors were compared, one set calculated with the algorithm under evaluation, and the other set with high accuracy. The histograms of the absolute differences found for every algorithm under test are given in Figure 6.5 and Figure 6.6. The maximum absolute difference found for every algorithm is given in Table 6.4.

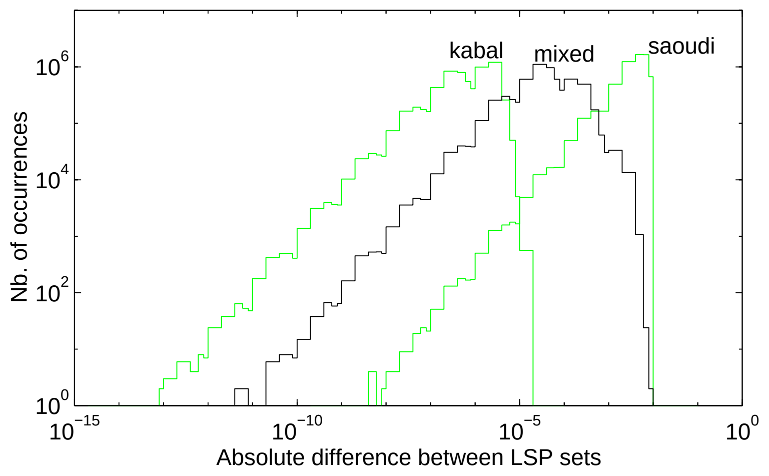


Figure 6.5: Histogram of the absolute difference between LSP sets calculated with high accuracy on one side, and Kabal's, Saoudi's and Mixed-LSP on the other side.

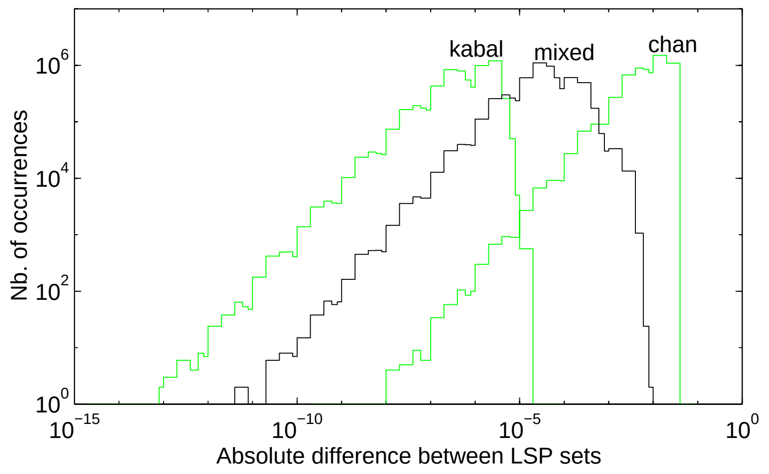


Figure 6.6: Histogram of the absolute difference between LSP sets calculated with high accuracy on one side, and Kabal's, Chan's and Mixed-LSP on the other side.

Note that the LSPs calculated with Chan's algorithm were converted from the "ω-domain" to the "x-domain" in order to make a meaningful comparison.

It is observed that Kabal's is the most accurate among the algorithms under evaluation, followed by Mixed-LSP and then Saoudi's and Chan's. Saoudi's algorithm is slightly more accurate than Chan's algorithm.

LSP calculation method	***Maximum absolute difference***
Kabal	0.0000137
Mixed-LSP	0.0091698
Saoudi	0.0078124
Chan	0.0294831

Table 6.4: Maximum absolute difference between LSP sets calculated with high accuracy on one side, and Kabal's, Saoudi's and Mixed-LSP methods on the other side.

Spectral Distortion

Kabal's, Saoudi's, Chan's and Mixed-LSP algorithms, as well as the high accuracy method were used to calculate the LSPs, which were then quantized with the 34-bit scalar quantizer. Spectral distortion was measured in all cases, and the results are given in Table 6.5, together with the spectral distortion measured for the "quantized-search Kabal" algorithm.

Algorithms to obtain the quantized LSPs	***Ave. SD (dB)***	***% 2-4 dB outliers***	***% >4 dB outliers***
High accuracy + quant. in x	1.53287	12.34501	0.18884
Kabal + quant. in x	1.53288	12.34532	0.18884
Mixed-LSP + quant. in x	1.53308	12.36305	0.18853
Q.-search Kabal (H-SC + EV-CC)	1.53295	12.35014	0.18946
Saoudi + quant. in x	1.65362	19.11655	0.20253
Chan + quant. in ω	1.72734	24.46555	0.22648

Table 6.5: Comparison among different methods to calculate quantized LSPs, in terms of spectral distortion.

The results obtained using Kabal's, Mixed-LSP and "quantized-search Kabal" algorithms are very close to those obtained with the high accuracy method. Furthermore, the histograms of spectral distortion of these four algorithms superpose. Thus, although the Mixed-LSP method is less accurate than Kabal's method, it is sufficient for speech coding applications using the 34-bit scalar quantizer of the CELP FS1016. It is also observed that the "quantized-search Kabal" algorithm can be used to speed up the calculation and quantization processes, without degradation of the quantization performance.

On the other hand, the quantization performance is degraded when the algorithms of Saoudi and Chan are used for LSP calculation. This is due to the inaccuracy of these algorithms, observed in Figure 6.5 and Figure 6.6.

The accuracy of Chan's and Saoudi's algorithms could be increased by adding bisections and/or linear interpolation, at the cost of increased computational complexity.

The most cost effective way of improving the accuracy of Chan's algorithm is through the use of linear interpolation, at the added cost of 20 multiplications, 20 additions, and 10 divisions. The resulting algorithm was used to calculate the LSPs, which were then quantized. Spectral distortion was measured using the TIMIT database, and the results are reported in Table 6.6. It is observed that the performance using Chan's method with linear interpolation is very close to the performance of the high accuracy method.

Algorithms to obtain the quantized LSPs	***Ave. SD (dB)***	***% 2-4 dB outliers***	***% >4 dB outliers***
Chan with linear interp. + quant. in ω	1.53341	12.37892	0.18853
High accuracy + quant. in x	1.53287	12.34501	0.18884

Table 6.6: Evaluation of Chan's method with linear interpolation used to calculate quantized LSPs, in terms of spectral distortion.

In Saoudi's method, it is more cost effective to improve the accuracy by adding extra bisections, at the cost of 10 additions and 8 multiplications per bisection. Four versions of Saoudi's algorithm, differing in the number of bisections, were used to calculate the LSPs before quantization. Spectral distortion was

measured in all cases, and is given in Table 6.7. It is observed that a performance close to the performance using the high accuracy method is obtained with Saoudi's method using 11 bisections. A value of 10 bisections could also be acceptable.

Algorithms to obtain the quantized LSPs	***Ave. SD (dB)***	***% 2-4 dB outliers***	***% >4 dB outliers***
Saoudi, nbis. = 8, + quant. in x	1.65362	19.11655	0.20253
Saoudi, nbis. = 9, + quant. in x	1.55839	13.62490	0.19133
Saoudi, nbis. = 10, + quant. in x	1.54039	12.76484	0.18961
Saoudi, nbis. = 11, + quant. in x	1.53535	12.51223	0.18884
High accuracy + quant. in x	1.53287	12.34501	0.18884

Table 6.7: Evaluation, in terms of spectral distortion, of Saoudi's method with different number of bisections, used to calculate quantized LSPs.

6.7. Reliability of the Different Algorithms

An important aspect of an LSP calculation algorithm is its reliability, which is the ability to find the true LSP parameters, without missing any zero-crossing. An additional requirement is that the obtained LSP comply with the ordering property of Equations (5.25) and (5.31), for stability of the corresponding LPC synthesis filter.

The minimum difference between adjacent LSPs, as well as the minimum difference between LSPs with the same suffix type (either odd- or even-suffixed) plays an important role in the reliability of LSP calculation algorithms. These differences were measured on the TIMIT database. The LSPs were calculated in both the "x-domain" and the "ω-domain" with high accuracy, from LPC vectors obtained as in the CELP FS1016, with and without the 15 Hz bandwidth expansion. The minimum differences found are given in Table 6.8 for LSPs in the "x-domain" and in Table 6.9 for LSPs in the "ω-domain".

The minimum LSP differences reported in [Kaba86] are also given in Table 6.8. These LSP differences were measured on 10 s of speech sampled at 8 kHz, using a 20 ms Hamming window and 10-th order LPC autocorrelation method [Kaba86].

	Minimum differences		
Type of LSP (in the "x-domain")	***on TIMIT with BW expansion***	***on TIMIT without BW expansion***	***as reported in [Kaba86]***
Odd-suffixed LSPs	0.0265	0.0167	0.0232
Even-suffixed LSPs	0.0389	0.0319	0.0564
Adjacent LSPs	0.0026	0.0006	0.0015

Table 6.8: Minimum differences between LSPs in the "x-domain", as reported in [Kaba86] and as measured on the TIMIT database, with and without bandwidth expansion.

A search grid of $\Delta = 0.02$ was chosen in [Kaba86]. This grid value was chosen smaller than the minimum distance between LSPs of the same type (0.0232) found in [Kaba86], to avoid missing zero-crossings. In Table 6.8 it is observed that this choice of Δ is also valid for the TIMIT database, but only when the 15 Hz bandwidth expansion is used. In case that the separation between LSPs of the same type is smaller than the grid size, Kabal's algorithm will fail to locate the right LSPs. Nevertheless, we decided not to add any additional (computational expensive) checking to avoid this unlikely condition.

As bandwidth expansion smoothes out spectral peaks (see § 5.3 and § 6.5), it increases the distance between LSPs (see § 5.7), improving the reliability of LSP calculation algorithms based on zero-crossing search, such as Kabal's and Chan's algorithm. Note that bandwidth expansion is commonly used in speech coders.

In Kabal's algorithm, the number of bisections is specified by the acceptable uncertainty in an root position,$\in$. This value must be small enough so that, in switching the search from the polynomial $P'_{10}(x)$ to $Q'_{10}(x)$ and vice versa, a root is not missed or roots are not interchanged in order. Thus $\in$ must be smaller than the minimum difference between adjacent LSPs:

$$\in = \frac{\Delta}{2^{nbis}} < \min_i (x_i - x_{i+1}) \tag{6.14}$$

A number of 4 bisections is selected in [Kaba86], giving an uncertainty of $\in = 0.00125$ in the root position. This uncertainty

is smaller than the minimum difference between adjacent roots found in [Kaba86] (0.0015). This choice of number of bisections is also valid for the TIMIT database, but only if the 15 Hz bandwidth expansion is used. Note also that the uncertainty in the root position is significantly decreased by the linear interpolation, which have thus a beneficial effect on reliability. Furthermore, knowledge of the direction of sign changes (see § D.3) was included in Kabal's algorithm as a cost-effective way of improving reliability in the case where the value of $\in$ is bigger than the difference between two adjacent LSPs.

It was found by simulation on the whole TIMIT database, using bandwidth expanded LPC, that the LSPs calculated with Kabal's algorithm always produce an ordered set.

In Chan's algorithm [Chan91], the search is done in the "ω-domain" with a grid of $\Delta = 0.0078\pi$. In Table 6.9 it is seen that this grid is largely smaller than the minimum separation between LSPs of the same type, whether bandwidth expansion is used or not.

	Minimum differences	
Type of LSP (in the "ω-domain")	***on TIMIT with BW exp.***	***on TIMIT without BW exp.***
Odd-suffixed LSPs	0.02071π	0.01338π
Even-suffixed LSPs	0.02708π	0.01911π
Adjacent LSPs	0.00430π	0.00069π

Table 6.9: Minimum differences between LSPs in the "ω-domain", measured on the TIMIT database, with and without bandwidth expansion.

As Chan's algorithm does not use bisections or interpolation, the uncertainty in the LSP determination has the same value as the grid size, $\in = 0.0078\pi$. In Table 6.9, it is observed that this value could become bigger than the minimum separation of adjacent LSPs. Knowledge of the direction of sign changes (see § D.3) was included in Chan's algorithm to improve reliability under this condition. The LSPs calculated with Chan's algorithm, on the whole TIMIT database and using bandwidth expanded LPC, were always ordered.

Saoudi's algorithm is intrinsically reliable [Saou95]. Each LSP is calculated independently starting from the interval

$(-2,+2)$ and using 8 successive bisections: the value of the bisection point is used to evaluate the recursion of Equation (5.37) or (5.38), and the number of sign changes incurred in this evaluation is used to know with exactitude in which of the two bisected interval the LSP is located. Thus zero-crossings cannot be missed, and this independently of the speech database [Saou95].

On the other hand, as each LSP is searched independently, and with a coarse precision of $2^{-7} = 0.0078$, it is possible that two contiguous LSP take the same value (i.e. the ordering property is not respected) if their separation is smaller than the precision. In Table 6.8 it is observed that this condition can occur, even using bandwidth expansion, and in fact it was found by simulation (using bandwidth expanded LPC) that this condition occurs in a large amount of speech files of the TIMIT database. Additionally, it was found that at least 10 bisections would be required to avoid this condition on the TIMIT database. As this problem is corrected by the quantization program (see § 6.4), it is not important in the case of application in the CELP FS1016. Nevertheless, care must be taken when using Saoudi's algorithm in other applications.

In the proposed Mixed-LSP algorithm (see § 6.1) five intervals, containing each only one zero-crossing of $P'_{10}(x)$ and one zero-crossing of $Q'_{10}(x)$, are calculated. Thus, zero-crossings cannot be missed.

Inside each interval, the positions of the root of $P'_{10}(x)$ and $Q'_{10}(x)$ are refined, independently of each other, using five bisections. Thus there is the possibility that two roots take the same value, or are interchanged in order. In practice, it was found by simulation (using bandwidth expanded LPC) that this condition never occurs on the TIMIT database. Furthermore, a slight interchange in order would be corrected by the quantization program. This condition could also be avoided with certitude by calculating first the root of $P'_{10}(x)$, and then using this value to limit the calculation interval of the root of $Q'_{10}(x)$.

The “quantized-search Kabal” algorithm always produces an ordered set of LSPs. Simulations on the TIMIT database showed that in four cases the zero-crossing were missed, due to the coarse search grid. In one case a zero-crossing was detected twice. More information on these conditions is given in

Appendix E.2. In listening tests using the CELP FS1016 speech coder, these exceptions did not introduce additional audible distortion (see § E.2). Thus, to keep the low complexity of “quantized-search Kabal” we decided not to add any checking, or to interpolate the grid values as proposed in [Chan95], to avoid these unlikely conditions.

6.8. LSP Interpolation in the “x-domain” versus LSP Interpolation in the “ω-domain”

In both the receiver and the transmitter of the CELP FS1016 (see § 5.11), two adjacent sets of quantized LSP parameters are interpolated obtaining four sets of LSP parameters, which are then converted to LPC coefficients and used in the synthesis filter. The interpolation is usually done using LSPs expressed in the “ω-domain” [Fede91], [CELP3.2a].

As the LSPs obtained with the methods of Kabal, Saoudi, Mixed-LSP and “quantized-search Kabal” are in the “x-domain”, it is desirable to perform the interpolation in this domain. This avoids the computationally expensive conversion from the “x-domain” to the “ω-domain” for interpolation, and then from the “ω-domain” to “x-domain” for LSP to LPC transformation (see § 5.10).

Both interpolation in the “ω-domain” and in the “x-domain” were evaluated as it is proposed in [Pali95b] using spectral distortion on the TIMIT database. This is explained next.

The LPC used for interpolation were calculated, for each frame of 30 ms, as in the CELP FS1016, using high-pass filtering of the speech input, non-overlapping 30 ms Hamming windowing, autocorrelation method, and 15 Hz bandwidth expansion. The Hamming window is centered at the end of every frame, as indicated in Figure 5.9. For every frame, the LSPs were calculated with high accuracy, in both the “x-domain” and the “ω-domain”.

The interpolation process is explained as follows. For each frame, two sets of LSP parameters, corresponding to the window positions A and B in Figure 5.9, are used for interpolation with the weights given in Table 5.1, obtaining four sets of LSP parameters. Each of these LSP sets is converted to LPC coefficients, obtaining four sets of interpolated LPC coefficients, one set per subframe.

The interpolation process was repeated twice, in one case the LSPs were interpolated in the “x-domain” and in the other case they were interpolated in the “ω-domain”. In both cases, the obtained interpolated LPC synthesis filters were compared with respect to the “true” LPC synthesis filters, using spectral distortion measure (see § 5.4).

The “true” LPC coefficients were calculated, for each subframe of 7.5 ms, using high-pass filtering of the speech input, an overlapping 30 ms Hamming window centered at the subframe, autocorrelation method and 15 Hz bandwidth expansion.

The resulting average spectral distortion and percentage of outliers is given in Table 6.10. It is observed that the measures for both interpolation in the “ω-domain” and in the “x-domain” are very similar. Hereafter, the interpolation will be done in the “x-domain” for LSPs calculated with the methods of Kabal, Saoudi, Mixed-LSP and “quantized-search Kabal”.

The reader is reminded that the measures reported in Table 6.10 do not represent audible distortion introduced by the interpolation processes, but a “distance measure” to a (somehow arbitrarily chosen) reference LPC, used to compare two different types of interpolation.

Type of interpolation	***Average SD (dB)***	***% 2-4 dB outliers***	***% >4 dB outliers***
Interpolation in “ω-domain”	1.5568	21.8325	4.2183
Interpolation in “x-domain”	1.5656	21.9702	4.3733

Table 6.10: Comparison between interpolation performed in the “x-domain” and interpolation performed in the “ω-domain”.

6.9. Computational Complexity

The proposed Mixed-LSP, "quantized-search Kabal" and "quantized-search Chan" algorithms are compared in complexity with the algorithms of Kabal, Saoudi, and Chan (see § 5.9). The number of operations required by each algorithm is shown in Figure 6.7.

The total number of operations per frame needed to obtain the LSPs with the different algorithms is given in Table 6.11. Here the algorithms of Chan and Saoudi are considered as they are proposed originally in [Chan91] and [Saou92]. A similar information is given in Table 6.12, but with the algorithms of Chan and Saoudi modified to obtain a quantization performance equivalent to the performance obtained with the algorithms of Kabal’s, Mixed-LSP and “quantized-search Kabal” (see § 6.6).

Methods to obtain the LSPs	***Mult***	***Add***	***Div***	***Sqrt***
Chan	3930	2660	20	
“quantized-search Chan”	2220	1520	20	
Kabal	730	1530	20	
Saoudi, nbis. = 8	706	941	20	
Mixed-LSP	390	764	22	5
“quantized-search Kabal”	394	769	10	

Table 6.11: Total number of operations per frame needed to obtain the LSPs, using different algorithms. (The algorithms of Chan’s and Saoudi’s are as proposed in [Chan91] and [Saou92]).

Methods to obtain the LSPs	***Mult***	***Add***	***Div***	***Sqrt***
Chan with linear interpolation	3950	2680	30	
Saoudi, nbis. = 11	946	1241	20	
Saoudi, nbis. = 10	866	1141	20	

Table 6.12: Total number of operations per frame needed to obtain the LSPs, using the algorithms of Chan and Saoudi, modified to have a quantization performance equivalent to the other algorithms, for application in the CELP FS1016.

Based on Table 6.11 we would conclude that Saoudi’s algorithm is more efficient than Kabal’s. But, as Saoudi’s algorithm does not have the accuracy required by the application (see § 6.6), the comparison is not valid. Saoudi’s algorithm with 11 bisections and Kabal’s algorithm have a similar quantization performance. In this case, it is observed in Table 6.11 and Table 6.12 that Kabal’s algorithm is less computationally expensive.

Saoudi’s algorithm could be used with 10 bisections, with a small degradation in the quantization performance, and a computational complexity no much higher than in Kabal's algorithm. Nevertheless, the use of Kabal's algorithm in the CELP FS1016 was preferred, due to the possibility to apply directly the bandwidth expansion (see § 6.5).

Both versions of Chan's algorithm, with and without linear interpolation, are computationally too expensive. Originally this algorithm was retained for its possible benefits in a fixed-point implementation and because the LSPs are obtained in the "ω-domain", which is the domain in which the quantization and interpolation is done in the original CELP FS1016 implementation [CELP3.2a]. As it is shown in sections 6.2 and 6.8, the quantization can be done in the "x-domain" with basically no degradation in the performance. On the other hand, given the enormous amount of computation required by Chan's algorithm, it was preferred to chose Kabal's algorithm and do the necessary effort to adapt the algorithm to a fixed-point implementation (see Chapter 7). Furthermore, direct application of bandwidth expansion is not possible in Chan's algorithm.

The proposed "Mixed-LSP" and "quantized-search Kabal" are computationally more efficient than Kabal's algorithm, and have a similar quantization performance, but "Mixed-LSP" algorithm is specific for an LPC order of 10, while "quantized-search Kabal" is tied to the utilization of the 34-bit scalar quantizer of the CELP FS1016. These three algorithms were retained for fixed-point optimization and implementation on a DSP56001 processor, as explained next.

DSP56001 Implementation

The algorithms of Kabal, Mixed-LSP and "quantized-search Kabal", as well as the quantization in the "x-domain" were implemented on a DSP56001 with a clock frequency of 20 MHz.

A simulation of the fixed-point quantization effects was done, following the methodology explained in [Gras94] and

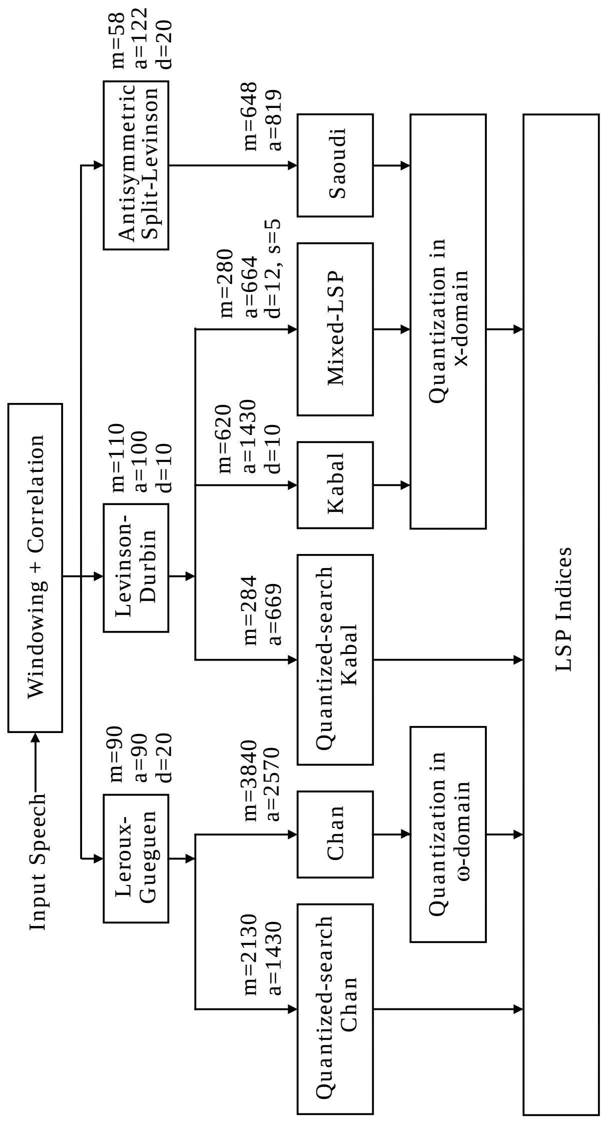


Figure 6.7: Computational complexity of different LSP calculation methods (10-th order LPC) (m=multiplications, a=add/sub, d=divisions, s=square roots).

[Gras95], in order to determine the minimum word-length and the scaling required at every node of the algorithms. The results obtained in the study of the quantization effects were used for efficient implementation on a DSP56001 processor. More details on the implementation and testing are given in Chapter 7.

The computational complexity in MIPS and the maximum number of clock cycles needed for processing one frame of 30 ms are given in Table 6.13 and Table 6.14. The MIPS are calculated assuming an “average instruction” of 2 cycles (thus, the computational power of a DSP56001 at 20 MHz is 10 MIPS).

It is observed that the Mixed-LSP algorithm needs 33 % less cycles than Kabal's algorithm, while "quantized-search Kabal" algorithm needs 66% less cycles than Kabal's algorithm + quantization.

Algorithm	***Number of cycles***	***Execution time (μs)***	***MIPS***
Kabal	10540	527.0	0.1757
Mixed-LSP	6986	349.3	0.1164
“Q.-search Kabal”	4262	213.1	0.0710
Quantization (Q34)	2168	108.4	0.0361

Table 6.13: Computational complexity of the DSP56001 implementation of Kabal’s, Mixed-LSP and “quantized-search Kabal” algorithms, and quantization in the “x-domain” using the 34-bit scalar quantizer of the CELP FS1016.

Algorithm	***Number of cycles***	***Execution time [μs]***	***MIPS***
Kabal + Q34	12708	635.4	0.2118
Mixed-LSP + Q34	9154	457.7	0.1526
“Q.-search Kabal”	4262	213.1	0.0710

Table 6.14: Total computational complexity on a DSP56001 implementation, to obtain quantized LSPs, using either the methods of Kabal’s, Mixed-LSP or “quantized-search Kabal”.

6.10. Program Listings

The listings for the C, Matlab, and DSP56001 assembly programs, used for the simulation and evaluation of the different LSP calculation methods, are given in [Gras97b].

6.11. Further Work

In the case of LPC of order $p = 8$, using Kabal’s method (see § 5.9) the LSPs can be calculated as the roots of two fourth-order polynomials, $P'_8(x)$ and $Q'_8(x)$. As the LSPs are real, different, and inside the interval $(-1,+1)$, the optimized calculation and ordering of the roots of $D_{10}(x)$, given in Appendices D.4 and D.5 (see § 6.1), can be used for efficient LSP calculation from 8-th order LPC coefficients.

For higher order LPC systems, this efficient calculation of LSP from 8-th order LPC can be combined with the mixed LSP/Parcor representation proposed in [Chan94], for obtaining both, better quantization performance and computational efficiency.

Although “quantized-search Kabal" algorithm is more efficient than Mixed-LSP and Kabal’s, its utilization is tied to the 34-bits scalar quantizer of the CELP FS1016. Nevertheless, this algorithm could find application in spectral quantization systems in which this 34-bit scalar quantizer is used as preprocessing, before further scalar quantization [Sade95a] or vector quantization [Sade95b].

6.12. Conclusions and Summary of the Chapter

In this chapter two novel efficient algorithms for LSP calculation from LPC coefficients, named Mixed-LSP and “quantized-search Kabal”, were presented. These algorithms were compared with the algorithms of Kabal, Chan and Saoudi from the point of view of accuracy, reliability and computational complexity.

It was found that Kabal's algorithm is more accurate and computationally efficient than Saoudi's and Chan's algorithms.

The proposed Mixed-LSP algorithm is computationally less expensive but also less accurate than Kabal's method. On the other hand, the accuracy of the Mixed-LSP algorithm is sufficient for speech coding applications using the 34-bit quantizer of the CELP FS1016.

The accuracy of the Mixed-LSP algorithm can be improved using more bisections, at the cost of 10*(4·Mult+10·Add) operations per bisection, and can also be decreased, trading precision against computational complexity. In Kabal's method, the accuracy can be increased at the cost of more bisections, but it cannot be decreased, reducing complexity, without affecting the zero-crossing search.

The Mixed-LSP algorithm can be used not only with scalar quantization but also with vector quantization or predictive quantization. Its utilization is limited to an LPC order of 10. Nevertheless, an LPC order of 10 is used in nearly all the standard and emerging low bit rate narrowband speech coders. In the case of a higher order LPC, Kabal's algorithm should be used, or, alternatively, a mixed LSP/Parcor representation combined with Mixed-LSP algorithm.

"Quantized-search Kabal" algorithm is more efficient than Mixed-LSP and Kabal's, but is tied to the utilization of the 34-bits non-uniform scalar quantizer of the CELP FS1016.

The results obtained in these chapter are used in the DSP56001 implementation of the CELP FS1016 spectral analysis block explained the next chapter.

6.13. References

[Camp89] J. Campbell et al., "The New 4800 bps Voice Coding Standard", *Proc. of Military and Government Speech Technology*, pp. 64-70, 1989.

[CELP3.2a] The US FS1016 based 4800 bps CELP voice coder, Fortran and C simulation source codes, version 3.2a (CELP 3.2a). Available by ftp from ftp.super.org and other sites.

[Chan91] C. Chan and K. Law , "An Algorithm for Computing the LSP Frequencies Directly from the Reflection Coefficients", *Proc. European Conference on Speech Communication and Technology*, EUROSPEECH'91, pp. 913-916, 1991.

[Chan94] C. Chan, "Efficient Quantization of LPC Parameters Using a Mixed LSP/Parcor Representation", *Proc. EUSIPCO'94*, Vol. 1, pp. 939-942, 1994.

[Chan95] C. Chan, "An Extremely Low Complexity CELP Speech Coder for Digital Telephone Answering Device Applications", *Proc. Int. Conf. on Signal Processing Applications and Technology*, ICSPAT'95, Vol. 2, pp. 1892-1896, 1995.

[Fede91] "Federal Standard 1016, Telecommunications: Analog to Digital Conversion of Radio Voice by 4,800 bit/second Code Excited Linear Prediction (CELP)", National Communications Systems, Office of Technology and Standards, Washington, DC20305-2010, 1991.

[Garo90] J. Garofolo et al., "Darpa TIMIT, Acoustic-phonetic Continuous Speech Corpus CD-ROM", National Institute of Standards and Technology, NISTIR 493, Oct. 1990.

[Gras94] S. Grassi, A. Heubi, M. Ansorge, and F. Pellandini, "Study of a VLSI Implementation of a Noise Reduction Algorithm for Digital Hearing Aids", *Proc. EUSIPCO'94*, Vol.3, pp. 1661-1664, 1994.

[Gras95] S. Grassi, *Simulation of Fixed-point Quantization Effects on DSP Algorithms*, IMT Report No 375 PE 03/95, University of Neuchâtel, IMT, 1995.

[Gras97a] S. Grassi, A. Dufaux, M. Ansorge, and F. Pellandini, "Efficient Algorithm to Compute LSP Parameters from 10-th order LPC Coefficients", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol. 3, pp. 1707-1710, 1997.

[Gras97b] S. Grassi, *DSP56001 Implementation of the Spectral Analysis and Quantization for the CELP FS1016 Speech Coder*, IMT Report No 421 PE 10/97, University of Neuchâtel, IMT, Oct. 1997.

[Kaba86] P. Kabal and P. Ramachandran, "The Computation of Line Spectral Frequencies Using Chebyshev Polynomials", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 34, No. 6, pp. 1419-1426, 1986.

[Pali95b] K. Paliwal, "Interpolation Properties of Linear Prediction Parametric Representations", *Proc. European Conference on Speech Communication and Technology*, EUROSPEECH'95, Vol. 2, pp. 1029-1032, 1995.

[Sade95a] H. Sadegh Mohammadi and W. Holmes, "Predictive Delta Adaptive Scalar Quantization: an Efficient Method for Coding the Short-term Speech Spectrum", *Proc. European Conference on Speech Communication and Technology*, EUROSPEECH'95, Vol. 2, pp. 1045-1048, 1995.

[Sade95b] H. Sadegh Mohammadi and W. Holmes, "Low Cost Vector Quantization Methods for Spectral Coding in Low Rate Speech Coders", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'95, Vol. 1, pp. 720-723, 1995.

[Saou92] S. Saoudi and J. Boucher, "A New Efficient Algorithm to Compute the LSP Parameters for Speech Coding", *Signal Processing*, Elsevier, Vol. 28, No. 2 , pp. 201-212, 1992.

[Saou95] A. Goalic and S. Saoudi, "An Intrinsically Reliable and Fast Algorithm to Compute the Line Spectrum Pairs (LSP) in Low Bit Rate CELP Coding", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'95, Vol. 1, pp. 728-731, 1995.

[Tohk78] Y. Tohkura et al., "Spectral Smoothing Technique in Parcor Speech Analysis-synthesis", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 26, No. 6, pp. 587-596, 1978.

Chapter 7 DSP56001 Implementation of the CELP FS1016 Spectral Analysis and Quantization

In this chapter, the optimization methodology explained in Chapter 3 is used for an efficient real-time DSP56001 implementation of the CELP FS1016 short-term spectral analysis block. The concepts to understand the algorithms and algorithmic optimization used in this chapter are given in Chapter 5 and 6.

7.1. Short-term Spectral Analysis and Quantization in the CELP FS1016 Coder

In Section 5.11 it is explained how spectral analysis and quantization is done in the CELP FS1016. The different functional blocks which were implemented on the DSP56001 are shown in Figure 7.1. The shadowed regions correspond to the following subsystems:

(1) Calculation of bandwidth expanded LPC.

(2) LSP calculation and quantization.

(3) LSP interpolation and conversion to LPC.

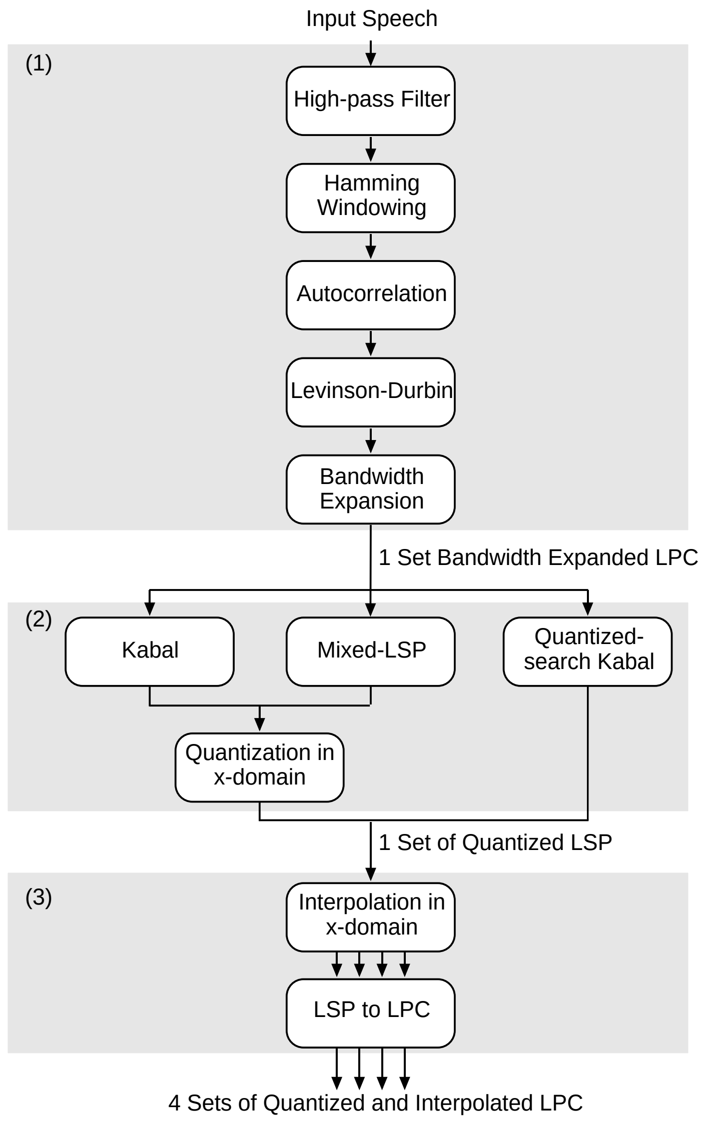


Figure 7.1: Different functional blocks of the CELP FS1016 spectral analysis and quantization implemented on DSP56001.

The details on the DSP56001 implementation of these three subsystems are given in Sections 7.4 to 7.6.

An explanation of the DSP56001 and its arithmetic is given in [MOTO90] and [MOTO93]. The simulation of the DSP56001 quantization effects using C language and Matlab is explained in Section 3.10.

In [Segu97], some of the functional blocks shown in Figure 7.1 were implemented on a DSP56001. This implementation is inefficient because it does not always exploit the parallelism and other resources of the DSP56001. Nevertheless, the study of the quantization effects done in [Segu97], using the methodology described in Chapter 3, as well as the algorithm transformations to account for these quantization effects are valid. This information was partially used in the implementation described in [Grass97b] which is presented in this chapter.

7.2. Testing the Implemented Blocks

All the functional blocks shown in Figure 7.1 were coded in C language (using double-precision floating-point arithmetic) and interfaced as Matlab functions. These programs are used to characterize the "infinite precision" behavior of each block and are the “reference system” to evaluate the degradation in performance in the case of a fixed-point implementation.

Each functional block was then implemented on a DSP56001 in assembly language and on a workstation in C language, including the DSP56001 arithmetic effects (see § 3.10). This C program is thus a “model” of the corresponding DSP56001 implementation. Each of these C “models” is also interfaced as a Matlab function.

It was checked that each DSP56001 implementation and its corresponding C model have exactly the same output under the same input. This verification was carried out using the whole TIMIT database. After that, the C model is used to measure the performance of the DSP56001 implementation.

The advantage of this approach is that the C models are easily interfaced (within Matlab) with the rest of the system. The C models can also be used to try out different implementation options before actually implementing them on the DSP56001.

Each model is introduced in the reference system to obtain a modified system, and the deviation from the “infinite-precision” behavior is observed using spectral distortion measure (see § 5.4). In this context, there are two possible ways to use this measure. The first way is comparing two sets of LPC coefficients, the one produced with the reference system and the other with the modified system, using Equations 5.15 to 5.17. The second way is explained as follows.

Spectral distortion measure was used in Section 6.2 to evaluate LSP-quantization in the “x-domain” on the whole TIMIT database. The LSPs were calculated with high accuracy and quantized using the 34-bit scalar quantizer of the CELP FS1016. All the operations were carried out using double precision floating-point arithmetic. The measured average spectral distortion, due to the 34-bit scalar quantizer, and the percentage of outliers are given in Table 7.1. This corresponds to the “infinite-precision” behavior of this system. The characterization of this “infinite-precision” behavior was also done for other systems which use Kabal’s, Mixed-LSP, and “quantized-search Kabal” methods for LSP calculation, and is given in Table 7.1. In all cases, double precision floating-point arithmetic was used.

The same measurement will be repeated for each modified system under test, to observe the deviation from the values given in Table 7.1.

Algorithms to obtain the quantized LSPs	*Ave. SD (dB)*	*% 2-4 dB outliers*	*% >4 dB outliers*
High accuracy + quant. in x	1.53287	12.34501	0.18884
Kabal + quant. in x	1.53288	12.34532	0.18884
Mixed-LSP + quant. in x	1.53308	12.36305	0.18853
Q.-search Kabal	1.53295	12.35014	0.18946

Table 7.1: Quantization performance of different “reference systems” differing in their LSP calculation method, and using double-precision floating-point operations.

The reader should not confuse “LSP-quantization” which is a (desired) functionality to be implemented, with the (undesired)

quantization effects introduced in the implemented functional blocks due to the use of fixed-point arithmetic.

7.3. Measure of Computational Complexity

The maximum number of clock cycles and time needed for processing one frame of 30 ms, are used to measure the computational complexity of the different blocks. The clock frequency is 20 MHz.

The computational complexity is also given in MIPS (million instructions per second) calculated assuming an “average instruction” of 2 cycles. The computational power of a DSP56001 at 20 MHz is thus 10 MIPS.

7.4. Calculation of Bandwidth-expanded LPC

This subsystem does the calculation of the bandwidth-expanded LPC vectors, as specified in the CELP FS1016 (see § 5.11). It contains the following functional blocks: high-pass filtering, 30 ms Hamming windowing, calculation of the autocorrelation coefficients, Levinson-Durbin recursion, and 15 Hz bandwidth expansion [CELP3.2a]. The computational load for the DSP56001 implementation of these blocks is given in Table 7.2.

Algorithm	*Number of cycles*	*Execution time (μs)*	*MIPS*
High-pass filter	3390	169.5	0.0565
Windowing	984	49.2	0.0164
Autocorrelation	6158	307.9	0.1026
Levinson-Durbin	1592	79.6	0.0265
Bandwidth Expansion	52	2.6	0.0009
Total	12176	608.8	0.2029

Table 7.2: Computational load for the DSP56001 implementation of the calculation of the bandwidth expanded LPC coefficients.

High-pass Filter

The input signal is filtered with the second order high-pass digital Butterworth filter, with a 100 Hz cut-off frequency [CELP3.2a]. The transfer function of the filter is given by:

$$H(z) = 0.9459 \cdot \frac{1 - 2z^{-1} + z^{-2}}{1 - 1.889033z^{-1} + 0.8948743z^{-2}} \qquad (7.1)$$

This filter was implemented using the canonical direct form II [Proa89] shown in Figure 7.2. The gain G = 0.9459 ensures an amplification of slightly less than 0 dB in the pass-band.

The coefficients of the filter are scaled by a factor of two, in order to represent them in the fractional arithmetic of the DSP56001. By simulation on the TIMIT database, it was found that a scaling of 1/32 at the input is needed to avoid overflow in the internal nodes of the filter [Segu97]. This scaling is compensated at the output of the filter, to avoid unnecessary loss of dynamic range. The resulting structure is shown in Figure 7.3.

The filtering is done at every speech sample (240 times per frame). It is thus essential to carefully optimize this block, as every DSP instruction inside the time loop of the filter would add 480 clock cycles per frame to the execution time.

In the DSP56001 there is no barrel shifter, and a scaling needs 1 instruction (2 clock cycles) per factor of two. On the other hand, it is possible to realize a scaling by 2^{24} by an internal transfer in the accumulator [MOTO93]. Thus the structure of the filter was modified as shown in Figure 7.4. The operations in the shadowed regions are done using 4 MAC (multiply accumulate). There is a total of seven arithmetic instructions per sample (scaling by 2^{24} and delays are done with data transfer). This corresponds to the minimum possible number of instructions in the time loop, as the DSP56001 has only one ALU. The scheduling of the operations was carefully optimized to take advantage of the parallelism of the DSP56001, performing all the data transfer in parallel with the arithmetic instructions. The time loop contains thus seven instructions (14 cycles).

The high-passed values are stored in X memory, at the same location of the input data.

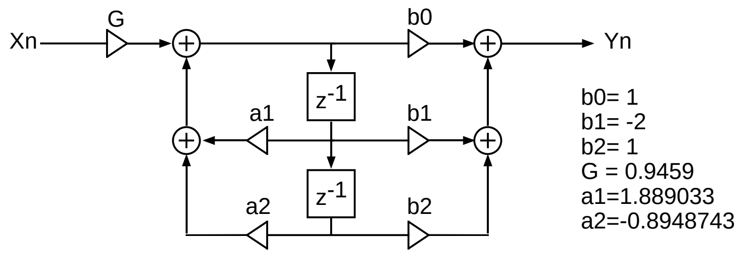


Figure 7.2: Canonical direct form II structure of the high-pass filter.

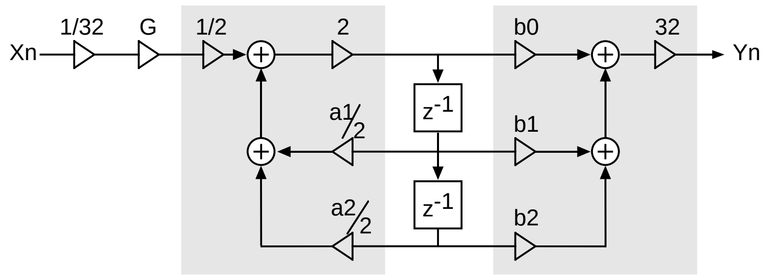


Figure 7.3: Scaling factors in the high-pass filter.

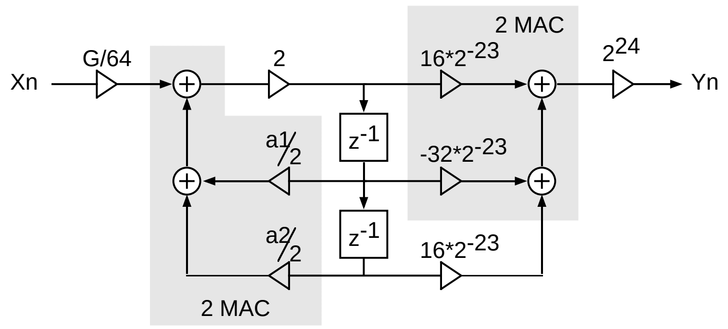


Figure 7.4: High-pass filter, as implemented on the DSP56001.

In the implementation of all the functional blocks shown in Figure 7.1 that are explained in the following sections, there is a kind of optimization that was always done (although not always mentioned explicitly): *as much of the data transfer as possible is done in parallel to the arithmetic instructions*. The high-pass filter implementation is a good example of this kind of optimization.

Windowing

As the 240 coefficients of the Hamming window are in the range $[-1,+1)$ no scaling is needed. These coefficients are quantized to 24 bits for storage in the DSP56001 memory. The quantization effects introduced are negligible. As the high-passed signal values are stored in X memory the window coefficients are stored in Y memory to take advantage of the parallelism of the DSP56001. One instruction is needed for performing multiplication and fetching the next sample and its corresponding window coefficient. Another instruction is used to store the windowed samples in both X and Y memory, for efficient implementation of the next functional block. A total of two instructions (4 cycles) per sample is needed.

Calculation of the Autocorrelation Coefficients

The autocorrelation coefficients $\{r_0, r_1, \ldots, r_{10}\}$ needed by the Levinson-Durbin recursion (see § 5.2) are calculated as:

$$r_k = \sum_{n=k}^{239} s_{hw}(n) \cdot s_{hw}(n-k) \tag{7.2}$$

here $\{s_{hw}(n)\}$ are the high-passed and windowed speech samples, which are in the range $[-1,+1)$. Thus, the autocorrelation coefficients are limited by:

$$|r_k| < |r_0| \le \sum_{n=0}^{239} w(n)^2 = 94.9850 < 2^7 \qquad k = 1,\ldots,10 \tag{7.3}$$

When the inner product of Equation (7.2) is done using MAC operations and the partial sum is accumulated in the DSP56001

accumulators A or B, there is no overflow. The 56-bit dynamic range of the accumulator registers can largely accommodate the dynamic range needs of this operation.

A problem arises when the correlation coefficient have to be stored in 24-bit X or Y memory, or used in the 24-bit input of the multiplier. It was found by simulation [Segu97] that the dynamic range of the autocorrelation coefficients is larger than 24 bits. This is due to the signal multiplication observed in Equation (7.2) that doubles the dynamic range of the signals. The problem could be overcome by storing the variables in the 48-bit concatenated XY memory. But, as this coefficients are used in the Levinson-Durbin recursion, time-consuming long-word multiplications and divisions would be needed [Moto93].

An efficient solution is to reduce the dynamic range of the $\{r_k\}$ by means of normalization steps. The first autocorrelation coefficient or energy, r_0, is calculated on the accumulator A and then normalized using 23 normalization instructions of the DSP56001, obtaining a mantissa m_0 and exponent e_0:

$$\begin{aligned} r_0 &= m_0 \cdot 2^{e0} \\ 0.5 \le m_0 &< 1 \end{aligned} \tag{7.4}$$

The mantissa is stored in a 24-bit register for further use, and the exponent is used to scale the correlation coefficients:

$$\begin{aligned} r'_k &= r_k \cdot 2^{-e0} \qquad \text{for } k = 1,\ldots,10 \\ r'_0 &= r_0 \cdot 2^{-e0} = m_0 \end{aligned} \tag{7.5}$$

Note that this scaling does not change the functionality of the Levinson-Durbin recursion (see Equation 5.12). From Equations (7.3) to (7.5) it is seen that:

$$|r'_k| < r'_0 = m_0 < 1 \tag{7.6}$$

the scaled correlation coefficients are within the range $[-1,+1)$. Also, the scaling reduces the dynamic range of the correlation coefficient so that the $\{r'_k\}$ can be represented with 24-bit for storage and further use as input of the multiplier.

The calculation of each of the 11 correlation coefficients uses a loop repeated (240-k) times. Thus any extra instruction inside this loop would add 5170 clock cycles per frame to the execution time. It is thus essential to reduce the number of instructions in the loop to the bare minimum. The windowed samples were stored in both X and Y memory in the windowing operation, to take advantage of the parallelism of the DSP56001. In this way, only one instruction (2 cycles) is needed inside the loop, for performing multiplication and accumulation and fetching the next samples $s_{hw}(n)$ and $s_{hw}(n-k)$.

Levinson-Durbin Recursion

The Levinson-Durbin recursion is used to calculate the 10-th order LPC coefficients $\{a_{10}(k)\}$ from the autocorrelation coefficients as shown in Equation (5.12). The autocorrelation coefficients were calculated and dynamically scaled as explained in the previous subsection, to obtain a set of bounded autocorrelation coefficients, $\{r'_k\}$, which have reduced dynamic range needs. The use of this scaled correlation coefficients also decreases the dynamic range needed in the Levinson-Durbin recursion and makes it easier to prevent overflows.

It was found by simulation that the final LPC coefficients $\{a_{10}(k)\}$ need a scaling of 1/16 to be represented in the fractional arithmetic of the DSP56001. The Levinson-Durbin recursion was modified to include this scaling, as shown in Equation (7.7).

Bandwidth Expansion

In bandwidth expansion (see § 5.3) each LPC coefficient $a_{10}(k)$ is multiplied by a factor 0.994^k. This operation is similar to the windowing. The ten expansion factors are quantized to 24-bit and stored in Y memory, while the LPC coefficients are stored in X memory. One instruction is needed for performing multiplication and fetching the next $a_{10}(k)$ and its corresponding expansion factor. Another instruction is used to store the expanded LPC coefficients in X memory, overwriting the original LPC coefficients.

$$k_1 = -\frac{r'_1}{r'_0}$$

$$\varepsilon_1 = r'_0(1-k_1^2)$$

$$k_2 = -\frac{r'_2 + k_1 \cdot r'_1}{\varepsilon_1}$$

$$a_2(1) = \frac{k_1 + k_2.k_1}{16}, \quad a_2(0) = \frac{1}{16}, \quad a_2(2) = \frac{k_2}{16}$$

for $m = 3,\ldots,10$:

$$\left[\begin{array}{l} \varepsilon_m = \varepsilon_{m-1}\left(1-k_m^2\right) \\ a_m(m) = -\dfrac{\dfrac{r'_m}{16} + \sum_{i=1}^{m-1} a_{m-1}(i).r'_{m-i}}{\varepsilon_{m-1}}, \quad a_m(0) = \dfrac{1}{16} \\ k_m = 16 \cdot a_m(m) \\ a_m(j) = a_{m-1}(j) + k_m . a_{m-1}(m-j) \qquad \text{for} \quad 1 \le j \le m \end{array}\right. \qquad (7.7)$$

Experimental Evaluation of the Calculation of Bandwidth Expanded LPC

Spectral distortion measure was used to compare the bandwidth expanded LPC obtained with the DSP56001 implementation and with a C (double precision floating-point) program. The measured average spectral distortion was 0.013 dB.

Also, a reference system using high accuracy method for LSP calculation and double precision floating-point arithmetic is compared with a modified system. The modified system is obtained by substituting, in the reference system, the calculation of bandwidth expanded LPC by its corresponding DSP56001 model. The quantization performance is measured in both systems and given in Table 7.3. It is observed that the degradation introduced in the performance is negligible.

	Ave. SD (dB)	***% 2-4 dB outliers***	***% >4 dB outliers***
Reference system	1.53287	12.34501	0.18884
Modified system	1.53310	12.34874	0.18899

Table 7.3: Quantization performance of the “reference system” and the modified system which uses DSP56001 LPC calculation.

7.5. LSP Calculation and Quantization

Algorithmic optimization of LSP calculation and quantization is discussed in Chapter 6. Based on comparisons in accuracy, reliability and computational complexity, three LSP calculation algorithms were retained for DSP56001 implementation. These algorithms are Kabal’s, Mixed-LSP and “quantized-search Kabal”.

As the LSPs obtained with Kabal’s and Mixed-LSP algorithm are in the “x-domain”, the quantization is done in this domain for computational saving (see § 6.2). Furthermore, in the “quantized-search Kabal“ algorithm, the LSP calculation and quantization are embedded for efficiency (see § 6.3).

The computational load for the DSP56001 implementation of the different functional blocks is given in Table 7.4. Details on the implementation of these blocks are given in the next subsections. In Table 7.5, the total computational load for the three implemented ways to obtain quantized LSP is given.

Algorithm	***Number of cycles***	***Execution time (µs)***	***MIPS***
Kabal	10540	527.0	0.1757
Mixed-LSP	6986	349.3	0.1164
“Q.-search Kabal”	4262	213.1	0.0710
Quantization (Q34)	2168	108.4	0.0361

Table 7.4: Computational load for the DSP56001 implementation of LSP-quantization and Kabal’s, Mixed-LSP and “quantized-search Kabal” algorithms for LSP calculation.

Algorithm	***Number of cycles***	***Execution time [µs]***	***MIPS***
Kabal + Q34	12708	635.4	0.2118
Mixed-LSP + Q34	9154	457.7	0.1526
“Q.-search Kabal”	4262	213.1	0.0710

Table 7.5: Total computational load for DSP56001 implementation, to obtain quantized LSPs, using either the methods of Kabal’s, Mixed-LSP or “quantized-search Kabal”.

Kabal’s Algorithm

Kabal’s algorithm for LSP calculation is explained in Section 5.9. The main arithmetic task in this algorithm is polynomial evaluation which is done with the efficient recursion of Equation (5.32). It was found, by simulation on the TIMIT database, that a scaling of 1/64 is needed on the coefficients $\{q'_i\}$ and $\{p'_i\}$. As the LPC coefficients $a_{10}(k)$ were already scaled by 1/16 in the Levinson-Durbin recursion, they are scaled by 1/4, before using them in the calculation of $\{q'_i\}$ and $\{p'_i\}$ with Equation 5.27. The recursion of Equation 5.32 was modified to account for this scaling:

$$\begin{aligned}
\text{temp}_1 &= 2\cdot x_i/64 + p'_1 \\
\text{temp}_2 &= 2\cdot x_i\cdot \text{temp}_1 - 1/64 + p'_2 \\
\text{temp}_0 &= 2\cdot x_i\cdot \text{temp}_2 - \text{temp}_1 + p'_3 \\
\text{temp}_1 &= 2\cdot x_i\cdot \text{temp}_0 - \text{temp}_2 + p'_4 \\
\text{temp}_2 &= x_i\cdot \text{temp}_1 - \text{temp}_0 + 0.5\cdot p'_5
\end{aligned} \tag{7.8}$$

For the zero-crossing search and bisections, only the sign of temp_2 is used. On the other hand, a linear interpolation is done on the final interval, using the last two values calculated with Equation (7.8). As this two values are close to zero, they are scaled up by 32, before moving them to a 24-bit register to avoid excessive round-off error in the linear interpolation.

Experimental Evaluation of Kabal's Algorithm Implementation

A reference system using Kabal's method for LSP calculation and double precision floating-point arithmetic is compared with a modified system. The modified system is obtained by substituting, in the reference system, Kabal's algorithm by its corresponding DSP56001 model.

The LSPs obtained with the reference and the modified system were converted to LPC, and compared using spectral distortion measure. The measured average spectral distortion was 0.0023 dB.

Quantization performance is measured in both the reference and the modified system and given in Table 7.6. It is observed that there is no degradation in the performance due to the DSP56001 implementation of Kabal's algorithm.

	Ave. SD (dB)	***% 2-4 dB outliers***	***% >4 dB outliers***
Reference system	1.53288	12.34532	0.18884
Modified system	1.53287	12.34454	0.18884

Table 7.6: Quantization performance of the "reference system" and the modified system which uses DSP56001 Kabal's algorithm.

Mixed-LSP

Mixed-LSP algorithm for LSP calculation is explained in Section 6.1. In this algorithm the roots of the 4-th order polynomial $D_{10}(x)$ are calculated and ordered, to obtain the intervals containing a zero-crossing. The position of the zero-crossings are refined by five bisections and a final linear interpolation, similarly to Kabal's method.

The root calculation and ordering is explained and optimized in Appendix D.4 to D.5. As this calculation is a highly non-linear algorithm, care must be taken in scaling the internal nodes of this algorithm, as the propagation of the amplification is non-linear.

First a simulation was done to determine the optimum scaling, and then this scaling was partially included in the algorithm, taking into account its (non-linear) propagation

through the algorithm. The algorithm was then re-simulated, to determine again the needed scaling. Several steps of this optimization were done, until an algorithm that has all the internal signals and parameters in the range $[-1,+1)$ was obtained. It was also checked that the dynamic range did not exceed 24 bits.

As there is no square root operation in the DSP56001, the five square roots were done using a 4-th order polynomial fitting in the region $0.5 \le x \le 1$:

$$f(x) = \sqrt{x} \approx \rho_0 + \rho_1 \cdot x + \rho_2 \cdot x^2 + \rho_3 \cdot x^3 + \rho_4 \cdot x^4 \qquad 0.5 \le x \le 1 \tag{7.9}$$

The argument of the square root is normalized using 23 normalization instructions, obtaining a mantissa x and exponent e:

$$\arg = x \cdot 2^e, \quad 0.5 \le x < 1 \tag{7.10}$$

The square root of the mantissa is done with the following polynomial evaluation:

$$\begin{aligned} temp &= \rho_3 + x \cdot \rho_4 \\ temp &= \rho_2 + x \cdot temp \\ temp &= \rho_1 + x \cdot temp \\ temp &= \rho_0 + x \cdot temp \end{aligned} \tag{7.11}$$

and the result is multiplied by $\sqrt{2}$ if the exponent is odd, and shifted by half of the exponent. In total, a square root evaluation needs 140 clock cycles.

Once the intervals containing the zero-crossings are found, the rest of the implementation (bisections and linear interpolation) is done as in Kabal's algorithm, previously explained.

Experimental Evaluation of Mixed-LSP Implementation

A reference system, using Mixed-LSP method for LSP calculation and double precision floating-point arithmetic, is compared with a modified system. The modified system is obtained by substituting, in the reference system, Mixed-LSP algorithm by its corresponding DSP56001 model.

The LSPs obtained with the reference and the modified system were converted to LPC, and compared using spectral distortion measure. The measured average spectral distortion was 0.0011 dB.

Quantization performance is measured in both the reference and the modified system and given in Table 7.7. It is observed that there is no degradation in the performance.

	Ave. SD (dB)	***% 2-4 dB outliers***	***% >4 dB outliers***
Reference system	1.53308	12.36305	0.18853
Modified system	1.53308	12.36274	0.18853

Table 7.7: Quantization performance of the "reference system" and the modified system which uses DSP56001 Mixed-LSP algorithm.

Quantized-search Kabal

"Quantized-search Kabal" algorithm for LSP calculation (and quantization) is explained in Section 6.3. In this algorithm the search for zero crossings is done on the grid defined by the quantization levels of the 34-bit quantizer of Figure 5.5. (converted to the "x-domain").

The values of the quantization tables of Figure 5.5 were quantized to 24 bits for storage in the DSP56001 memory. The quantization effects introduced are negligible. This tables were converted to the "x-domain" using the mapping $\xi_i = \cos(2\pi f_i)$, thus the values are in the range $[-1,+1)$, and no scaling is needed.

Polynomial evaluation is done using the efficient Kabal's recursion of Equation (5.32), thus the scaling and quantization issues are the same as in Kabal's algorithm, previously explained. In particular, the modified recursion given in Equation (7.8) is used. In appendix E.1 it is shown that the maximum possible number of polynomial evaluations is 71 (while it is 150 in Kabal's algorithm). This number is used in the

calculation of the maximum number of clock cycles and time needed for processing one frame of 30 ms.

To avoid expensive comparisons, the quantization tables of Figure 5.5 were modified to include, with each quantization level, an index (offset) to the first "allowed" value of the next quantization table. Also, some flags indicating conditions such as "first element of the table", "last element of the table" and "particular case of qx_5" ($qx_5 = 0.2563$, $qx_5 = 0.0392$, or $qx_5 = -0.1175$) are stored in the quantization table, to be used in the control flow of the algorithm.

The flow control of the algorithm was simplified by the use of two flags, called "mark" and "last_mark", which are used to keep track of the conditions needed in the correction criteria (see § 6.3). The use of these two flags is explained as follows.

The flag "last_mark" is cleared at the beginning of the algorithm. This flag is used to store the value of "mark" at the end of the last search for a quantized LSP.

At the beginning of the search for a quantized LSP, the flag "mark" is set to one, to indicate that the first allowed value of the quantization table is being tested. This value of the quantization table is used in the polynomial evaluation of Equation (7.8). If the zero-crossing is not detected, the flag "mark" is cleared, indicating that the zero crossing is not at the first allowed value, and the search proceeds, testing the next value of the quantization table, until either the zero crossing is detected, or the last element of the table is reached.

Then, the flag "mark" is used to decide whether a "single-correction" ("mark" = 0) or a "coupled-correction" ("mark"=1) is to be tested.

If the position of the quantized LSP is single- or coupled-corrected, the value of "mark" is set to one, to indicate that a coupled correction cannot be done in the next search.

If the 5-th quantized LSP takes one of these three values: $qx_5 = 0.2563$, $qx_5 = 0.0392$, or $qx_5 = -0.1175$, the flag "mark" is set to one, to indicate that a coupled correction cannot be done in the next search.

Then the flag "mark" is copied onto "last_mark" to be used in the next search (if "last_mark"=1, coupled correction cannot be done).

The resulting algorithm is robust and simple in its implementation. In Table 7.5 it is observed that this algorithm needs 66 % less cycles than Kabal's algorithm + quantization and 53 % less cycles than Mixed-LSP algorithm + quantization.

Experimental Evaluation of Q.-search Kabal Implementation

A reference system, using "quantized-search Kabal" for LSP calculation and double precision floating-point arithmetic, is compared with a modified system. The modified system is obtained by substituting, in the reference system, "quantized-search Kabal" algorithm by its corresponding DSP56001 model.

The LSPs obtained with the reference and the modified system were converted to LPC, and compared using spectral distortion measure. The measured average spectral distortion was 0.0072 dB.

Quantization performance is measured in both the reference and the modified system and given in Table 7.8. It is observed that there is no degradation in the performance.

	Ave. SD (dB)	***% 2-4 dB outliers***	***% >4 dB outliers***
Reference system	1.53295	12.35014	0.18946
Modified system	1.53292	12.34967	0.18946

Table 7.8: Quantization performance of the "reference system" and the modified system which uses DSP56001 "quantized-search Kabal" algorithm.

LSP Quantization in the "x-domain"

The algorithm for LSP quantization in the "x-domain" is explained in Section 6.4. This algorithm is used to quantize the LSPs which were first calculated with the methods of Kabal or Mixed-LSP.

This algorithm for LSP quantization is very similar to the "quantized-search Kabal" algorithm, except that the actual

LSPs are known. Thus the single- and coupled-correction criteria of Equations (6.4) and (6.9) are used.

Efficient DSP56001 implementation is obtained with the use of flags and offsets, as it is done in the implementation of "quantized-search Kabal" explained previously.

Experimental Evaluation of LSP Quantization in the "x-domain"

A reference system using high accuracy method for LSP calculation and double precision floating-point arithmetic is compared with a modified system. The modified system is obtained by substituting, in the reference system, LSP quantization by its corresponding DSP56001 model.

Both systems gave exactly the same LSP indices. The only source of error is due to the 24-bit representation (for storage in the DSP56001 memory) of the quantization tables.

Quantization performance is measured in both the reference and the modified system and given in Table 7.9. It is observed that there is no degradation in the performance.

	Ave. SD (dB)	***% 2-4 dB outliers***	***% >4 dB outliers***
Reference system	1.53287	12.34501	0.18884
Modified system	1.53284	12.34516	0.18884

Table 7.9: Quantization performance of the "reference system" and the modified system which uses DSP56001 LSP-quantization.

7.6. LSP Interpolation and Conversion to LPC

In this subsystem, two adjacent sets of quantized LSP parameters are interpolated obtaining four sets of LSP parameters, which are then converted to LPC coefficients to be used in the synthesis filter (see § 5.11). The computational load for the DSP56001 implementation of these blocks is given in Table 7.10.

Algorithm	***Number of cycles***	***Execution time (μs)***	***MIPS***
Interpolation	236	11.8	0.0039
4 set LSP to 4 set LPC	906	45.3	0.0151

Table 7.10: Computational load for the DSP56001 implementation of LSP interpolation and conversion to LPC.

Algorithmic optimization of LSP interpolation and conversion to LPC is discussed in Section 5.10 and 6.8.

In Section 6.8 it is shown that LSP interpolation can be done in the “x-domain” instead of “ω-domain”. As the LSPs obtained with the methods of Kabal, Mixed-LSP and “quantized-search Kabal” are in the “x-domain”, the computationally expensive conversion from “x-domain” to “ω-domain” for interpolation, and then from “ω-domain” to “x-domain” for LSP to LPC transformation is avoided.

In Section 5.10, three methods for LSP to LPC transformation are discussed. It is shown that Kabal’s method is the least expensive. Besides, this algorithm is highly regular and numerically stable which is advantageous for efficient implementation.

As the quantized LSPs used in the linear interpolation are in the range $[-1,+1)$, the resulting interpolated LSPs are also in this range. The weights used in the interpolation, given in Table 5.1 are also in the range $[-1,+1)$. Furthermore, the dynamic range needs of the interpolation operation can largely be accommodated in 24-bits registers.

In LSP to LPC conversion, it was found that a scaling of 1/32 is needed in the recursion of Equation C.7, which is modified as follows:

$$
\begin{aligned}
c_{10} &= -x_1/32 & c'_{10} &= -x_2/32\\
c_{20} &= -2\cdot x_3\cdot c_{10} + 1/32 & c'_{20} &= -2\cdot x_4\cdot c'_{10} + 1/32\\
c_{21} &= 2\cdot(c_{10} - x_3/32) & c'_{21} &= 2\cdot(c'_{10} - x_4/32)\\
c_{30} &= -2\cdot x_5\cdot c_{20} + c_{21} & c'_{30} &= -2\cdot x_6\cdot c'_{20} + c'_{21}\\
c_{31} &= 2\cdot(c_{20} - x_5\cdot c_{21}) + 1/32 & c'_{31} &= 2\cdot(c'_{20} - x_6\cdot c'_{21}) + 1/32\\
c_{32} &= c_{21} - 2\cdot x_5/32 & c'_{32} &= c'_{21} - 2\cdot x_6/32\\
c_{40} &= -2\cdot x_7\cdot c_{30} + c_{31} & c'_{40} &= -2\cdot x_8\cdot c'_{30} + c'_{31}\\
c_{41} &= 2\cdot(c_{30} - x_7\cdot c_{31}) + c_{32} & c'_{41} &= 2\cdot(c'_{30} - x_8\cdot c'_{31}) + c'_{32}\\
c_{42} &= c_{31} - 2\cdot x_7\cdot c_{32} + 1/32 & c'_{42} &= c'_{31} - 2\cdot x_8\cdot c'_{32} + 1/32\\
c_{43} &= c_{32} - 2\cdot x_7/32 & c'_{43} &= c'_{32} - 2\cdot x_8/32\\
c_{50} &= -2\cdot x_9\cdot c_{40} + c_{41} & c'_{50} &= -2\cdot x_{10}\cdot c'_{40} + c'_{41}\\
p'_4 &= 2\cdot(c_{40} - x_9\cdot c_{41}) + c_{42} & q'_4 &= 2\cdot(c'_{40} - x_{10}\cdot c'_{41}) + c'_{42}\\
p'_3 &= c_{41} - 2\cdot x_9\cdot c_{42} + c_{43} & q'_3 &= c'_{41} - 2\cdot x_{10}\cdot c'_{42} + c'_{43}\\
p'_2 &= c_{42} - 2\cdot x_9\cdot c_{43} + 1/32 & q'_2 &= c'_{42} - 2\cdot x_{10}\cdot c'_{43} + 1/32\\
p'_1 &= c_{43} - 2\cdot x_9/32 & q'_1 &= c'_{43} - 2\cdot x_{10}/32\\
p'_5 &= 2\cdot c_{50} & q'_5 &= 2\cdot c'_{50}
\end{aligned}
\tag{7.12}
$$

The scaling by 1/32 is done with a multiplication (2 cycles) instead of 5 shifts (10 cycles). The last terms of the recursion give the coefficients $\{q'_i\}$ and $\{p'_i\}$ scaled by a factor of 1/32. This coefficients are used to obtain the LPC coefficients:

$$
\begin{aligned}
a_1 &= p'_1 + q'_1 & a_{10} &= p'_1 - q'_1 + 1/16\\
a_2 &= (p'_2 + p'_1) + (q'_2 - q'_1) & a_9 &= (p'_2 + p'_1) - (q'_2 - q'_1)\\
a_3 &= (p'_3 + p'_2) + (q'_3 - q'_2) & a_8 &= (p'_3 + p'_2) - (q'_3 - q'_2)\\
a_4 &= (p'_4 + p'_3) + (q'_4 - q'_3) & a_7 &= (p'_4 + p'_3) - (q'_4 - q'_3)\\
a_5 &= (p'_5 + p'_4) + (q'_5 - q'_4) & a_6 &= (p'_5 + p'_4) - (q'_5 - q'_4)
\end{aligned}
\tag{7.13}
$$

Note that the obtained LPC coefficients are scaled by a factor of 1/16, which is needed to avoid overflows. This scaling should be taken into account in the implementation of the synthesis filter for the stochastic codebook search.

Experimental Evaluation of LSP Interpolation and Conversion to LPC

The reference system contains all the algorithms to obtain the four interpolated LPC, and uses high accuracy method for LSP calculation and double precision floating-point arithmetic.

The modified system is obtained by substituting, in the reference system, the LSP interpolation and conversion to LPC, by its corresponding DSP56001 model.

Both systems are compared using spectral distortion measure. The measured average spectral distortion was 0.00083 dB, and the maximum value of spectral distortion was 0.017 dB.

It is seen that the distortion introduced by this subsystem is really small, due to the numerical robustness of Kabal's method for LSP to LPC conversion and the fact that the interpolation was done in the "x-domain" avoiding (inexact) trigonometric calculations.

7.7. Total Computational Complexity

In Figure 7.1 it is observed that three possible ways to obtain the 4 sets of interpolated LPC, from 30 ms of speech (240 samples) were implemented. These three variants depend on the LSP calculation method used, which is either Kabal's, Mixed-LSP or "quantized-search Kabal". The total computational load for each variant is given in Table 7.11.

LSP calculation method	***Number of cycles***	***Execution time [μs]***	***MIPS***
Kabal	26026	1301.3	0.4338
Mixed-LSP	22472	1123.6	0.3745
"Q.-search Kabal"	17580	879.0	0.2930

Table 7.11: Total computational load for DSP56001 implementation to obtain the four sets of interpolated LPC, from a frame of 240 speech samples. Three variants are shown, depending on the LSP calculation method used.

7.8. Program Listings

The listings for the C, Matlab, and DSP56001 assembly programs are given in [Gras97b].

7.9. Further Work

The optimization of the implementation of the CELP FS1016 spectral analysis and quantization can be seen as a preparation for an optimal low power, small size custom VLSI implementation. The careful algorithmic optimization and transformation of the algorithms to improve the use of the dynamic range available and to prevent overflows, is of great importance for an optimal VLSI implementation.

In Table 7.11 it is observed that the most efficient implementation is the one that uses "quantized-search Kabal" for LSP calculation. This implementation should be chosen for the VLSI implementation.

The next step is to use the quantized C models to find the minimum wordlength needed, while keeping an acceptable performance. Preliminary work suggests that a word-length of 16 to 20 is needed as input of the multiplier and storage. The accumulator of the ALU should have 32 to 40 bits, plus 8 bits extension.

For the VLSI implementation, we propose an architecture similar to the one of the DSP56001, with a bit-parallel MAC and separate X and Y data memories, buses and address generation units. Only the subset of instructions used in the algorithms need to be implemented. As the sequencing of DSP56001 instructions was carefully optimized, it can be used directly in the controller of the designed unit.

In the efficient implementation of the CELP FS1016 it is also of great importance to optimize the search on the stochastic codebook, using methods such as the methods proposed in [Bour97] or [Chan95].

7.10. Conclusions and Summary of the Chapter

In this Chapter, the optimized implementation of the CELP FS1016 spectral analysis and quantization on a DSP56001 was presented. The key points for this optimized implementation are careful algorithmic optimization, study of the fixed-point quantization effects, and careful match between algorithms and target architecture.

Algorithmic optimization is discussed in Chapter 6, and deals with the choice and modification of the algorithms, as well as their optimal interrelation in the whole system.

The study of the quantization effects is done to find the optimum scaling, modifying the algorithms to include this scaling and also to improve the use of the dynamic range available. This is done by using normalization and denormalization at some localized nodes of the algorithms which have higher dynamic range needs.

The parallelism of the DSP56001 is exploited, trying to perform as much as the data transfer as possible in parallel to the arithmetic instructions.

Finally, it is shown that the optimal implementation on a fixed-point commercial DSP such as the DSP56001 can be seen as a preparation for an optimal low power, small size, custom VLSI implementation.

7.11. References

[Bour97] M. Bouraoui et al., "HCELP: Low Bit Rate Speech Coder for Voice Storage Applications", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'97, Vol. 2, pp. 739-742, 1997.

[CELP3.2a] The US FS1016 based 4800 bps CELP voice coder, Fortran and C simulation source codes, version 3.2a (CELP 3.2a). Available by ftp from ftp.super.org and other sites.

[Chan95] C. Chan, "An Extremely Low Complexity CELP Speech Coder for Digital Telephone Answering Device Applications", *Proc. Int. Conf. on Signal Processing Applications and Technology*, ICSPAT'95, Vol. 2, pp. 1892-1896, 1995.

[Gras97b] S. Grassi, *DSP56001 Implementation of the Spectral Analysis and Quantization for the CELP FS1016 Speech Coder*, IMT Report No 421 PE 10/97, University of Neuchâtel, IMT, Oct. 1997.

[MOTO90] *DSP56000/DSP56001 Digital Signal Processor User's Manual*, DSP56000UM/AD Rev.2, Motorola Inc., 1990.

[MOTO93] A. Chrysafis and S. Lansdowne, “Fractional and Integer Arithmetic Using the DSP56000 Family of General-purpose Digital Signal Processors”, APR3/D Rev. 1, Motorola Inc., 1993.

[Proa89] J. Proakis and D. Manolakis, *Introduction to Digital Signal Processing*, Macmillan, New York, 1989.

[Rupp96] B. Rupp, *Implantation Temps Réel d'un Codec CELP selon la norme "U.S. Fédéral Standard 1016" sur un Processeur de Traitement de Signal DSP56001*, (in French), practical semester project, winter semester 1995/96, Ecole polytechnique fédérale de Lausanne, Laboratoire de microtechnique EPFL - UNI NE, Neuchâtel, 1996.

[Saou95] A. Goalic and S. Saoudi, "An Intrinsically Reliable and Fast Algorithm to Compute the Line Spectrum Pairs (LSP) in Low Bit Rate CELP Coding", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP'95, Vol. 1, pp. 728-731, 1995.

[Segu97] F. Seguin, *Implantation en Temps Réel d'un Codec CELP sur un Processeur de Traitement de Signal DSP56001*, (in French), diploma work, winter semester 1996/97, Ecole polytechnique fédérale de Lausanne, Laboratoire de microtechnique EPFL - UNI NE, Neuchâtel, 1997.

Blank page

Chapter 8
Conclusions

The research presented in this Ph.D. report addressed the optimized implementation of some functional blocks which are found frequently in digital speech processing applications.

It was shown that algorithmic optimization and the choice of a fixed-point arithmetic are essential to meet the tight constraints in power consumption and size of applications such as digital hearing aids or portable communications devices.

A methodology for optimization of speech processing algorithms was proposed, as well as a practical and simple method for evaluating fixed-point quantization effects on these algorithms. Although the application is restricted to digital speech processing algorithms, the method presented is general enough to be easily extended to other classes of DSP algorithms.

The developed method allows the simulation of a system in final working conditions and at the same time benefit of the flexibility of using a high level language, independently of the hardware. In this way, different implementation possibilities can be easily tried out, before doing the actual implementation.

The proposed optimization methodology was used in the implementation of a noise reduction/speech enhancement algorithm for digital hearing aids on a fixed-point commercial DSP and using a low power VLSI architecture.

Two novel efficient algorithms for LSP calculation from LPC coefficients, named Mixed-LSP and "quantized-search Kabal" were presented. These proposed LSP calculation algorithms were compared with existing algorithms from the point of view of accuracy, reliability and computational complexity.

Kabal's algorithm was found to be the most efficient and accurate of the existing methods. This algorithm, as well as Mixed-LSP and “quantized-search Kabal”, were implemented on a DSP56001 and their computational complexity in MIPS was compared. It was found that “quantized-search Kabal" algorithm was more efficient than Kabal's algorithm, for the implementation in the CELP FS1016 speech coder.

These results were used in the efficient DSP56001 implementation of the CELP FS1016 spectral analysis and quantization.

To summarize, the key points for optimized low power implementations are careful algorithmic optimization, study of the fixed-point quantization effects, and careful match between algorithms and target architecture.

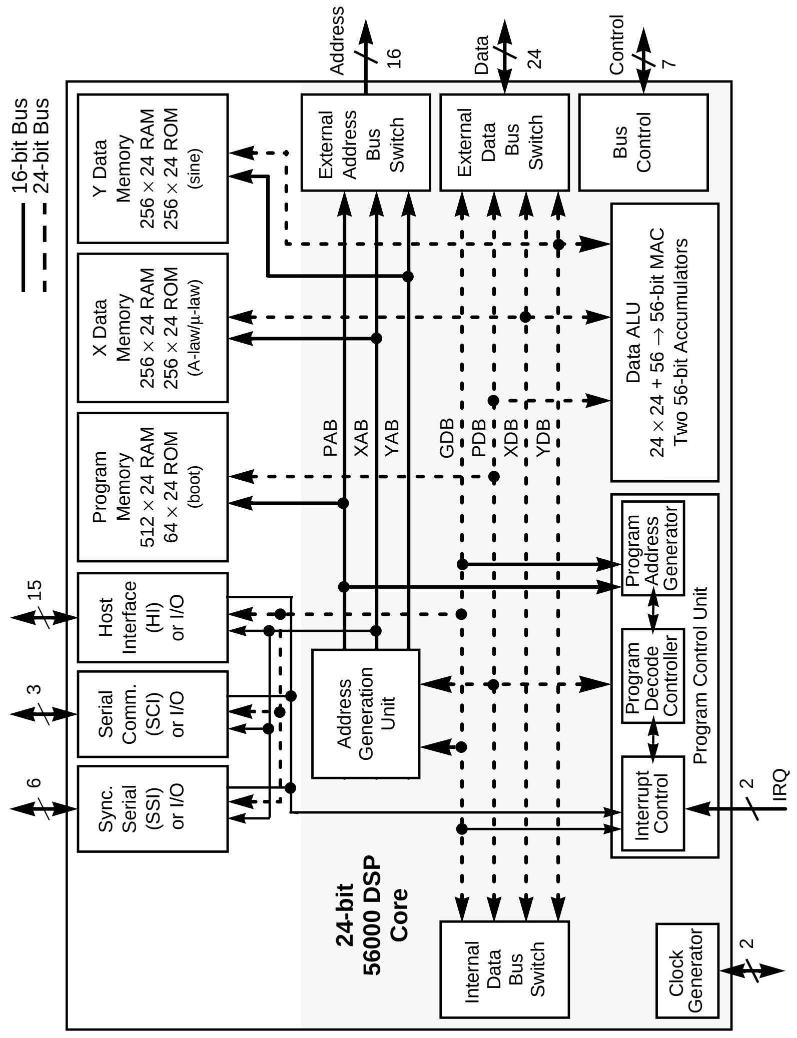
Possible extensions of the work done were given at the end of Chapter 4, 6 and 7.

Appendix A
Fixed-point Quantization Effects

A.1. Macros and Functions to Simulate Different Types of Truncation or Rounding

```
/* rounding */
#define RND1(a)   (((a)<0) ? ceil((a-0.5):floor((a)+0.5))

/* 2sc truncation */
#define TCTRNK(a) (floor(a))

/* Sign magnitude truncation */
#define SMTRNK(a) (((a) < 0) ? ceil(a) : floor(a))

/* convergent rounding */
#define RND(a)      crnd(a, prec)

/* convergent rounding */
double crnd(double a, int prec)
{
     double a1; long a0;

     a0= 2.0*conv[prec]*fabs(modf(a,&a1));

     if (a < 0) a= ceil(a-0.5) ;
     else a= floor(a+0.5);

     /* Correction for convergent rounding */
     if ((a0==conv[prec])&&(fmod(a1,2.0)==0))
          (a < 0) ? (a++) : (a--);

     return a;
}
```

A.2. Block Diagram of the DSP56001


16-bit Bus
24-bit Bus
Y Data Memory 256 × 24 RAM 256 × 24 ROM (sine)
X Data Memory 256 × 24 RAM 256 × 24 ROM (A-law/μ-law)
Program Memory 512 × 24 RAM 64 × 24 ROM (boot)
Host Interface (HI) or I/O
Serial Comm. (SCI) or I/O
Sync. Serial (SSI) or I/O
15
3
6
Address
16
Data
24
Control
7
External Address Bus Switch
External Data Bus Switch
Bus Control
Data ALU 24 × 24 + 56 → 56-bit MAC Two 56-bit Accumulators
PAB
XAB
YAB
GDB
PDB
XDB
YDB
Program Address Generator
Program Decode Controller
Interrupt Control
Program Control Unit
Address Generation Unit
2
IRQ
24-bit 56000 DSP Core
Internal Data Bus Switch
Clock Generator
2

A.3. Arithmetic Instructions of the DSP56001

Instruction	Description
ABS D (parallel move)	Store the absolute value of the destination operand D in the destination accumulator
ADD S,D (parallel move)	Add the source operand S to the destination operand D and store the result in the destination accumulator
ADDR S,D (parallel move)	Add the source operand S to one-half the destination operand D and store the result in the destination accumulator
ASR D (parallel move)	Arithmetically shift the destination operand D one bit to the right and store the result in the destination accumulator
CMP S1, S2 (parallel move)	Subtract the source one operand, S1, from the source two accumulator, S2, and update the condition code register
DIV S,D	Divide the destination operand D (48-bit positive fraction dividend sign extended to 56-bit) by the source operand S (24-bit signed fraction divisor) and store the partial remainder and the formed quotient (one new bit) in the destination accumulator D
MACR (+,-) S1,S2,D (parallel move)	Multiply the two signed 24-bit source operands S1 and S2 and add/subtract the product to/from the specified 56-bit destination accumulator D, and then round the result using convergent rounding
MPYR (+,-) S1,S2,D (parallel move)	Multiply the two signed 24-bit source operands S1 and S2 round the result using convergent rounding, and store the resulting product (with optional negation) in the specified 56-bit destination accumulator
NORM Ra,D	Based upon the result of one 56-bit normalization iteration on the specified destination operand D, update the specified address register Ra and store the result back in the destination accumulator
SBC S,D (parallel move)	Subtract the source operand S and the carry bit C of the condition code register from the destination operand D and store the result in the destination accumulator
SUBL S,D (parallel move)	Subtract the source operand S from two times the destination operand D and store the result in the destination accumulator
Tcc S1,D1	Transfer data from the specified source register S1 to the specified destination accumulator D1 if the specified condition “cc” is true

TST S (parallel move)	Compare the specified source accumulator S with zero and set the condition code accordingly
ADC S,D (parallel move)	Add the source operand S and the carry bit C of the condition code register to the destination operand D and store the result in the destination accumulator
ADDL S,D (parallel move)	Add the source operand S to two times the destination operand D and store the result in the destination accumulator
ASL D (parallel move)	Arithmetically shift the destination operand D one bit to the left and store the result in the destination accumulator
CLR D (parallel move)	Clear the 56-bit destination accumulator
CMPM S1,S2 (parallel move)	Subtract the magnitude of the source one operand, S1, from the magnitude of the source two accumulator, S2, an update the condition code register
MAC (+,-) S1,S2,D (parallel move)	Multiply the two signed 24-bit source operands S1 and S2 and add/subtract the product to/from the specified 56-bit destination accumulator D
MPY (+,-) S1,S2,D (parallel move)	Multiply the two signed 24-bit source operands S1 and S2 and store the resulting product (with optional negation) in the specified 56-bit destination accumulator
NEG D (parallel move)	Negate (56-bit twos-complement) the destination operand D and store the result in the destination accumulator
RND D (parallel move)	Round the 56-bit value in the specified destination operand D by convergent rounding and store the result in the most significant portion of the destination accumulator (A1 to B1)
SUB S,D (parallel move)	Subtract the source operand S from the destination operand D and store the result in the destination operand D
SUBR S,D (parallel move)	Subtract the source operand S from the one-half the destination operand D and store the result in the destination accumulator
TFR S,D (parallel move)	Transfer data from the specified source data ALU register S to the specified destination data ALU accumulator D

Appendix B LeRoux-Gueguen Algorithm

The Levinson-Durbin recursion given in Equation (5.12) is an efficient way to determine the LPC coefficients (see § 5.2), but if the goal is to compute the reflection coefficients, $\{k_m\}$, the LPC coefficients are also computed, as intermediate quantities. As the LPC coefficients are not bounded and have large dynamic range, the implementation of the Levinson-Durbin algorithm in a fixed-point device is difficult. LeRoux and Gueguen [Lero77] solved the problem by introducing the quantities:

$$e_m(i) = r_i + a_m(1)\cdot r_{i-1} + \ldots + a_m(m)\cdot r_{i-m} \qquad \text{(B.1)}$$

where $\{a_m(i)\}$ are the *m*-th order LPC coefficients and r_k is the *k*-th autocorrelation coefficient of the windowed speech signal:

$$r_k = \sum_{n=k}^{N-1} w(n)\cdot s(n)\cdot w(n-k)\cdot s(n-k)$$
$$r_k = r_{-k} \qquad \text{(B.2)}$$

B.1. LeRoux-Gueguen Algorithm

The reflection coefficients, $\{k_m\}$, are computed using the relations:

$$\begin{aligned}
&e_0(i) = r_i \qquad \text{for} \quad i = -p+1,\ldots,p \\
&\text{for} \quad m = 1,\ldots,p: \\
&k_m = \frac{-e_{m-1}(m)}{e_{m-1}(0)} \\
&e_m(i) = e_{m-1}(i) + k_m e_{m-1}(m-i) \qquad \text{for} \quad i = -p+1+m,\ldots,p
\end{aligned} \tag{B.3}$$

the values $e_m(i)$ for $i = 1,\ldots,m$, turn out to be zero, thus they need not be computed.

One important result of the formulation is that if the autocorrelation sequence is normalized, i.e., $|r_k| \le 1$, all the quantities $e_m(i)$ lie between –1 and +1. Consequently the computation can be easily implemented using fixed-point arithmetic. The normalization of the autocorrelation sequence needs 10 divisions.

The total computational cost is 90 multiplications, 90 additions and 20 divisions.

B.2. References

[Lero77] J. LeRoux and C. Gueguen, "A Fixed Point Computation of Partial Correlation Coefficients", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 25, No. 3, pp. 257-259, 1977.

[Papa87] P. Papamichalis, *Practical Approaches to Speech Coding* (Chapter 5), Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

Appendix C
LSP to LPC Transformation

C.1. Direct Expansion Method

The symmetrical and antisymmetrical polynomials, $P_{10}(z)$ and $Q_{10}(z)$, are given by:

$$\begin{aligned}
&P_{10}(z) = (1+z^{-1}) \prod_{i=1,3,5,7,9} \left(1 + c_i \cdot z^{-1} + z^{-2}\right) = \sum_{i=0}^{11} p_i z^{-i} \\
&Q_{10}(z) = (1-z^{-1}) \prod_{i=2,4,6,8,10} \left(1 + c_i \cdot z^{-1} + z^{-2}\right) = \sum_{i=0}^{11} q_i z^{-i} \\
&\text{with} \quad c_i = -2\cos\left(\omega_i\right)
\end{aligned} \tag{C.1}$$

where the $\{\omega_i\}$ are the LSPs. The coefficients $\{p_i\}$ and $\{q_i\}$ are found by multiplying the product terms of Equation (C.1):

$$\begin{aligned}
&p_0 = p_{11} = 1 \\
&p_1 = p_{10} = 1 + s_1 \\
&p_2 = p_9 = 5 + s_1 + s_2 \\
&p_3 = p_8 = 5 + 4s_1 + s_2 + s_3 \\
&p_4 = p_7 = 10 + 4s_1 + 3s_2 + s_3 + s_4 \\
&p_5 = p_6 = 10 + 6s_1 + 3s_2 + 2s_3 + s_4 + s_5
\end{aligned}$$

$$
\begin{aligned}
q_0 &= -q_{11} = 1\\
q_1 &= -q_{10} = -1 + s'_1\\
q_2 &= -q_9 = 5 - s'_1 + s'_2\\
q_3 &= -q_8 = -5 + 4s'_1 - s'_2 + s'_3\\
q_4 &= -q_7 = 10 - 4s'_1 + 3s'_2 - s'_3 + s'_4\\
q_5 &= -q_6 = -10 + 6s'_1 - 3s'_2 + 2s'_3 - s'_4 + s'_5
\end{aligned} \tag{C.2}
$$

where $\{s_i\}$ and $\{s'_i\}$ are the summation of product terms of the odd- and even-suffixed c_i respectively:

$$
\begin{aligned}
s_1 &= c_1 + c_3 + c_5 + c_7 + c_9\\
s_2 &= c_1c_3 + c_1c_5 + c_1c_7 + c_1c_9 + c_3c_5 + c_3c_7 + c_3c_9 + c_5c_7 +\\
&\quad + c_5c_9 + c_7c_9\\
s_3 &= c_1c_3c_5 + c_1c_3c_7 + c_1c_3c_9 + c_1c_5c_7 + c_1c_5c_9 + c_1c_7c_9 +\\
&\quad + c_3c_5c_7 + c_3c_5c_9 + c_3c_7c_9 + c_5c_7c_9\\
s_4 &= c_1c_3c_5c_7 + c_1c_3c_5c_9 + c_1c_3c_7c_9 + c_1c_5c_7c_9 + c_3c_5c_7c_9\\
s_5 &= c_1c_3c_5c_7c_9\\
s'_1 &= c_2 + c_4 + c_6 + c_8 + c_{10}\\
s'_2 &= c_2c_4 + c_2c_6 + c_2c_8 + c_2c_{10} + c_4c_6 + c_4c_8 + c_4c_{10} + c_6c_8 +\\
&\quad + c_6c_{10} + c_8c_{10}\\
s'_3 &= c_2c_4c_6 + c_2c_4c_8 + c_2c_4c_{10} + c_2c_6c_8 + c_2c_6c_{10} + c_2c_8c_{10} +\\
&\quad + c_4c_6c_8 + c_4c_6c_{10} + c_4c_8c_{10} + c_6c_8c_{10}\\
s'_4 &= c_2c_4c_6c_8 + c_2c_4c_6c_{10} + c_2c_4c_8c_{10} + c_2c_6c_8c_{10} + c_4c_6c_8c_{10}\\
s'_5 &= c_2c_4c_6c_8c_{10}
\end{aligned} \tag{C.3}
$$

finally the LPC filter is given by:

$$A_{10}(z) = \frac{P_{10}(z) + Q_{10}(z)}{2} \tag{C.4}$$

The total computational cost is 62 multiplications and 92 additions. The shift operations (by a factor of two) were not counted.

C.2. LPC Analysis Filter Method

When the filter of Figure C.1 is excited with an 11-term impulse sequence, the resulting output sequence $\{1,a_1,\ldots,a_{10}\}$ gives the LPC coefficients. Each second-order section requires 1 multiplication and 2 additions and the first-order sections require 1 addition. Thus, the total cost would be 110 multiplications and 253 additions, but several savings are possible. The output sequences of each section, denoted as h_i and h'_i, for the upper and lower branch respectively, are shown in Figure C.1 and given by:

$$
\begin{aligned}
h_0 &= \{1,1,0,0,0,0,0,0,0,0,0,0\}\\
h_1 &= \{1, p_{11}, p_{11}, 1,0,0,0,0,0,0,0,0\}\\
h_2 &= \{1, p_{21}, p_{22}, p_{22}, p_{21}, 1,0,0,0,0,0,0\}\\
h_3 &= \{1, p_{31}, p_{32}, p_{33}, p_{33}, p_{32}, p_{31}, 1,0,0,0,0\}\\
h_4 &= \{1, p_{41}, p_{42}, p_{43}, p_{44}, p_{44}, p_{43}, p_{42}, p_{41}, 1,0,0\}\\
h_5 &= \{1, p_{51}, p_{52}, p_{53}, p_{54}, p_{55}, p_{55}, p_{54}, p_{53}, p_{52}, p_{51}, 1\}\\
h'_0 &= \{1,-1,0,0,0,0,0,0,0,0,0,0\}\\
h'_1 &= \{1, q_{11}, -q_{11}, -1,0,0,0,0,0,0,0,0\}\\
h'_2 &= \{1, q_{21}, q_{22}, -q_{22}, -q_{21}, -1,0,0,0,0,0,0\}\\
h'_3 &= \{1, q_{31}, q_{32}, q_{33}, -q_{33}, -q_{32}, -q_{31}, -1,0,0,0,0\}\\
h'_4 &= \{1, q_{41}, q_{42}, q_{43}, q_{44}, -q_{44}, -q_{43}, -q_{42}, -q_{41}, -1,0,0\}\\
h'_5 &= \{1, q_{51}, q_{52}, q_{53}, q_{54}, q_{55}, -q_{55}, -q_{54}, -q_{53}, -q_{52}, -q_{51}, -1\}
\end{aligned} \tag{C.5}
$$

Thus, only 30 different terms are calculated, as outputs of a second order section. Ten extra additions are needed to obtain the LPC coefficients from these terms. The total cost is reduced to 30 multiplications and 70 additions, at the cost of increased effort in the flow control of the algorithm.

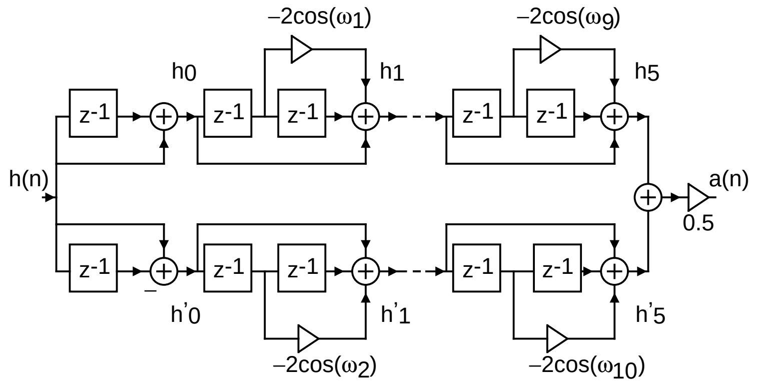


Figure C.1: Filter used to generate the LPC coefficients, in the LPC analysis filter method. The $\{\omega_i\}$ are the LSP parameters.

C.3. Kabal's Method

The polynomials $P_{10}(z)$ and $Q_{10}(z)$ are expressed as:

$$P_{10}(z) = (1+z^{-1}) \cdot \sum_{i=0}^{10} p'_i z^{-i}$$

$$Q_{10}(z) = (1-z^{-1}) \cdot \sum_{i=0}^{10} q'_i z^{-i} \qquad \text{(C.6)}$$

As $P'_{10}(z)$ and $Q'_{10}(z)$ are symmetrical, only their first five coefficients need to be calculated. The coefficients $\{q'_i\}$ and $\{p'_i\}$ are obtained using the following recursions:

$$\begin{aligned}
c_{10} &= -x_1 & c'_{10} &= -x_2 \\
c_{20} &= -2\cdot x_3\cdot c_{10}+1 & c'_{20} &= -2\cdot x_4\cdot c'_{10}+1 \\
c_{21} &= 2\cdot c_{10}-2\cdot x_3 & c'_{21} &= 2\cdot c'_{10}-2\cdot x_4 \\
c_{30} &= -2\cdot x_5\cdot c_{20}+c_{21} & c'_{30} &= -2\cdot x_6\cdot c'_{20}+c'_{21} \\
c_{31} &= 2\cdot c_{20}-2\cdot x_5\cdot c_{21}+1 & c'_{31} &= 2\cdot c'_{20}-2\cdot x_6\cdot c'_{21}+1 \\
c_{32} &= c_{21}-2\cdot x_5 & c'_{32} &= c'_{21}-2\cdot x_6 \\
c_{40} &= -2\cdot x_7\cdot c_{30}+c_{31} & c'_{40} &= -2\cdot x_8\cdot c'_{30}+c'_{31} \\
c_{41} &= 2\cdot c_{30}-2\cdot x_7\cdot c_{31}+c_{32} & c'_{41} &= 2\cdot c'_{30}-2\cdot x_8\cdot c'_{31}+c'_{32} \\
c_{42} &= c_{31}-2\cdot x_7\cdot c_{32}+1 & c'_{42} &= c'_{31}-2\cdot x_8\cdot c'_{32}+1 \\
c_{43} &= c_{32}-2\cdot x_7 & c'_{43} &= c'_{32}-2\cdot x_8 \\
c_{50} &= -2\cdot x_9\cdot c_{40}+c_{41} & c'_{50} &= -2\cdot x_{10}\cdot c'_{40}+c'_{41} \\
c_{51} &= 2\cdot c_{40}-2\cdot x_9\cdot c_{41}+c_{42} & c'_{51} &= 2\cdot c'_{40}-2\cdot x_{10}\cdot c'_{41}+c'_{42} \\
c_{52} &= c_{41}-2\cdot x_9\cdot c_{42}+c_{43} & c'_{52} &= c'_{41}-2\cdot x_{10}\cdot c'_{42}+c'_{43} \\
c_{53} &= c_{42}-2\cdot x_9\cdot c_{43}+1 & c'_{53} &= c'_{42}-2\cdot x_{10}\cdot c'_{43}+1 \\
c_{54} &= c_{43}-2\cdot x_9 & c'_{54} &= c'_{43}-2\cdot x_{10}
\end{aligned} \qquad \text{(C.7)}$$

where $\{x_i\}$ are the LSPs in the "x domain", with $x_i = \cos(\omega_i)$. The last terms of this recursion give the coefficients $\{q'_i\}$ and $\{p'_i\}$:

$$\begin{aligned}
p'_5 &= 2\cdot c_{50} & q'_5 &= 2\cdot c'_{50} \\
p'_4 &= c_{51} & q'_4 &= c'_{51} \\
p'_3 &= c_{52} & q'_3 &= c'_{52} \\
p'_2 &= c_{53} & q'_2 &= c'_{53} \\
p'_1 &= c_{54} & q'_1 &= c'_{54}
\end{aligned} \qquad \text{(C.8)}$$

Following Equation (C.6), the LPC coefficients are given by:

$$\begin{aligned}
a_1 &= \frac{p'_1 + q'_1}{2} & a_{10} &= \frac{p'_1 - q'_1 + 2}{2} \\
a_2 &= \frac{(p'_2 + p'_1) + (q'_2 - q'_1)}{2} & a_9 &= \frac{(p'_2 + p'_1) - (q'_2 - q'_1)}{2} \\
a_3 &= \frac{(p'_3 + p'_2) + (q'_3 - q'_2)}{2} & a_8 &= \frac{(p'_3 + p'_2) - (q'_3 - q'_2)}{2} \\
a_4 &= \frac{(p'_4 + p'_3) + (q'_4 - q'_3)}{2} & a_7 &= \frac{(p'_4 + p'_3) - (q'_4 - q'_3)}{2} \\
a_5 &= \frac{(p'_5 + p'_4) + (q'_5 - q'_4)}{2} & a_6 &= \frac{(p'_5 + p'_4) - (q'_5 - q'_4)}{2}
\end{aligned} \tag{C.9}$$

The total computational cost is 20 multiplications and 59 additions. This is the least expensive of the three algorithms for LSP to LPC conversion given in this appendix. Besides, the algorithm is highly regular, which is a advantageous for efficient implementation.

Appendix D Mixed-LSP Method

D.1. Derivation of the Polynomials $P'_{10}(x)$ and $Q'_{10}(x)$

The derivation given in this section is done for an LPC order p = 10, but it can be extended to any even p. Starting from the auxiliary function $\psi_m(z)$, given in Equation (5.39):

$$\psi_{10}(z) = z^{\frac{11}{2}} \cdot A_{10}(z) \tag{D.1}$$

where $A_{10}(z)$ is the 10-th order LPC analysis filter, given by:

$$A_{10}(z) = 1 + \sum_{k=1}^{10} a_k \cdot z^{-k} \tag{D.2}$$

the function $\psi_m(z)$ is evaluated on the unit circle, $z = e^{j\omega}$:

$$\psi_{10}(e^{j\omega}) = e^{\frac{j \cdot 11 \cdot \omega}{2}} A_{10}(e^{j\omega}) = \psi_{10}^{(r)}(e^{j\omega}) + j \cdot \psi_{10}^{(i)}(e^{j\omega}) \tag{D.3}$$

The symmetrical and antisymmetrical polynomials, $P_{10}(z)$ and $Q_{10}(z)$, given in Equation (5.24) are also evaluated on the unit circle:

$$P_{10}(e^{j\omega}) = A_{10}(e^{j\omega}) + e^{-j\cdot 11\cdot\omega} A_{10}(e^{-j\omega}) = 2\,e^{-j\frac{11}{2}\omega}\,\psi^{(r)}_{10}(e^{j\omega})$$
$$Q_{10}(e^{j\omega}) = A_{10}(e^{j\omega}) - e^{-j\cdot 11\cdot\omega} A_{10}(e^{-j\omega}) = 2\,j\,e^{-j\frac{11}{2}\omega}\,\psi^{(i)}_{10}(e^{j\omega}) \qquad (D.4)$$

Thus, the zero crossings of $\psi^{(r)}_{10}(e^{j\omega})$ and $\psi^{(i)}_{10}(e^{j\omega})$ correspond to the odd- and even-suffixed LSPs, respectively. Equation (D.3) can be arranged as:

$$\begin{aligned}\psi_{10}(e^{j\omega}) &= e^{\frac{j\cdot 11\cdot\omega}{2}}(1 + a_1\cdot e^{-j\omega} + \ldots + a_{10}\cdot e^{-j\cdot 10\cdot\omega}) \\ &= e^{j\frac{\omega}{2}}(e^{j5\omega} + a_1\cdot e^{j4\omega} + \ldots + a_9\cdot e^{-j4\omega} + a_{10}\cdot e^{-j5\omega}) \\ &= \left[\cos\left(\frac{\omega}{2}\right) + j\cdot\sin\left(\frac{\omega}{2}\right)\right]\cdot\left[R_{10}(e^{j\omega}) + j\cdot I_{10}(e^{j\omega})\right] \end{aligned} \qquad (D.5)$$

thus, the real and imaginary parts of $\psi_{10}(e^{j\omega})$ are:

$$\begin{aligned}\psi^{(r)}_{10}(e^{j\omega}) &= \cos\left(\frac{\omega}{2}\right)\cdot R_{10}(e^{j\omega}) - \sin\left(\frac{\omega}{2}\right)\cdot I_{10}(e^{j\omega}) \\ \psi^{(i)}_{10}(e^{j\omega}) &= \sin\left(\frac{\omega}{2}\right)\cdot R_{10}(e^{j\omega}) + \cos\left(\frac{\omega}{2}\right)\cdot I_{10}(e^{j\omega})\end{aligned} \qquad (D.6)$$

where $R_{10}(e^{j\omega})$ and $I_{10}(e^{j\omega})$ are given by:

$$\begin{aligned}R_{10}\left(e^{j\omega}\right) &= \left(1 + a_{10}\right)\cdot\cos(5\omega) + \ldots + \left(a_4 + a_6\right)\cdot\cos(\omega) + a_5 \\ &= A_5\cos(5\omega) + A_4\cos(4\omega) + \ldots + A_1\cos(\omega) + A_0 \\ I_{10}\left(e^{j\omega}\right) &= \left(1 - a_{10}\right)\cdot\sin(5\omega) + \ldots + \left(a_4 - a_6\right)\cdot\sin(\omega) \\ &= E_5\sin(5\omega) + E_4\sin(4\omega) + \ldots + E_1\sin(\omega)\end{aligned} \qquad (D.7)$$

Using the mapping $x = \cos(\omega)$, the Chebyshev polynomials of first kind are given by:

$$\begin{aligned}T_n(x) &= \cos(n\omega) = 2xT_{n-1}(x) - T_{n-2}(x) \\ T_0(x) &= \cos(0) = 1 \\ T_1(x) &= \cos(\omega) = x \\ T_2(x) &= \cos(2\omega) = 2x^2 - 1 \\ T_3(x) &= \cos(3\omega) = 4x^3 - 3x \\ T_4(x) &= \cos(4\omega) = 8x^4 - 8x^2 + 1 \\ T_5(x) &= \cos(5\omega) = 16x^5 - 20x^3 + 5x\end{aligned} \qquad (D.8)$$

while the Chebyshev polynomials of second kind are given by:

$$\begin{aligned}U_n(x) &= \sin(n\omega) = 2xU_{n-1}(x) - U_{n-2}(x) \\ U_0(x) &= 0 \\ U_1(x) &= \sin(\omega) = \pm\sqrt{1 - x^2} \\ U_2(x) &= \sin(2\omega) = \pm\sqrt{1 - x^2}\cdot(2x) \\ U_3(x) &= \sin(3\omega) = \pm\sqrt{1 - x^2}\cdot(4x^2 - 1) \\ U_4(x) &= \sin(4\omega) = \pm\sqrt{1 - x^2}\cdot(8x^3 - 4x) \\ U_5(x) &= \sin(5\omega) = \pm\sqrt{1 - x^2}\cdot(16x^4 - 12x^2 + 1)\end{aligned} \qquad (D.9)$$

In the upper semicircle of the z-plane, $\omega\in[0,\pi]$, in which the LSPs are located, the trigonometric formulas for double-angle give:

$$\begin{aligned}\cos(\omega) &= 2\cos^2\left(\frac{\omega}{2}\right) - 1 \quad\Rightarrow\quad \cos\left(\frac{\omega}{2}\right) = \sqrt{\frac{1 + x}{2}} \\ \cos(\omega) &= 1 - 2\sin^2\left(\frac{\omega}{2}\right) \quad\Rightarrow\quad \sin\left(\frac{\omega}{2}\right) = \sqrt{\frac{1 - x}{2}}\end{aligned} \qquad (D.10)$$

Then, using the mapping $x = \cos(\omega)$ and Chebyshev polynomials of first and second kind Equation (D.7) is expressed as:

$$\begin{aligned}R_{10}(x) &= 16A_5x^5 + 8A_4x^4 + (-20A_5 + 4A_3)x^3 + (-8A_4 + 2A_2)x^2 + \\ &+(5A_5 - 3A_3 + A_1)x + (A_4 - A_2 + A_0) \\ &= D_5x^5 + D_4x^4 + D_3x^3 + D_2x^2 + D_1x + D_0 \\ I_{10}(x) &= \pm\sqrt{1-x^2}\left[16E_5x^4 + 8E_4x^3 + (4E_3 - 12E_5)x^2 + \right. \\ &\left. +(2E_2 - 4E_4)x + (E_5 - E_3 + E_1)\right] \\ &= \pm\sqrt{1-x^2}\left[B_4x^4 + B_3x^3 + B_2x^2 + B_1x + B_0\right] \\ &= \pm\sqrt{1-x^2}\,I'_{10}(x)\end{aligned} \tag{D.11}$$

and using the mapping $x = \cos(\omega)$ and the trigonometric formulas for double angle, Equation (D.6) can be expressed as:

$$\begin{aligned}\psi_{10}^{(r)}(x) &= \sqrt{\frac{1+x}{2}}\cdot R_{10}(x) - \sqrt{\frac{1-x}{2}}\cdot\sqrt{1-x^2}\cdot I'_{10}(x) \\ &= \sqrt{\frac{1+x}{2}}\cdot\left[R_{10}(x) - (1-x)\cdot I'_{10}(x)\right] = \sqrt{\frac{1+x}{2}}\cdot\psi'^{(r)}_{10}(x) \\ \psi_{10}^{(i)}(x) &= \sqrt{\frac{1-x}{2}}\cdot R_{10}(x) + \sqrt{\frac{1+x}{2}}\cdot\sqrt{1-x^2}\cdot I'_{10}(x) \\ &= \sqrt{\frac{1-x}{2}}\cdot\left[R_{10}(x) + (1+x)\cdot I'_{10}(x)\right] = \sqrt{\frac{1-x}{2}}\cdot\psi'^{(i)}_{10}(x)\end{aligned} \tag{D.12}$$

In the upper semicircle of the z-plane, $\omega\in[0,\pi]$, in which the LSPs are located, the terms:

$$\sqrt{\frac{1+x}{2}} = \cos\left(\frac{\omega}{2}\right) \quad \text{and} \quad \sqrt{\frac{1-x}{2}} = \sin\left(\frac{\omega}{2}\right) \tag{D.13}$$

are different from zero, except at $x = -1$ and $x = +1$, respectively. Thus, these terms can be removed without affecting the position of the other zeros (LSPs). Then the functions:

$$\begin{aligned}\psi'^{(r)}_{10}(x) &= \left[R_{10}(x) - (1-x)\cdot I'_{10}(x)\right] \\ \psi'^{(i)}_{10}(x) &= \left[R_{10}(x) + (1+x)\cdot I'_{10}(x)\right]\end{aligned} \tag{D.14}$$

have all the zero crossings (LSPs) of Kabal's polynomials $P'_{10}(x)$ and $Q'_{10}(x)$. The leading coefficient, which is the coefficient that multiplies the higher power of x (x^5) is, for both functions:

$$\gamma = 16\cdot(A_5 + E_5) = 16\cdot\left[(1+a_{10}) + (1-a_{10})\right] = 32 \tag{D.15}$$

while in Equation (5.30) it is seen that the leading coefficient of Kabal polynomials, $P'_{10}(x)$ and $Q'_{10}(x)$, is 16. Thus, Kabal's polynomials can be expressed as:

$$\begin{aligned}P'_{10}(x) &= \frac{\psi'^{(r)}_{10}(x)}{2} = C_{10}(x) - D_{10}(x) \\ Q'_{10}(x) &= \frac{\psi'^{(i)}_{10}(x)}{2} = C_{10}(x) + D_{10}(x)\end{aligned} \tag{D.16}$$

where $C_{10}(x)$ is a 5-th order polynomial and $D_{10}(x)$ is the 4-th order polynomial, given by:

$$\begin{aligned}2\cdot D_{10}(x) = I'_{10}(x) &= 16E_5x^4 + 8E_4x^3 + (4E_3 - 12E_5)x^2 + \\ &+(2E_2 - 4E_4)x + (E_5 - E_3 + E_1)\end{aligned} \tag{D.17}$$

where:

$$\begin{aligned}&E_5 = 1 - a_{10}, \quad E_4 = a_1 - a_9, \quad E_3 = a_2 - a_8, \\ &E_2 = a_3 - a_7, \quad E_1 = a_4 - a_6\end{aligned} \tag{D.18}$$

From Equation (D.16):

$$D_{10}(x) = \frac{Q'_{10}(x) - P'_{10}(x)}{2} \tag{D.19}$$

Thus the polynomial $D_{10}(x)$ could have also been obtained from Kabal's derivation, using Equation (5.30) and realizing that $P'_{10}(x)$ and $Q'_{10}(x)$ have the same leading term. Nevertheless, the derivation given in this section leads naturally to Equation (D.16) which is the base of the Mixed-LSP algorithm.

For the purpose of the Mixed-LSP algorithm, only the position of the roots of $D_{10}(x)$ is of interest. Thus the factor of 2, multiplying $D_{10}(x)$ in Equation (D.17) will be ignored.

D.2. Properties of the Roots of $D_{10}(x)$

As $D_{10}(x)$ is a fourth order polynomial, it has four roots, in which $D_{10}(x)=0$. In section D.6, it is proved that the roots of $D_{10}(x)$ are real, different, and inside the interval (-1,1). Furthermore, the roots of $D_{10}(x)$ correspond to the values of x in which functions $P'_{10}(x)$ and $Q'_{10}(x)$, cross each other, as it can be seen in Equation (D.16), setting $D_{10}(x)=0$.

D.3. Direction of the Sign Changes

Kabal's polynomials, $P'_{10}(x)$ and $Q'_{10}(x)$, given in Equation (5.30), can be expressed as:

$$\begin{aligned} P'_{10}(x) &= 16(x-x_1)(x-x_3)(x-x_5)(x-x_7)(x-x_9) \\ Q'_{10}(x) &= 16(x-x_2)(x-x_4)(x-x_6)(x-x_8)(x-x_{10}) \end{aligned} \tag{D.20}$$

where the $\{x_i\}$, which are the roots of $P'_{10}(x)$ and $Q'_{10}(x)$, correspond to the LSPs in the "x domain". On the other hand, $D_{10}(x)$, can be expressed as:

$$\begin{aligned} D_{10}(x) &= 8\cdot E_5(x-r_1)(x-r_2)(x-r_3)(x-r_4) \\ &= 8\cdot(1-a_{10})(x-r_1)(x-r_2)(x-r_3)(x-r_4) \end{aligned} \tag{D.21}$$

where $\{r_i\}$ are the roots of $D_{10}(x)$. In Equation (5.18), it is seen that the LPC coefficient a_{10}, is equal to the reflection coefficient k_{10}. Thus, a_{10} is bounded in magnitude by 1, and $(1-a_{10})$ is always positive. As the leading terms of $P'_{10}(x)$, $Q'_{10}(x)$ and $D_{10}(x)$ are positive, and their roots are smaller than +1, then:

$$D_{10}(x=+1)>0,\quad P'_{10}(x=+1)>0,\quad Q'_{10}(x=+1)>0 \tag{D.22}$$

Therefore the directions of the sign changes at every zero crossing (+ to –, or – to +) are known. This property can be used for improving efficiency and reliability of the Mixed-LSP algorithm. From Equation (D.19) and (D.22), it is also seen that:

$$\begin{aligned} Q'_{10}(x=+1)-P'_{10}(x=+1) &= 2\cdot D_{10}(x=+1)>0 \\ \Rightarrow Q'_{10}(x=+1) &> P'_{10}(x=+1) \end{aligned} \tag{D.23}$$

D.4. Calculation of the Roots of $D_{10}(x)$

In this section the calculation of the roots of the 4-th order polynomial $D_{10}(x)$ is optimized. It is reminded that $D_{10}(x)$ has four roots which are real, different and in the interval (-1,1).

Resolution of a 4-th Order Polynomial

Given the 4-th order polynomial, with real coefficients [Ango72], [Barb89]:

$$x^4+ax^3+bx^2+cx+d=0 \tag{D.24}$$

factoring this polynomial as a multiplication of two second order polynomials with real coefficients:

$$x^4+ax^3+bx^2+cx+d=\left(x^2+p_1x+q_1\right)\left(x^2+p_2x+q_2\right)=0 \tag{D.25}$$

The following system of equations has to be solved for p_1, p_2, q_1 and q_2:

$$\begin{aligned} a &= p_1+p_2 \\ b &= q_1+q_2+p_1p_2 \\ c &= p_1q_2+p_2q_1 \\ d &= q_1q_2 \end{aligned} \tag{D.26}$$

Considering the trial solutions for the system (D.26), where the unknown variable z has been introduced:

$$\begin{aligned} p_1 &= \frac{a}{2}+\sqrt{\left(\frac{a}{2}\right)^2-b+z}\ , \qquad q_1=\frac{z}{2}+\varepsilon\cdot\sqrt{\left(\frac{z}{2}\right)^2-d} \\ p_2 &= \frac{a}{2}-\sqrt{\left(\frac{a}{2}\right)^2-b+z}\ , \qquad q_2=\frac{z}{2}-\varepsilon\cdot\sqrt{\left(\frac{z}{2}\right)^2-d} \end{aligned} \tag{D.27}$$

where $\varepsilon=\pm1$. Introducing these trial solutions in the system of equations given in (D.26), leads to Equations (D.28) and (D.29):

$$a = \left(\frac{a}{2} + \sqrt{\left(\frac{a}{2}\right)^2 - b + z}\right) + \left(\frac{a}{2} - \sqrt{\left(\frac{a}{2}\right)^2 - b + z}\right) = a$$

$$b = \left(\frac{z}{2} + \varepsilon\cdot\sqrt{\left(\frac{z}{2}\right)^2 - d}\right) + \left(\frac{z}{2} - \varepsilon\cdot\sqrt{\left(\frac{z}{2}\right)^2 - d}\right) +$$

$$+\left(\frac{a}{2} + \sqrt{\left(\frac{a}{2}\right)^2 - b + z}\right)\left(\frac{a}{2} - \sqrt{\left(\frac{a}{2}\right)^2 - b + z}\right) = b$$

$$d = \left(\frac{z}{2} + \varepsilon\cdot\sqrt{\left(\frac{z}{2}\right)^2 - d}\right)\left(\frac{z}{2} - \varepsilon\cdot\sqrt{\left(\frac{z}{2}\right)^2 - d}\right) = d \qquad \text{(D.28)}$$

It is seen that the equalities in (D.28) are always satisfied for any value of z. Thus z is chosen to satisfy the equation of c:

$$c = \left(\frac{a}{2} + \sqrt{\left(\frac{a}{2}\right)^2 - b + z}\right)\left(\frac{z}{2} - \varepsilon\cdot\sqrt{\left(\frac{z}{2}\right)^2 - d}\right)$$

$$+\left(\frac{a}{2} - \sqrt{\left(\frac{a}{2}\right)^2 - b + z}\right)\left(\frac{z}{2} + \varepsilon\cdot\sqrt{\left(\frac{z}{2}\right)^2 - d}\right) \qquad \text{(D.29)}$$

giving:

$$c = \frac{az}{2} - 2\varepsilon\cdot\sqrt{\left[\left(\frac{a}{2}\right)^2 - b + z\right]\left[\left(\frac{z}{2}\right)^2 - d\right]}$$

$$\text{where} \quad \varepsilon = \operatorname{sign}\left(\frac{az}{2} - c\right) \quad \text{for consistency} \qquad \text{(D.30)}$$

thus z must be a solution of the 3-rd order equation:

$$z^3 + rz^2 + sz + t = 0$$

$$r = -b, \quad s = ac - 4d, \quad t = d(4b - a^2) - c^2 \qquad \text{(D.31)}$$

then there are three possible solutions for z, namely z_a, z_b, and z_c, and at least one of these solutions is real. For p_1, p_2, q_1 and q_2 to be real, a real value of z must be chosen.

Thus, to solve the 4-th order polynomial given in Equation (D.24), the 3-rd order polynomial of Equation (D.31) must be solved first.

Property

If the roots of the original 4-th order polynomial of Equation (D.24) are real and different, then the roots of the 3-rd order polynomial of Equation (D.31) are real and different.

Proof

If the 4-th order polynomial of (D.24) has four different real roots, x_1, x_2, x_3 and x_4, then it can be expressed as:

$$x^4 + ax^3 + bx^2 + cx + d = (x - x_1)(x - x_2)(x - x_3)(x - x_4)$$

$$a = -x_1 - x_2 - x_3 - x_4$$

$$b = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$$

$$c = -x_1x_2x_3 - x_1x_2x_4 - x_1x_3x_4 - x_2x_3x_4$$

$$d = x_1x_2x_3x_4 \qquad \text{(D.32)}$$

there are three possible ways to factor the fourth order polynomial of (D.24) into a multiplication of two second order polynomials with real coefficients, as in Equation (D.25), each factorization corresponds to one root, z_a, z_b, or z_c, of the third order polynomial of Equation (D.31). These factorizations are given next:

Factorization 1:

$$x^4 + ax^3 + bx^2 + cx + d = \left(x^2 + p_{1a}x + q_{1a}\right)\left(x^2 + p_{2a}x + q_{2a}\right)$$

$$p_{1a} = -x_1 - x_2, \qquad p_{2a} = -x_3 - x_4$$

$$q_{1a} = x_1x_2, \qquad q_{2a} = x_3x_4$$

$$\text{Note: } (p_{1a}, q_{1a}) \text{ can be exchanged with } (p_{2a}, q_{2a}) \qquad \text{(D.33)}$$

and from Equation (D.27):

$$z_a = q_{1a} + q_{2a} = x_1x_2 + x_3x_4 \qquad \text{(D.34)}$$

Factorization 2:

$$x^4 + ax^3 + bx^2 + cx + d = \left(x^2 + p_{1b}x + q_{1b}\right)\left(x^2 + p_{2b}x + q_{2b}\right)$$
$$p_{1b} = -x_1 - x_3, \quad p_{2b} = -x_2 - x_4$$
$$q_{1b} = x_1x_3, \quad q_{2b} = x_2x_4$$
$$\text{Note: } \left(p_{1b}, q_{1b}\right) \text{ can be exchanged with } \left(p_{2b}, q_{2b}\right) \tag{D.35}$$

and from Equation (D.27):

$$z_b = q_{1b} + q_{2b} = x_1x_3 + x_2x_4 \tag{D.36}$$

Factorization 3:

$$x^4 + ax^3 + bx^2 + cx + d = \left(x^2 + p_{1c}x + q_{1c}\right)\left(x^2 + p_{2c}x + q_{2c}\right)$$
$$p_{1c} = -x_1 - x_4, \quad p_{2c} = -x_3 - x_2$$
$$q_{1c} = x_1x_4, \quad q_{2c} = x_3x_2$$
$$\text{Note: } \left(p_{1c}, q_{1c}\right) \text{ can be exchanged with } \left(p_{2c}, q_{2c}\right) \tag{D.37}$$

and from Equation (D.27):

$$z_c = q_{1c} + q_{2c} = x_1x_4 + x_3x_2 \tag{D.38}$$

From Equations (D.34), (D.36) and (D.38) it is seen that if the roots of the 4-th order polynomial of Equation (D.24) are real, then the roots of the 3-rd order polynomial of Equation (D.31) are real. Furthermore, if the roots of the 4-th order polynomial of Equation (D.24), are different, that is $x_1 \neq x_2 \neq x_3 \neq x_4$, then:

$$\begin{aligned} z_a - z_b &= \left(x_1x_2 + x_3x_4\right) - \left(x_1x_3 + x_2x_4\right) = \left(x_1 - x_4\right)\cdot\left(x_2 - x_3\right) \neq 0 \\ z_a - z_c &= \left(x_1x_2 + x_3x_4\right) - \left(x_1x_4 + x_3x_2\right) = \left(x_1 - x_3\right)\cdot\left(x_2 - x_4\right) \neq 0 \\ z_b - z_c &= \left(x_1x_3 + x_2x_4\right) - \left(x_1x_4 + x_3x_2\right) = \left(x_1 - x_2\right)\cdot\left(x_3 - x_4\right) \neq 0 \end{aligned} \tag{D.39}$$

it is seen that the roots of the 3-rd order polynomial of Equation (D.31) are also different.

Resolution of a 3-rd Order Polynomial

Given the 3-rd order polynomial, with real coefficients[Ango72], [Barb89]:

$$z^3 + rz^2 + sz + t = 0 \tag{D.40}$$

The change of variable $z = w - r/3$, gives a third order polynomial in w, with zero quadratic coefficient:

$$w^3 - pw - q = 0 \tag{D.41}$$

with:

$$p = -s + \frac{r^2}{3} \quad \text{and} \quad q = -t + \frac{rs}{3} - \frac{2r^3}{27} \tag{D.42}$$

setting $w = u + v$, gives:

$$u^3 + v^3 + (3uv - p)(u + v) - q = 0 \tag{D.43}$$

imposing the condition $3uv - p = 0$, the following system of equations is obtained:

$$\begin{aligned} u^3 + v^3 &= q \\ uv &= \frac{p}{3} \end{aligned} \tag{D.44}$$

Thus, u^3 and v^3 are the roots of the quadratic equation:

$$x^2 - qx + \frac{p^3}{27} = 0 \tag{D.45}$$

There are three possible cases :

$$\begin{aligned} &\text{Case 1:} \quad D = 27q^2 - 4p^3 < 0 \Rightarrow \left(\frac{p}{3}\right)^3 > \left(\frac{q}{2}\right)^2 \\ &\text{Case 2:} \quad D = 27q^2 - 4p^3 = 0 \Rightarrow \left(\frac{p}{3}\right)^3 = \left(\frac{q}{2}\right)^2 \\ &\text{Case 3:} \quad D = 27q^2 - 4p^3 > 0 \Rightarrow \left(\frac{p}{3}\right)^3 < \left(\frac{q}{2}\right)^2 \end{aligned} \tag{D.46}$$

Only in case 1, which is explained next, the roots of the 3-rd order polynomial of (D.40) are real and different. Therefore, only this case is used in the resolution of the roots of $D_{10}(x)$.

Case 1

In this case, the two solutions of the quadratic Equation (D.45) are complex conjugate, $x_1=m.e^{j\theta}$ and $x_2=m.e^{-j\theta}$, and the system given in Equation (D.44) is satisfied by:

$$(u,v)=\begin{cases}(m^{\frac{1}{3}}e^{j\frac{\theta}{3}}, m^{\frac{1}{3}}e^{-j\frac{\theta}{3}})\\(m^{\frac{1}{3}}e^{j\frac{\theta+2\pi}{3}}, m^{\frac{1}{3}}e^{j\frac{-\theta-2\pi}{3}})\\(m^{\frac{1}{3}}e^{j\frac{\theta-2\pi}{3}}, m^{\frac{1}{3}}e^{j\frac{-\theta+2\pi}{3}})\end{cases} \quad \text{(D.47)}$$

with:

$$\cos(\theta)=\frac{3q}{2p}\sqrt{\frac{3}{p}} \quad \text{and} \quad m=\frac{p}{3}\sqrt{\frac{p}{3}} \quad \text{(D.48)}$$

and the original 3-th order polynomial of Equation (D.40) has three different real zeros:

$$\begin{aligned}z_a &= m^{\frac{1}{3}}e^{j\frac{\theta}{3}}+m^{\frac{1}{3}}e^{-j\frac{\theta}{3}}-\frac{r}{3}=2\sqrt{\frac{p}{3}}\cos\left(\frac{\theta}{3}\right)-\frac{r}{3}\\ z_b &= m^{\frac{1}{3}}e^{j\frac{\theta+2\pi}{3}}+m^{\frac{1}{3}}e^{j\frac{-\theta-2\pi}{3}}-\frac{r}{3}=-2\sqrt{\frac{p}{3}}\cos\left(\frac{\pi-\theta}{3}\right)-\frac{r}{3}\\ z_c &= m^{\frac{1}{3}}e^{j\frac{\theta-2\pi}{3}}+m^{\frac{1}{3}}e^{j\frac{-\theta+2\pi}{3}}-\frac{r}{3}=-2\sqrt{\frac{p}{3}}\cos\left(\frac{\pi+\theta}{3}\right)-\frac{r}{3}\end{aligned} \quad \text{(D.49)}$$

Note that case 2 is included in case 3 when:

$$\cos(\theta)=\frac{3q}{2p}\sqrt{\frac{3}{p}}=1 \quad \text{(D.50)}$$

and, in this case, the original cubic polynomial of Equation (D.40) has three real zeros, but two of these zeros are equal:

$$z_a=2u_0-\frac{r}{3}, \quad z_b=z_c=-u_0-\frac{r}{3}$$
$$\text{with} \quad u_0=\frac{3q}{2p} \quad \text{(D.51)}$$

Calculation of the Roots of $D_{10}(x)$

The coefficients of $D_{10}(x)$ are calculated from the LPC values, $\{a_i\}$, and normalized, to obtain the 4-th order equation:

$$x^4+ax^3+bx^2+cx+d=0 \quad \text{(D.52)}$$

with the coefficients:

$$\begin{aligned}&E_5=1-a_{10}, && B_4=16E_5, && \\ &E_4=a_1-a_9, && B_3=8E_4, && a=\frac{B_3}{B_4}\\ &E_3=a_2-a_8, && B_2=4E_3-12E_5, && b=\frac{B_2}{B_4}\\ &E_2=a_3-a_7, && B_1=2E_2-4E_4, && c=\frac{B_1}{B_4}\\ &E_1=a_4-a_6, && B_0=E_5-E_3+E_1, && d=\frac{B_0}{B_4}\end{aligned} \quad \text{(D.53)}$$

Equation (D.52) is solved using the resolution of a 4-th order polynomial explained previously. Thus, the following 3-rd order equation must be solved:

$$z^3+rz^2+sz+t=0 \quad \text{(D.54)}$$

with the coefficients:

$$r=-b, \quad s=ac-4d, \quad t=d(4b-a^2)-c^2 \quad \text{(D.55)}$$

As the roots of $D_{10}(x)$ are real and different, only case 1 of the method for resolution of a 3-rd order equation explained in the previous subsection applies, and z_a, z_b, and z_c are given by Equation (D.49). If Z is chosen as the biggest in absolute value among z_a, z_b, and z_c:

$$\begin{aligned}&\text{if } \cos\theta\ge 0 \Rightarrow Z=z_a=2\sqrt{\frac{p}{3}}\cos\left(\frac{\theta}{3}\right)-\frac{r}{3}\\ &\text{else} \qquad\quad \Rightarrow Z=z_b=-2\sqrt{\frac{p}{3}}\cos\left(\frac{\pi-\theta}{3}\right)-\frac{r}{3}\end{aligned} \quad \text{(D.56)}$$

where $t=\cos(\theta)$ is given by Equation (D.48). The curves for $\cos(\pi/3-\arccos(t)/3)$ and $\cos(\arccos(t)/3)$ for $t\in[-1,+1]$ are given in Figure D.1.

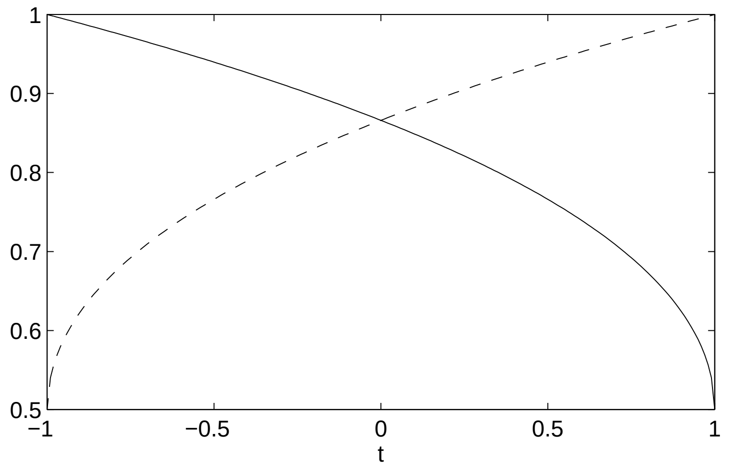


Figure D.1: The curves for cos(π/3–arccos(t)/3), plotted with a continuous line, and cos(arccos(t)/3), plotted with a dashed line, for t∈[-1,+1].

It can be noticed that the curve of interest, which corresponds to cos(arccos(t)/3) for t∈[0,+1], is almost linear. This curve can be modeled either using polynomial fitting of five coefficients, or a table of 9 elements and linear interpolation.

Finally, calculation of the roots of $D_{10}(x)$ is summarized as:

$$\begin{cases} E_5 = 1 - a_{10} \\ E_4 = a_1 - a_9 \\ E_3 = a_2 - a_8 \\ E_2 = a_3 - a_7 \\ E_1 = a_4 - a_6 \end{cases} \rightarrow \begin{cases} B_4 = 16E_5 \\ B_3 = 8E_4 \\ B_2 = 4E_3 - 12E_5 \\ B_1 = 2E_2 - 4E_4 \\ B_0 = E_5 - E_3 + E_1 \end{cases} \rightarrow \begin{cases} a = \dfrac{B_3}{B_4}, \quad b = \dfrac{B_2}{B_4} \\ c = \dfrac{B_1}{B_4}, \quad d = \dfrac{B_0}{B_4} \end{cases}$$

$$\begin{cases} r = -b \\ s = ac - 4d \\ t = d(4b - a^2) - c^2 \end{cases} \rightarrow \begin{cases} p = -s + \dfrac{r^2}{3} \\ q = -t + \dfrac{rs}{3} - \dfrac{2r^3}{27} \end{cases} \rightarrow \cos(\theta) = \frac{3q}{2p}\sqrt{\frac{3}{p}}$$

(D.57)

$$\begin{cases} s\_co = \mathrm{sign}[\cos(\theta)] \\ a\_co = \mathrm{abs}[\cos(\theta)] \end{cases} \rightarrow Z = s\_co \cdot 2\sqrt{\frac{p}{3}}\cos\left(\frac{a\cos(a\_co)}{3}\right) - \frac{r}{3}$$

$$\varepsilon = \mathrm{sign}\left(\frac{az}{2} - c\right)$$

$$\begin{cases} p_1 = \dfrac{a}{2} + \sqrt{\left(\dfrac{a}{2}\right)^2 - b + z}\,, \quad q_1 = \dfrac{z}{2} + \varepsilon \cdot \sqrt{\left(\dfrac{z}{2}\right)^2 - d} \\ p_2 = \dfrac{a}{2} - \sqrt{\left(\dfrac{a}{2}\right)^2 - b + z}\,, \quad q_2 = \dfrac{z}{2} - \varepsilon \cdot \sqrt{\left(\dfrac{z}{2}\right)^2 - d} \end{cases}$$

$$\begin{cases} r_1 = -\dfrac{p_1}{2} - \sqrt{\left(\dfrac{p_1}{2}\right)^2 - q_1}\,, \quad r_2 = -\dfrac{p_1}{2} + \sqrt{\left(\dfrac{p_1}{2}\right)^2 - q_1} \\ r_3 = -\dfrac{p_2}{2} - \sqrt{\left(\dfrac{p_2}{2}\right)^2 - q_2}\,, \quad r_4 = -\dfrac{p_2}{2} + \sqrt{\left(\dfrac{p_2}{2}\right)^2 - q_2} \end{cases}$$

(D.58)

The C program for the resolution of the roots of $D_{10}(x)$ is given in [Gras97b]. This calculation needs the following operations: 20 multiplications, 34 add/sub, 2 divisions and 5 square roots. Three comparison/swapping operations are needed for root ordering, as explained in the next section.

D.5. Optimization of the Root Sorting

In order to obtain the five intervals where only one even-suffixed and one odd-suffixed LSP are contained, the roots of $D_{10}(x)$ must be ordered. Given that these roots are related by Equation (D.58), only the following ordered sets are possible:

$$\begin{gathered} r_1 < r_2 < r_3 < r_4 \\ r_1 < r_3 < r_2 < r_4 \\ r_1 < r_3 < r_4 < r_2 \\ r_3 < r_1 < r_2 < r_4 \end{gathered} \qquad (D.59)$$

The ordering algorithm needs three comparisons and swapping.

D.6. Property of the Roots of $D_{10}(x)$

Property

The roots of $D_{10}(x)$ are real, different and inside the interval (-1,1).

Proof

The Levinson-Durbin recursion, given in Equation (5.12), leads to the recursion [Proa89]:

$$\begin{aligned}
&A_0(z) = B_0(z) = 1\\
&\text{for}\quad m = 1,\ldots,p:\\
&\qquad A_m(z) = A_{m-1}(z) + k_m z^{-1} B_{m-1}(z)\\
&\qquad B_m(z) = k_m A_{m-1}(z) + z^{-1} B_{m-1}(z)
\end{aligned} \tag{D.60}$$

where $B_m(z)$ are the reciprocal polynomials of $A_m(z)$, given by:

$$B_m(z) = z^{-m} A_m(z^{-1}) \tag{D.61}$$

If the LPC analysis filter $A_p(z)$ is minimum phase (i.e., all the zeros are inside the unit circle) then the reflection coefficients $\{k_1,\ldots,k_p\}$ are bounded in magnitude by one. Conversely, if the reflection coefficients $\{k_1,\ldots,k_p\}$ are bounded in magnitude by one, then $A_p(z)$ is minimum phase as well as all the lower-order LPC analysis filters $\{A_1(z),\ldots,A_{p-1}(z)\}$ [Proa89]. Note that the $A_p(z)$ obtained using the Levinson-Durbin recursion is minimum phase.

For any of the filters $\{A_1(z),\ldots,A_p(z)\}$, a symmetrical polynomial $P_m(z)$ and an antisymmetrical polynomial $Q_m(z)$ can be formed by adding and subtracting to $A_m(z)$ its time reversed system function $z^{-(m+1)}A_m(z^{-1})$ [Kaba86]:

$$\begin{aligned}
P_m(z) &= A_m(z) + z^{-1} B_m(z)\\
Q_m(z) &= A_m(z) - z^{-1} B_m(z)
\end{aligned} \tag{D.62}$$

If m is even, $P_m(z)$ and $Q_m(z)$ have a zero at $z = -1$ and at $z = +1$, respectively. If m is odd, $Q_m(z)$ have a zero at $z = -1$ and a zero at $z = +1$ [Kaba86]. These trivial zeros could be removed by polynomial division :

$$\begin{aligned}
P'_m(z) &= \frac{P_m(z)}{(1+z^{-1})} \quad \text{and} \quad Q'_m(z) = \frac{Q_m(z)}{(1-z^{-1})}, \quad m \text{ even}\\
P'_m(z) &= P_m(z) \qquad\quad \text{and} \quad Q'_m(z) = \frac{Q_m(z)}{(1-z^{-2})}, \quad m \text{ odd}
\end{aligned} \tag{D.63}$$

The polynomials $P'_m(z)$ and $Q'_m(z)$ are symmetrical, and if $A_m(z)$ is minimum phase, then the roots of $P'_m(z)$ and $Q'_m(z)$ lie on the unit circle and are interlaced [Soon84].

The m-th order LSP parameters are defined as the angular positions of the roots of $P'_m(z)$ and $Q'_m(z)$ located on the upper semicircle of the z-plane, they are denoted as $\{\omega_i\}$, in the angular frequency domain, and their ordering property is expressed as [Kaba86]:

$$0 < \omega_1 < \omega_2 < \ldots < \omega_m < \pi \tag{D.64}$$

Combining Equation (D.60) with Equation(D.62), gives:

$$\begin{aligned}
2P_m(z) &= P_{m-1}(z)(1+k_m)\left(1+z^{-1}\right) + Q_{m-1}(z)(1-k_m)\left(1-z^{-1}\right)\\
2Q_m(z) &= P_{m-1}(z)(1+k_m)\left(1-z^{-1}\right) + Q_{m-1}(z)(1-k_m)\left(1+z^{-1}\right)
\end{aligned} \tag{D.65}$$

and using Equations (D.63) and (D.65):

$$\frac{P'_{10}(z) - Q'_{10}(z)}{2} = \frac{-z^{-1}(1-k_{10})Q_9(z)}{(1-z^{-2})} = -z^{-1}(1-k_{10})Q'_9(z) \tag{D.66}$$

The polynomials $P'_{10}(z)$, $Q'_{10}(z)$ are symmetric and of 10-th order, while $Q'_9(z)$ is symmetric and of 8-th order. The symmetry of these three polynomials is used to group their terms as:

$$\begin{aligned}
P'_{10}(z) &= z^{-5}\cdot\left[(z^{+5}+z^{-5}) + p'_1(z^{+4}+z^{-4}) + \ldots + p'_5\right]\\
Q'_{10}(z) &= z^{-5}\cdot\left[(z^{+5}+z^{-5}) + q'_1(z^{+4}+z^{-4}) + \ldots + q'_5\right]\\
Q'_9(z) &= z^{-4}\cdot\left[(z^{+4}+z^{-4}) + \alpha'_1(z^{+3}+z^{-3}) + \ldots + \alpha'_4\right]
\end{aligned} \tag{D.67}$$

Introducing Equation (D.67) into Equation (D.66), applying the mapping $x = \cos(\omega)$ and using Equation (D.19), the following relation is obtained:

$$D_{10}(x) = \frac{P'_{10}(x) - Q'_{10}(x)}{2} = (1-k_{10})Q'_9(x) \tag{D.68}$$

As k_{10} is bound in magnitude by one, the factor $(1-k_{10})$ is different from zero. The roots of $D_{10}(x)$ correspond to the roots of $Q'_9(x)$, which are the even-suffixed LSPs of a 9-th order LPC system, in the "x domain", in which $x = \cos(\omega)$. From the ordering property of Equation (D.64), it is seen that these roots are real, different, and in the interval (-1,+1).

D.7. References

[Ango72] A. Angot, *Complements de mathematiques a l'usage des ingenieurs de l'electrotechnique et des telecommunications*, Masson, Paris, 1972.

[Barb89] E. Barbeau, *Polynomials*, Springer, New York, 1989.

[Gras97b] S. Grassi, *DSP56001 Implementation of the Spectral Analysis and Quantization for the CELP FS1016 Speech Coder*, IMT Report No 421 PE 10/97, University of Neuchâtel, IMT, Oct. 1997.

[Kaba86] P. Kabal and P. Ramachandran, "The Computation of Line Spectral Frequencies Using Chebyshev Polynomials", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 34, No. 6, pp. 1419-1426, 1986.

[Proa89] J. Proakis and D. Manolakis, *Introduction to Digital Signal Processing* (Chapter 7), Macmillan, New York, 1989.

[Soon84] F. Soong and B. Juang, "Line Spectrum Pair (LSP) and Speech Data Compression", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing,* ICASSP'84, pp. 1.10.1-1.10.4, 1984.

Appendix E
Quantized-search Kabal Method

The reader is reminded that the name "quantized-search Kabal" refers to the version of "quantized-search Kabal" algorithm which uses both "horizontal single-correction" and "enhanced vertical coupled-correction" criteria (see § 6.3).

E.1. Maximum Number of Evaluations

To obtain the maximum number of evaluation in "quantized-search Kabal" and "quantized-search Chan" algorithms, one of the longest possible search paths on the quantization tables of Figure 5.5 has to be found. The search is constrained by the ordering property of the LSP parameters given in Equation (5.25). One of longest path can be easily found by inspection of the quantization tables of Figure 5.5. A weight is assigned to each point of the *i*-th quantization table, corresponding to the maximum length to arrive to this point, starting from the first element of the first table, this weight is then used to find the longest path to arrive to every point of the *i+1*-th quantization table, and the process is repeated up to the *10*-th quantization table. One of the longest path, obtained with this method is described by the set of LSP indices { 1, 6, 4, 5, 14, 5, 4, 3, 3, 7 }. The length of this path is 52. Thus 52 evaluations are needed to search through this path. Additionally, the search could advance one extra point (except in the last LSP which is the last of the table) and then come back by means of a single correction. This would add 9 more evaluations. One extra evaluation per LSP could be used to test single correction, for a total of 10 evaluation. Thus the maximum number of possible evaluations is given by 52 + 9 + 10 = 71. In practice, the maximum number of evaluations found by simulation on the whole TIMIT database is 68.

E.2. Differences with the Reference Algorithm

The “quantized-search Kabal” algorithm was compared with the high accuracy method followed by quantization (reference algorithm). The differences between the LSP indices calculated with the algorithm under evaluation and the reference algorithm were counted, and the results are given in Table E.1. The number of frames containing one, two, three, four and more than four differences of one on the LSP indices are denoted as n1, n2, n3, n4 and n5 respectively. The number of frames containing at least one difference bigger than one on the LSP indices is denoted as nn.

	n_1	n_2	n_3	n_4	n_5	*nn*
“Q.-search Kabal”	0	706	3	0	0	2

Table E.1 : Comparison among “quantized-search Kabal” algorithm, and high accuracy method + quantization in the “x-domain”, in terms of differences in the obtained indices.

It is seen that there are no frames containing one difference of one on the LSP indices. This is due to the fact that the single-correction criterion proposed in Equations 6.6 to 6.8 is perfectly equivalent to the “horizontal single-correction” criterion.

The 706 frames containing two differences of one on the LSP indices are due to the fact that the “enhanced vertical coupled-correction” criterion of Equations 6.11 and 6.12 does not correspond exactly to the “horizontal coupled-correction” criterion. Thus there are some missed “coupled-corrections” and coupled corrections that were erroneously done. These 706

frames have an average spectral distortion of 2.42 dB while the same frames, using the reference algorithm, have a spectral distortion of 2.39 dB. Furthermore this frames correspond to the 0.1098 % of the tested frames. Thus they do not affect significantly the quantization performance.

In Table E.1 it is observed that there are three frames containing three differences of one on the LSP indices and two frames containing at least one difference bigger than one on the LSP indices. Four of these five particular cases correspond to missed zero-crossings due to the coarse quantization grid, as observed in Figure E.1, and one of the cases corresponds to a doubly detected zero crossing. The spectral distortion was measured on these five frames using both “quantized-search Kabal” and the reference algorithm, and is given in Table E.2.

Speech file / frame	***Reference***	***“Q.-search Kabal”***
2371 / 57	3.9462	15.9480
2785 / 24	2.2258	10.5166
2858 / 113	0.5224	5.7037
5199 / 9	3.4225	3.2096
6205 / 50	1.4617	5.8248

Table E.2 : Spectral distortion for the frames with missed or doubly-detected zero-crossing.

It is observed that most of these frames introduce a very high distortion, but they correspond only to the 0.0008 % of the tested frames. Furthermore, in listening tests using the CELP FS1016 speech coder, these cases did not introduce additional audible distortion. Thus, to keep the low complexity of “quantized-search Kabal” it was decided not to add any extra computation to avoid these unlikely conditions.

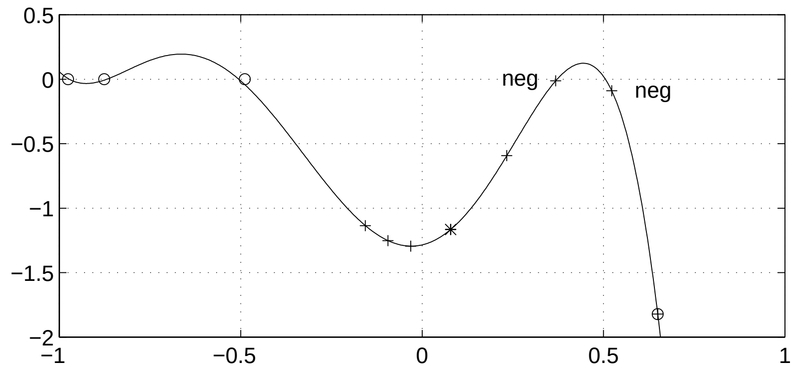


Figure E.1: Missing zero-crossing in speech file 2371, frame 57, when searching the 7-th LSP.

Sara Grassi

Electronics and Signal Processing Laboratory
Institute of Microtechnology, University of Neuchâtel,
Breguet 2, CH-2000 Neuchâtel, Switzerland
sara.grassi@imt.unine.ch

Born March 9th, 1966, in Maracay, Venezuela.

Citizenship :	Venezuelan / Italian
Current Field of Research :	Speech coding and enhancement
Languages :	Spanish / Italian / English / French

Educational Degrees

October 1990 - October 1991 : M.Sc. in Communications & Digital Signal Processing, Imperial College of Science and Technology, University of London.

September 1982 - January 1988 : Electronic Engineer, Universidad Simon Bolivar, Caracas, Venezuela.

Working Experience

Since March 1992 : Research Assistant at the Institute of Microtechnology, University of Neuchâtel, Switzerland.

March 1988 - August 1990 : Project Engineer with Microtel S.A., Venezuela.

April 1987 - Sept. 1987 : Degree project undertaken in association with GENTE C.A., Venezuela.

July 1985 - August 1985 : Industrial training with Maraven S.A., a branch of "Petroleos de Venezuela".

Teaching Experience

Since March 1992 definition and supervision of student projects at the University of Neuchâtel, Switzerland.

Sept. 1985 - March 1987 : Teaching assistant in Electronics I, Electronics II, Electronics III at the Electronics and Circuits Department, Universidad Simon Bolivar, Caracas, Venezuela.

Publications

S. Grassi, M. Ansorge, and F. Pellandini, "Fast LSP Calculation and Quantization with Application to the CELP FS1016 Speech Coder", Proc. EUSIPCO'98, Sept. 1998.

S. Grassi, M. Ansorge, and F. Pellandini, "Optimized Real Time Implementation of Spectral Analysis and Quantization for the CELP FS1016 Speech Coder", Proc. Cost #254 Workshop on Intelligent Communications, L'Aquila, Italy, June 4-6, 1998.

S. Grassi, A. Dufaux, M. Ansorge, and F. Pellandini, "Efficient Algorithm to Compute LSP Parameters from 10-th order LPC Coefficients", Proc. ICASSP'97, Vol. 3, pp. 1707-1710, 1997.

S. Grassi, A. Heubi, M. Ansorge, and F. Pellandini, "Study of a VLSI Implementation of a Noise Reduction Algorithm for Digital Hearing Aids", Proc. EUSIPCO'94, Vol.3, pp. 1661-1664, 1994.

S. Grassi, "Noise Elimination in Speech Using Adaptive Filtering in Subbands", Master Thesis, Imperial College, London, October 1991.

A. Heubi, S. Grassi, M. Ansorge, and F. Pellandini, "A Low Power VLSI Architecture for Digital Signal Processing with an Application to Adaptive Algorithms for Digital Hearing Aids", Proc. EUSIPCO'94, Vol. 3, pp. 1875-1878, 1994.

Errata

1. Page 95, Equation 5.33

The diagonal terms of M_5* are modified:

$$\mathbf{M}_5 = \begin{bmatrix} 2\alpha_1+\alpha_2-2 & 1 & 0 & 0 & 0 \\ \alpha_2\alpha_3 & \alpha_3+\alpha_4-2 & 1 & 0 & 0 \\ 0 & \alpha_4\alpha_5 & \alpha_5+\alpha_6-2 & 1 & 0 \\ 0 & 0 & \alpha_6\alpha_7 & \alpha_7+\alpha_8-2 & 1 \\ 0 & 0 & 0 & \alpha_8\alpha_9 & \alpha_9+\alpha_{10}-2 \end{bmatrix}$$

$$\mathbf{M}_5^* = \begin{bmatrix} \alpha_2^*-2 & 1 & 0 & 0 & 0 \\ \alpha_2^*\alpha_3^* & \alpha_3^*+\alpha_4^*-2 & 1 & 0 & 0 \\ 0 & \alpha_4^*\alpha_5^* & \alpha_5^*+\alpha_6^*-2 & 1 & 0 \\ 0 & 0 & \alpha_6^*\alpha_7^* & \alpha_7^*+\alpha_8^*-2 & 1 \\ 0 & 0 & 0 & \alpha_8^*\alpha_9^* & \alpha_9^*+\alpha_{10}^*-2 \end{bmatrix} \qquad (5.33)$$

2. Page 155, Equation 7.7

The sub-indices of all the terms for the equation for ε_m in the recursion are decremented by one:

$$k_1 = -\frac{r_1'}{r_0'}$$

$$\varepsilon_1 = r_0'(1-k_1^2)$$

$$k_2 = -\frac{r_2' + k_1 \cdot r_1'}{\varepsilon_1}$$

$$a_2(1) = \frac{k_1+k_2.k_1}{16}, \quad a_2(0) = \frac{1}{16}, \quad a_2(2) = \frac{k_2}{16}$$

for $m = 3,\ldots,10$:

$$\left[\begin{array}{l} \varepsilon_{m-1} = \varepsilon_{m-2}\left(1-k_{m-1}^2\right) \\ a_m(m) = -\dfrac{\dfrac{r_m'}{16} + \sum_{i=1}^{m-1} a_{m-1}(i).r_{m-i}'}{\varepsilon_{m-1}}, \quad a_m(0) = \dfrac{1}{16} \\ k_m = 16 \cdot a_m(m) \\ a_m(j) = a_{m-1}(j) + k_m . a_{m-1}(m-j) \qquad \text{for} \quad 1 \le j \le m \end{array}\right. \qquad (7.7)$$