# Waveform Interpolation Speech Coder at 4 kb/s

*Eddie L. T. Choy*

Department of Electrical and Computer Engineering
McGill University
Montréal, Canada

August 1998

---

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering.

## Abstract

Speech coding at bit rates near 4 kbps is expected to be widely deployed in applications such as visual telephony, mobile and personal communications. This research focuses on developing a speech coder based on the waveform interpolation (WI) scheme, with an attempt to deliver near toll-quality speech at rates around 4 kbps. A WI coder has been simulated in floating-point using the C programming language. The high performance of the WI model has been confirmed by subjective listening tests in which the unquantized coder outperforms the 32 kbps G.726 standard (ADPCM) 98% of the time under clean input speech conditions; the reconstructed speech is perceived to be essentially indistinguishable from the original. When fully quantized, the speech quality of the WI coder at 4.25 kbps has been judged to be equivalent to or better than that of G.729 (the ITU-T toll-quality 8 kbps standard) for 45% of the test sentences. Further refinements of the quantization techniques are warranted to bring the coder closer to the toll-quality benchmark. Yet, the existing implementation has produced good quality coded speech with a high degree of intelligibility and naturalness when compared to the conventional coding schemes operating in the neighbourhood of 4 kbps.

## Sommaire

Dans un futur proche, le codage de la parole à des taux autour de 4 kbps devrait être largement utilisé dans des applications comme, la teléphonie visuelle, et les communications personnelles et mobiles. Cette recherche a pour but de développer un codeur de parole basé sur l'interpolation d'un signal (abrégé WI pour *waveform interpolation*), avec comme objectif une reconstruction fidèle de la parole à des débits aussi faibles que 4 kbps. Un codeur basé sur le modèle WI a été simulé en arithmétique flottante en utilisant le language C. Les hautes performances du modèle ont été confirmées par des tests d'écoute dans lesquels la qualité de parole du codeur sans quantification est meilleure que le standard 32 kbps G.726 (ADPCM) dans 98% des cas lorsque la parole utilisée au départ était sans bruit. On peut conclure que la synthèse est perçue comme étant essentiellement indifférentialle de la parole originale. Quand les paramètres du codeur sont complètement quantifiés, la qualité de parole du codeur WI à 4.25 kbps a été jugée comme étant équivalente ou meilleure que le G.729 (le standard ITU-T toll-quality 8 kbps) pour 45% des sequences de test. Des améliorations plus poussées des techniques de quantification sont nécessaires pour que le codeur permette une reconstruction encore plus proche de la reconstruction fidèle. Néanmoins, le programme existant a donné de la parole codée de bonne qualité avec un haut degré d'intelligibilité et de naturel comparé aux autres codeurs conventionnels fonctionnant autour de 4 kbps.

# Acknowledgments

I would like to express my sincere thanks to my supervisor, Professor Peter Kabal, for his guidance and support throughout my graduate studies at McGill University. Also, I am thankful to Dr. Jacek Stachurski for co-implementing the waveform interpolation speech coder. This research could not have been possible without their technical expertise, critical insight and enlightening suggestions.

Moreover, I acknowledged all my fellow graduate students in the Telecommunications and Signal Processing Laboratory for their encouragement and companionship. Special thanks go to Hossein, Nadim and Khaled who constantly gave me both technical and non-technical advice. I am also obliged to Florence who helped me with the French abstract.

I am thankful to Jianming, Michael, Johnny and Mohammad who participated in the listening tests for this research. The postgraduate scholarship awarded by the Natural Sciences and Engineering Research Council of Canada is appreciated.

My deepest gratitude goes to my fiancée Jane for her love and understanding, and also to our respective families for their continuous support and encouragement in the past two years.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **ADPCM** | Adaptive Differential Pulse-Code Modulation |
| **CDMA** | Code Division Multiple Access |
| **CELP** | Code-Excited Linear Prediction |
| **CODEC** | Encoder and Decoder |
| **CW** | Characteristic Waveform |
| **DCVQ** | Dimension Conversion Vector Quantization |
| **DoD** | Department of Defense (U.S.) |
| **DSP** | Digital Signal Processing |
| **DTFS** | Discrete-Time Fourier Series |
| **EVRC** | Enhanced Variable Rate Codec |
| **FBR** | Fixed Bit-Rate |
| **FS** | Federal Standard (U.S.) |
| **GLA** | Generalized Lloyd Algorithm |
| **IMBE** | Improved Multi-Band Excitation |
| **ITU** | International Telecommunication Union |
| **ITU-T** | ITU Telecommunication Standardization Sector |
| **LD-CELP** | Low-Delay Code Excited Linear Prediction |
| **LP** | Linear Prediction |
| **LPC** | Linear Predictive Coding |
| **LSF** | Line Spectral Frequency |
| **LSP** | Line Spectral Pair |
| **MBE** | Multi-Band Excitation |
| **MELP** | Mixed Excitation Linear Prediction |
| **MIPS** | Million Instructions Per Second |
| **MOS** | Mean Opinion Score |
| **MSE** | Mean Square Error |
| **PCM** | Pulse Code Modulation |
| **PWI** | Prototype Waveform Interpolation |
| **REW** | Rapidly Evolving Waveform |
| **SEW** | Slowly Evolving Waveform |
| **SNR** | Signal-to-Noise Ratio |
| **V/UV** | Voiced/Unvoiced |
| **VBR** | Variable Bit-Rate |
| **VDVQ** | Variable Dimension Vector Quantization |
| **VQ** | Vector Quantization |
| **WI** | Waveform Interpolation |

# Chapter 1

# Introduction

## 1.1 Motivation for Speech Coding

In modern digital systems, a speech signal is represented in a digital format — a sequence of binary bits. It is often desirable for the signal to be represented by as few bits as possible. For storage applications, lower bit usage means less memory is required. For transmission applications, lower bit rate means less bandwidth, power and/or memory. It is therefore cost-effective to use an efficient speech compression algorithm in a digital speech storage or transmission system. Speech coding is the technology to offer such compression algorithms.

Although larger bandwidth has become available in wired communications as a result of the rapid development in optical transmission media, there is still a growing need for bandwidth conservation, particularly in the wireless and satellite communications. At the same time, with the growing trend of multimedia communications and other speech-related applications such as digital answering machine, the demand on memory conservation in voice storage system is increasing. These dual requirements will definitely keep speech coding a lively research and development area for the future.

In addition, the emergence of much faster DSP microprocessors provides speech coding researchers even more incentives for getting new and improved speech coding algorithms, algorithms which are allowed to have more computational effort than ever before. An explosion of research work on speech coding is expected to be seen in the coming millennium.

## 1.2 Propaedeutic of Speech Coding

### 1.2.1 Components in a Speech Coder

A speech coder (also known as a speech *codec*) always consists of an en*cod*er and a *dec*oder. The encoder is the compression function while the decoder is the decompression function. They usually coexist in typical speech transmission/storage systems. Figure 1.1 illustrates an example of such a system. At the compression stage, the speech encoder takes the original digital speech signal and produces a low-rate bitstream. This bitstream is then transmitted to a receiver or to a storage device. At the decompression stage, the speech decoder tries to undo what the encoder has done and constructs an approximation of the original signal from the compressed bitstream. Thus, the decoder should be structurally an approximate inverse of the encoder.

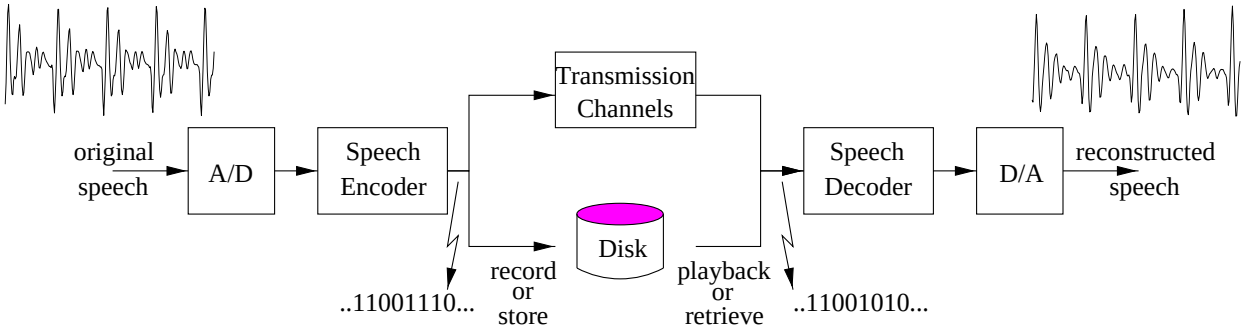


**Fig. 1.1** A block diagram of a speech transmission/storage system

### 1.2.2 Concept of a *Frame* and a *Subframe*

Speech is a time-varying signal [1]. In order to analyze a speech signal efficiently, a speech coder generally partitions the signal into successive blocks such that the samples within each block can be considered to be reasonably stationary. These blocks are referred to as *frames*. Furthermore, some processing steps may require a higher time-resolution and needs to be performed over smaller blocks. These smaller blocks are often called *subframes*.

### 1.2.3 Performance Dimensions

In selecting a speech coder, certain performance aspects must be considered and trade-offs need to be made. Different applications require the coder to be optimized for different dimensions or some balance between the dimensions. We have chosen eight important dimensions and each of these will be briefly described as follows:

(i) Average bit-rate: This parameter is usually measured in bits per second (bps). The word 'average' is used here because some coders operate at variable-rate, as opposed to fixed-rate. Note that all the bit-rates mentioned in this thesis do not include any additional bit-rates used for error corrections.

(ii) Speech quality: A popular method to evaluate speech quality is the MOS scale (Mean Opinion Score) which is a subjective measurement. Listeners are asked to give evaluations on speech quality based on a five-point scale — bad, poor, fair, good and excellent. Because of a wide variation among listeners, the MOS test requires a large number of speech data, speakers, and listeners to get an accurate rating of a speech coder. In North America, a MOS scale of between 4 and 4.5 generally means *toll-quality* while *synthetic quality* falls below 3.5. There are also objective measurements available such as SNR, known as signal-to-noise ratio. Generally, the objective measurements are not as lengthy and costly as the subjective ones, but the former do not fully account for perceptual properties of the human hearing system.

(iii) Algorithmic delay: As mentioned earlier, most speech coders tend to process samples in blocks, so a time delay often exists between the original and the coded speech. In the speech coding context, this time delay is referred to as the algorithmic delay which is generally defined as the sum of (i) the length of currently processed block of speech and (ii) the length of the look-ahead which is needed to process the samples of the current block. In some applications like telephony, there is often a strict limitation on the time delay. In others like voice storage systems, more delay can be tolerated.

(iv) Computational complexity: Speech coding algorithms are usually required to run on a single DSP chip. Memory usage and speed are therefore the two most important contributors to complexity. The former is specified by the size

of RAM used in executing an algorithm. The latter is measured in million instructions per second which is commonly known as MIPS. This MIPS can be measured in either a fixed-point or a floating-point processor. An algorithm of large complexity not only requires a faster chip to implement in real-time, it also results in a high power consumption in hardware which is extremely disadvantageous for portable systems.

(v) Channel-error sensitivity: This parameter is to measure the speech coder's robustness against channel errors, errors which are often caused by the presence of channel noise, signal fading and intersymbol interference. The channel-error issue has become increasingly important in speech coding as many newly developed speech coders are used in wireless communications. In such systems, the speech coder must be able to give reasonable speech quality with error rates as high as 10%.

(vi) Robustness against acoustic background noise: In real-word applications, we are faced with various types of background acoustic noise such as car-, babble-, street- and office-noise. Thus, it is essential that the performance of the speech coding algorithm does not suffer unduly from such adverse environments. The issue of background noise becomes particularly crucial when it comes to applications like military and mobile communications. In fact, the 1996 US D.o.D (Department of Defense) 2.4 kbps vocoder competition required all speech coder algorithms to have good performance in both quiet and noisy environments [2].

(vii) Encoded speech bandwidth: This means the bandwidth of a speech signal for which a coder is intended to encode. Narrowband speech coders are found in typical telephone transmission which requires a bandwidth from 200 to 3400 Hz. On the other hand, applications of wideband speech coding with bandwidth ranging from 7 to 20 kHz include audio transmission, teleconferencing and teleteaching.

(viii) Additional acoustic features: Some speech coders have the abilities to provide speech compression as well as other speech processing features. Examples of such features are pitch and formants modifications, fast/slow voice playback speech control without affecting pitch track, etc.

### 1.2.4 Quantization

In theory, a precise digital representation of a single or a set of numerical values requires an infinite number of bits, which is not an achievable goal. Therefore, the difference between the original value and its digitized version is always present when a signal is digitally transmitted or stored. The goal of *quantization* is to minimize this difference, which is also known as the quantization noise or quantization error.

There are two basic types of quantization: *scalar quantization* and *vector quantization* (VQ). A scalar quantizer maps a single numerical value to the nearest approximating value from a predetermined finite set of allowed values [3]. Vector quantization, on the other hand, operates on a block of values. Rather than quantizing each of the values in the block independently, VQ treats the whole block as a single entity or vector and represents it as a single vector index, and at the same time, minimizes the distortion introduced. In this way, coding efficiency can be greatly enhanced if there is redundant information within the block of values (the values within the block are correlated) [1].

In the context of VQ, a collection of the possible vector representations is referred to as a *codebook*. Each of these vector representations in a codebook defines a *codeword*. Further, the number of codewords in a codebook is referred to as the *size* of the codebook and the number of elements in each codeword is called the *dimension* of a codebook.

Depending on the specific applications, there are many distortion measures that can be adopted to evaluate and/or design a quantizer. The most ubiquitous one is the Euclidean distance measure. Distance measures which take perceptual relevance into account are also available. They are advantageous to speech coders, particularly when coding vectors of spectral parameters since human ear has a variable sensitivity to different frequencies and intensities. The details about human perceptual sensitivity will be further described in Section 1.4.

Due to its high coding efficiency, VQ has spurred tremendous research interest. Many different VQ-related algorithms have been developed to create and search codebooks efficiently, algorithms such as gain-shape VQ, split VQ and multistage VQ [4]. Recently, variable-dimension vector quantization (VDVQ) has drawn attention as well. Unlike conventional VQ, VDVQ is capable to handle variable-dimension input

[1] Even for uncorrelated samples, VQ may offer some advantages over scalar quantization [3, p.347].

vectors and each input vector can be quantized with a single universal codebook [5].

## 1.3 Speech Production and Properties

Many contemporary speech coders lower their bit rate consumptions by removing predictable, redundant or pre-determined information in human speech. In the search for better speech coding algorithms, it is therefore important to have a good understanding of the production of human speech and the properties of speech signals.

Physiologically, human speech is produced when air is exhaled from the lungs, through the vocal folds and the vocal tract to the mouth opening. From the signal processing point of view, this speech production mechanism can be modeled as an excitation signal exciting a time-varying filter (the vocal tract), which amplifies or attenuates certain sound frequencies in the excitation. The vocal tract is modeled as a time-varying system because it consists of a combination of the throat, mouth, the tongue, the lip, and the nose, that change shape during generation of speech. The properties of the excitation signal highly depends on the type of speech sounds, either voiced or unvoiced. Examples of voiced speech are vowels (/a/, /i/, /o/, /u/) while fricatives such as /p/ and /k/ are examples of unvoiced sounds.

The excitation for voiced speech is a quasi-periodic signal generated by the periodical abduction and adduction of the vocal folds where the airflow from the lungs is intercepted. Since the opening between the vocal folds is called the glottis, this excitation is sometimes referred as a glottal excitation. Generally, the vocal tract filter is considered linear in nature and therefore, not able to alter the periodicity of the glottal excitation. Hence, voiced sounds are quasi-periodic in nature as well.

For unvoiced speech, the vocal folds are widely open. The excitation is formed as the air is forced through a narrow constriction at some point in the vocal tract and creates a turbulence. The unvoiced speech and its excitation signal both tend to be noise-like and lower in energy as compared to the voiced case. Figure 1.2a illustrates an example of both unvoiced and voiced speech segment in time domain.

In spectral domain, due to the quasi-periodicity, voiced speech possesses a prominent harmonic line structure as depicted in figure 1.2c. The spacing between the harmonics is called the fundamental frequency. The envelope of the spectrum, also known as the formant structure, is characterized by a set of peaks, each of which is called a formant. The formant structure (poles and zeros of the envelope) is primar-

ily attributed to the shape of the vocal tract. Thus, by moving the tongue, jaw or lips, the structure would be changed correspondingly. Also, the envelope falls off at about -6 dB/octave due to the radiation from the lips and the nature of the glottal excitation [6].

Figure 1.2b shows the power spectrum of the unvoiced segment. As opposed to

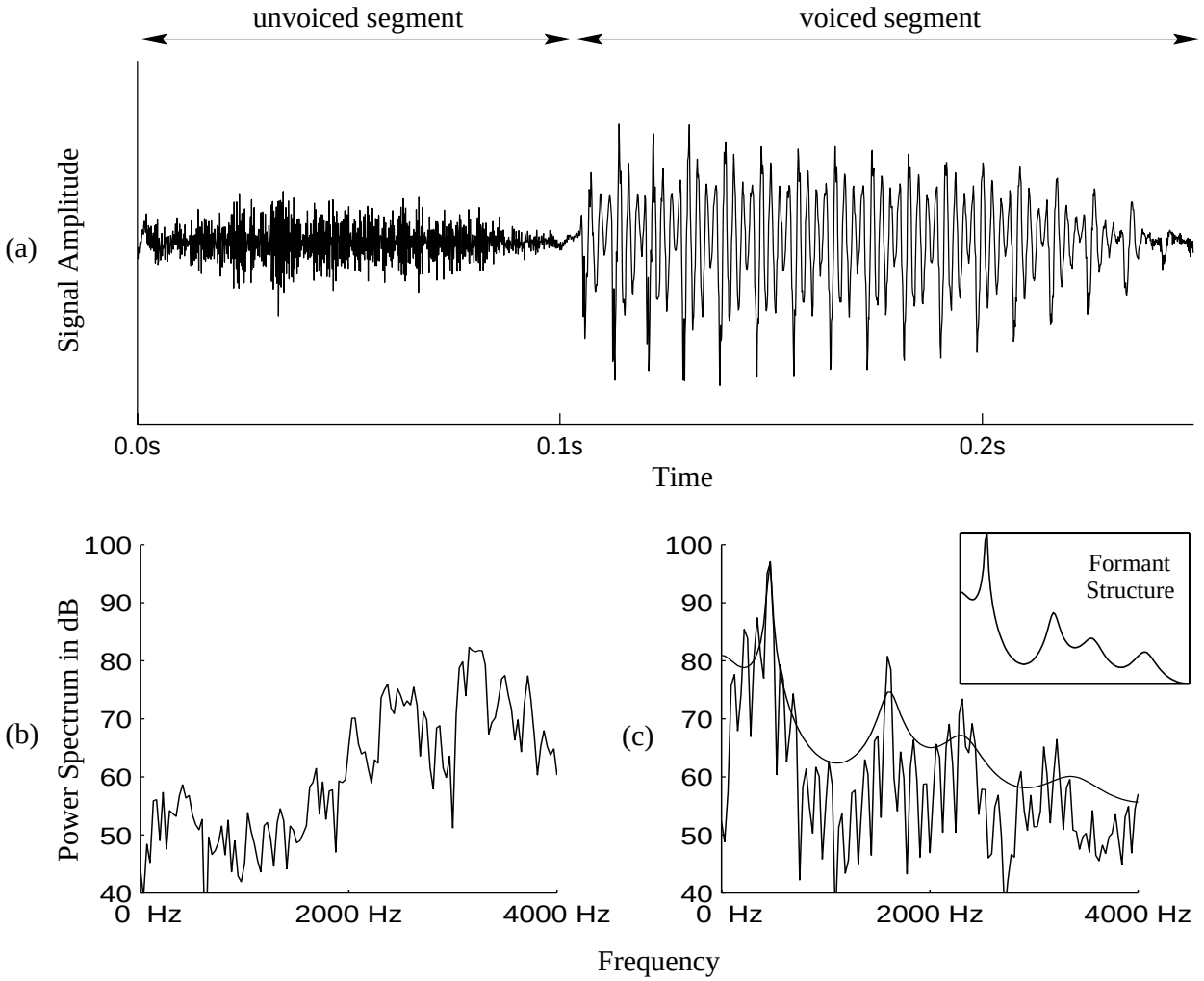


**Fig. 1.2** Time and frequency representations of a voiced and unvoiced speech segment. (a) A speech segment consists of an unvoiced and voiced segment in time domain. (b) The power spectrum for a 32 ms unvoiced segment starting at 50 ms. (c) The power spectrum and the corresponding formant structure for a 32 ms voiced segment starting at 150 ms. Both (b) and (c) are calculated based on a 32 ms Hanning window.

the voiced spectrum, there is relatively less useful spectral information embedded in an unvoiced segment. It does not have any distinctive harmonics and it is rather flat, broadband and noise-like.

## 1.4 Human Auditory Perception

In order to reach maximal performance in a speech coder, it is also essential to take advantage of human auditory system, even though it is not fully understood yet. Generally, exploiting the perceptual properties of the ear could lead to significant improvement in performance of a speech coder. This is particularly true as we pursue lower and lower bit-rate speech coders while avoiding major audible degradation.

One of the well-known properties of the auditory system is the auditory masking which has a strong effect on the perceptibility of one signal in the presence of another [6]. Noise is less likely to be heard at frequencies of strong speech energy (e.g., formants) and more likely to be heard at frequencies of low speech energy (e.g., valleys). *Spectral masking* is a popular technique that takes advantage of this perceptual limitation by concentrating most of the noise (resulting from compression) in high-energy spectral regions where it is least audible.

It is reported that humans perceive voiced and unvoiced sounds differently. For voiced signals, the correct degree of periodicity and the temporal continuity in voiced segments [7, 8, 9] are of great importance to human perception (although excessive periodicity would lead to reverberation and buzziness). In spectral domain, the amplitudes and the locations of the first three formants (usually below 3 kHz) and the spacing between the harmonics are important [10].

For unvoiced signals, it has been shown in [11] that the unvoiced speech segments can be replaced by a noise-like signal with a similar spectral envelope without a drop in the perceived quality of the speech signal.

In both voiced and unvoiced cases, the time envelope of the speech signal contributes to intelligibility and naturalness [12, 13].

## 1.5 Speech Coding Standardizations

The standardization of high quality low-bit-rate narrowband[2] speech coding has been intensifying since the beginning of this decade. In 1994, the International Telecommunication Union (ITU) adopted the LD-CELP (Low-Delay Code-Excited Linear Predictive) algorithm [14] for the toll-quality coding of speech at 16 kbps known as the ITU G.728. Shortly after this standard was adopted, another CELP based speech coding running at 8 kbps was developed by the University of Sherbrooke [15]. It was toll-quality as well and had a comparable performance to that of 16 kbps LD-CELP. In 1996, it finally became part of the ITU standards and was known as G.729. In the same year, U.S. Department of Defense (DoD) was standardizing a new 2.4 kbps vocoder with communications quality to replace both FS1015 and FS1016. There were seven candidates involved in this standardization and the winner was the Mixed-Excitation Linear Predictive Vocoder (MELP) developed by Texas Instruments [16]. It was reported that its speech quality is even better than FS1016 4.8 kbps vocoder, a vocoder with twice the bit-rate. It is also computationally efficient and robust in difficult background environments such as those encountered in commercial and military communication systems.

Recently, ITU has set a demanding goal of reducing the existing toll-quality rate by a further factor of two, down to the regions of 4 kbps with a quality equivalent to the existing 8 kbps standard (G.729). It is expected that this standardization will be finalized by the end of this century. There are numerous intended applications for this standardization such as visual telephony, multimedia applications in personal communication environments and internet telephony. A worldwide effort is currently underway to prepare for this standardization.

## 1.6 Objectives and Scope of Our Research

The current challenge ahead of us is to search for a narrowband speech coder delivering near-toll-quality speech at a rate of 4 kbps. It is well known that the speech quality of CELP-based algorithms (like G.729) deteriorates rapidly as the bit rate falls below

[2]In this context, a narrowband speech corresponds to a telephone-bandwidth speech which is bandlimited from 200 Hz to 3400 Hz, sampled at 8 kHz and represented with 16 bits uniform PCM (128 kbps).

6 kbps [17]. On the other hand, existing vocoders like MELP, which can provide a high degree of intelligible speech at around 2.4 kbps, cannot provide natural sounding speech by simply adding more bits. Therefore, in seeking for this 4 kbps toll-quality speech coding algorithm, it seems clear that neither coders designed for toll-quality at 8 kbps nor others designed at 2.4 kbps can fill this gap. A new generation of coding scheme is clearly needed.

One of the most promising candidates in the upcoming 4 kbps ITU standardization is the *waveform interpolation* (WI) coder. It was first developed at AT&T in the late 80's [7] and there have been several enhancements since then [18, 19, 20, 21, 22].

The primary objective of this thesis is to propose a WI quantization (bit allocation) scheme running at the neighborhood of 4 kbps, with an attempt to achieve speech quality comparable with G.729 coding at 8 kbps. With the addition of few refinements, a complete WI coder is successfully simulated using C language and its performance is studied. Also, effort is spent to examine the strengths and the weaknesses of the algorithm. A few other WI derivatives will be discussed and compared as well. Finally, we will identify a few problematic areas in the coder, areas that cause the most degradation in the output speech quality and should be improved before the coder is able to reach the toll-quality benchmark at 4 kbps.

This thesis can also be a reference for those who intend to implement a WI coder. For each component in the WI coder, the functional descriptions as well as the relevant mathematical derivations will be provided. Detailed implementation procedures and pitfalls are also documented. In addition, unlike most existing WI references which formulate the WI method for continuous-time signals, this thesis takes a different approach and attempts to represent all formulations in the discrete-time domain. In this way, readers can be exposed more directly to the details required to implement a WI coder.

In the course of this research, we have concentrated mostly on achieving high quality reconstructed speech but we have given little thought to computational complexity, memory requirements, the sensitivities to background acoustic noise and to transmission errors.

## 1.7 Organization of the Thesis

This thesis will be organized as follows. Since understanding the linear prediction concepts is considered as a strong prerequisite for the discussion of the WI method, we first spend Chapter 2 in discussing the basic concepts involved in linear predictive coding, concepts including the linear prediction analysis, bandwidth expansion and pre-emphasis. Chapter 3 introduces the concept and the overall structure of WI algorithm. A brief history and evolution of the algorithm are given. It then presents the implementation of the algorithm, with an emphasis on the analysis-synthesis layer. Each of the algorithmic blocks will be discussed in details and the relevant mathematical derivations will be provided. Various WI derivatives are also examined. In Chapter 4, the implementation of the quantization layer is provided. The resulting speech quality at around 4 kbps is compared with the output of a toll-quality speech coder at 8 kbps — G.729. Our work is summarized and the future research directions are outlined in Chapter 5.

# Chapter 2

# Linear Predictive Speech Coding

In this chapter, we focus on linear predictive coding (LPC) analysis which is an indispensable component in most speech coding algorithms. Specifically, we will examine the short-term LPC whose objective is to remove short-term correlation (redundancy) in a speech signal by employing a time-varying linear prediction (LP) filter. The filter coefficients are known as LP coefficients and the filter output is called an excitation signal or a residual signal. These LP coefficients characterize the spectral envelope of the speech signal governed by the human vocal tract while the residual describes the glottal excitation.

One key advantage of the LPC analysis is that speech is decomposed into two highly independent components, the vocal tract parameters (LP coefficients) and the glottal excitation (LP excitation). These two components have very different quantization requirements. As a result, separate analysis and quantization scheme can be applied to each to enhance coding efficiency. In the past decade, efficient quantization schemes have been developed for the LP coefficients [23]; however, the representation of the excitation signal still remains somewhat problematic. Numerous promising techniques have been proposed in recent years to tackle this problem, one of which is the WI scheme.

We proceed as follows. We first reveal the underlying principles of the short-term LPC analysis and discuss how to calculate the LP coefficients. Next, we introduce a popular representation of the LP coefficients — line spectral frequencies which offer better quantization and interpolation properties. At last, we discuss the concept of bandwidth expansion and pre-emphasis.

## 2.1 Linear Prediction in Speech Coding

Recalled from Section 1.3, the speech production is as a result of the glottal excitation exciting the vocal tract. In linear predictive coding, this process can be modeled as a residual signal exciting a time-varying linear filter, as shown in Fig. 2.1. The filter is

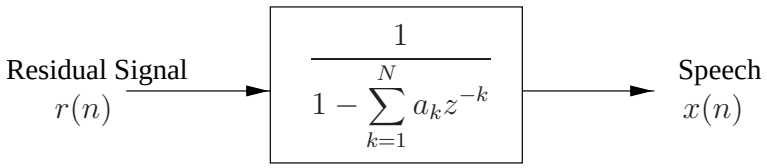


**Fig. 2.1** The LP synthesis filter

all-pole of order $N$. Since the filter synthesizes speech, it is usually referred to as the *LP synthesis filter* and its coefficients $a_1, a_2, \ldots, a_N$ are known as the LP coefficients.

The synthesis filter models the effect of the vocal tract imposed on the glottal excitation, thus the frequency response of the filter corresponds to the spectral envelope (short-term correlations) of the input speech signal. In other words, the center frequencies of the resonances of the filter should closely match the formant locations of the speech signal, as depicted in Fig. 1.2c. As a result, the order $N$ of the filter should be chosen such that there are a pair of poles allocated for each formant. For a speech signal sampled at 8 kHz, it is usually sufficient to set $N = 10$.

The inverse of the synthesis filter is called the LP analysis filter. Its main purpose is to retrieve the $r(n)$ buried in the speech signal as shown in Fig. 2.2.

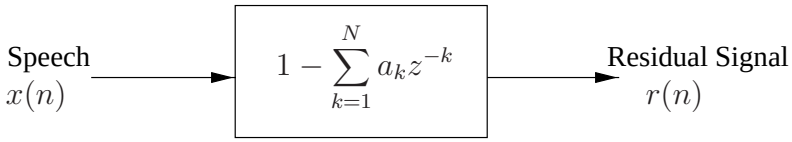


**Fig. 2.2** The LP analysis filter

From either Fig. 2.1 or Fig. 2.2, it is also possible to express the relationship between $x(n)$ and $r(n)$ in a difference equation. We can write

$$\begin{aligned} r(n) &= x(n) - \sum_{k=1}^{N} a_k x(n-k) \\ x(n) &= \sum_{k=1}^{N} a_k x(n-k) + r(n) \end{aligned} \tag{2.1}$$

Since the shape of the vocal tract changes with time, the LP synthesis and analysis filters are both considered time-varying and hence, the coefficients $\{a_k\}$ vary with time. Nevertheless, in a practical coder, these coefficients are typically estimated once per frame only for computational reasons. In the next section, we will concentrate on the estimation procedures for $\{a_k\}$.

## 2.2 Estimation of LP coefficients

There are two common approaches in estimating the LP coefficients, the autocorrelation method and the covariance method. Both methods use the classical least-squares technique and choose $\{a_k\}$ such that the mean energy of the resulting residual signal is minimized.

### 2.2.1 Autocorrelation Method

The speech signal $x(n)$ is first multiplied by an analysis window $w(n)$ of finite length $L_w$ to obtain a windowed speech segment $x_w(n)$.

$$x_w(n) = w(n)x(n) \tag{2.2}$$

The window $w(n)$ is typically chosen to be a Hamming window to minimize the sidelobe energy and is defined to be:

$$w(n) = \begin{cases} 0.54 - 0.46\cos(\dfrac{2\pi n}{L_w - 1}), & \text{for } 0 \le n < L_w \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

Next, we find an expression that corresponds to the energy of the prediction error $E$. From (2.1), we can obtain

$$E = \sum_{n=-\infty}^{\infty} r^2(n) = \sum_{n=-\infty}^{\infty} \left[ x_w(n) - \sum_{k=1}^{N} a_k x_w(n-k) \right]^2 \tag{2.4}$$

The values of $\{a_k\}$ that minimize $E$ are derived by setting

$$\frac{\partial E}{\partial a_k} = 0 \qquad \text{for} \quad k = 1, 2, \ldots, N \tag{2.5}$$

which yields $N$ linear system of equations

$$\sum_{n=-\infty}^{\infty} x_w(n)x_w(n-i) = \sum_{k=1}^{N} a_k \sum_{n=-\infty}^{\infty} x_w(n-i)x_w(n-k) \qquad \text{for} \quad i = 1, 2, \ldots, N \tag{2.6}$$

Defining the autocorrelation function of the windowed signal $x_w(n)$ as

$$R(i) = \sum_{n=-\infty}^{\infty} x(n)x(n-i) = \sum_{n=i}^{L_w-1} x_w(n)x_w(n-i) \tag{2.7}$$

and noting that the autocorrelation function is an even function where $R(n) = R(-n)$, the system of equations in (2.6) can then be expressed in a matrix form:

$$\begin{bmatrix} R(0) & R(1) & \ldots & R(N-1) \\ R(1) & R(0) & \ldots & R(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ R(N-1) & R(N-2) & \ldots & R(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ R(N) \end{bmatrix} \tag{2.8}$$

Since the matrix in (2.8) has a Toeplitz structure, the $\{a_k\}$ coefficients can be solved efficiently by Levinson-Durbin recursion [24]. In addition, the Toeplitz structure can guarantee the poles of the resulting LP synthesis filter to be inside the unit circle and hence, the filter stability is always fulfilled [25].

### 2.2.2 Covariance Method

The covariance method is another way to estimate the $\{a_k\}$ parameters. Although both approaches are similar, they differ in the placement of the analysis window. The covariance method windows the error signal rather than the speech signal. In this case, the energy of the prediction error $E$ becomes

$$E = \sum_{n=-\infty}^{\infty} r^2(n)w(n) \tag{2.9}$$

By solving (2.9) in the same fashion as in the autocorrelation method, one can obtain a system of $N$ linear equations which can be expressed in a matrix form:

$$\begin{bmatrix} \varphi(1,1) & \varphi(1,2) & \dots & \varphi(1,N) \\ \varphi(2,1) & \varphi(2,2) & \dots & \varphi(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi(N,1) & \varphi(N,2) & \dots & \varphi(N,N) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} \varphi(0,1) \\ \varphi(0,2) \\ \vdots \\ \varphi(0,N) \end{bmatrix} \tag{2.10}$$

where $\varphi(i,j)$ is the covariance function for $x(n)$ and is defined as:

$$\varphi(i,j) = \sum_{n=-\infty}^{\infty} x(n-i)x(n-j)w(n) \tag{2.11}$$

Though this matrix in (2.10) does not have the Toeplitz structure, it is symmetric positive definite which implies that the $\{a_k\}$ can be solved in an efficient manner by Cholesky decomposition [24].

The covariance method does not window the input signal, hence, it is advantageous for high resolution spectral estimation applications. However, it does not guarantee the stability of the all-pole LP synthesis filter; the poles of the estimated coefficients may lie outside of the unit circle. For this reason, the covariance method will not be used in our WI implementation.

## 2.3 Interpolation of LP coefficients

As previously mentioned, the LP coefficients $\{a_k\}$ are typically estimated on a frame-wise basis. In order to avoid rapid changes in the coefficients between two successive frames, the coefficients are interpolated for individual subframes so that they evolve smoothly over frames. Otherwise, a substantial amount of frame-to-frame variations in the estimated LP coefficients may lead to undesired transients, roughness and even audible clicks in the resulting speech quality [25].

As is well known, direct interpolation of the LP coefficients $\{a_k\}$ can result in an unstable analysis filter. Therefore, the coefficients are most commonly transformed into another domain, then interpolated and transformed back. One popular domain is known as line spectral frequency (LSF) or equivalently, line spectral pair (LSP). It provides not only the stability of the interpolated LP coefficients, but also easy

spectral manipulations and desirable quantization properties.

The conversion of the LP coefficients $\{a_k\}$ to the LSF domain can be done as follows [26]. We first denote

$$A(z) \equiv 1 - \sum_{k=1}^{N} a_k z^{-k} \tag{2.12}$$

Note that the zeros of $A(z)$ are the poles of the LP synthesis filter or the zeros of the LP analysis filter. These zeros are then mapped onto the unit circle through two z-transforms $P(z)$ and $Q(z)$ of $(N+1)$st order:

$$\begin{aligned} P(z) &= A(z) + z^{-(N+1)} A(z^{-1}) \\ Q(z) &= A(z) - z^{-(N+1)} A(z^{-1}) \end{aligned} \tag{2.13}$$

The zeros of $P(z)$ and $Q(z)$ lying on the unit circle are interlaced. The LSF coefficients are defined to be the angular positions $\{\omega_i\}$ of these zeros between 0 and $\pi$. Precisely, the LSFs can be written to be

$$0 = \omega_0 < \omega_1 < \ldots < \omega_N < \omega_{N+1} = \pi \tag{2.14}$$

The $\omega_0$ and $\omega_{N+1}$ are always 0 and $\pi$ respectively and need not to be coded. Furthermore, the ascending ordering property of the LSFs as indicated in (2.14) ensures the stability of the synthesis filter. This type of simple stability check does not exist for the LP coefficients $\{a_k\}$.

One other important characteristic of the LSF is the localized spectral sensitivity. For the LP coefficients, a small error in one coefficient might dramatically alter the spectral shape and even lead to an unstable synthesis filter. Whereas, if one LSF is distorted, the spectral alteration tends to occur only in a neighborhood near the LSF.

The zeros of the polynomials in (2.13) can be found by the method described in [27] where the Chebyshev polynomials are used to find the roots in the cosine domain.

## 2.4 Bandwidth Expansion

Occasionally, the LP analysis generates a synthesis filter with sharp spectral formant peaks. This implies that the poles of the filter are too close to the unit circle and

hence, the filter is marginally stable. Such marginal stability in the LP filters can increase the chances of getting cross-overs in LSF quantization which may in turn cause occasional chirps in quantized speech. One solution to this problem is to employ *bandwidth expansion* to expand the bandwidths in the frequency response of the filter.

In the process of bandwidth expansion, each LP coefficient $a_k$ is replaced by $\gamma^k a_k$, where $k = 1, 2, \ldots, N$. Such a multiplication moves all the filter poles away from the unit circle and toward the origin by a factor of $\gamma$. It results in smoothed peaks and broadened bandwidths in the frequency response of the analysis filter and hence, the filter becomes more stable. Also, it reduces the quantization cross-overs of closely spaced LSFs.

The $\gamma$, also called the *bandwidth expansion factor*, controls how much the poles move inward by. The typical values for $\gamma$ are between 0.988 and 0.996 which correspond to 10 to 30 Hz bandwidth expansion [25].

## 2.5 Pre-Emphasis

In the conventional A-to-D process, an analog speech waveform is lowpass filtered prior to sampling. Such operation prevents spectral aliasing in the digitized speech but at the same time, reduces the energy of the high frequency components. This is rather undesirable in the LP analysis since a relatively weak energy at high frequencies may cause the autocorrelation matrix in (2.8) to become ill-conditioned and subsequently, affect the numerical precision of the LP coefficients [28]. To overcome this problem, the speech energy is often boosted as a function of the frequency prior to computing the LP coefficients. Specifically, this can be accomplished by passing the speech signal $x(n)$ through the filter

$$H(z) = 1 - \alpha z^{-1} \tag{2.15}$$

where $\alpha$ determines the cut-off frequency of the single-zero filter. In this way, the relative energy of the high-frequency spectrum can be increased. This process is known as *pre-emphasis* and the $\alpha$ in $H(z)$ is called the *pre-emphasis factor* which is used to control the degree of pre-emphasis. The typical value for $\alpha$ is around 0.1 [6]. To undo the pre-emphasis effect, a de-emphasis filter defined to be the inverse of $H(z)$ can be employed.

# Chapter 3

# Waveform Interpolation

## 3.1 Background and Principles of WI Coding

It was the perceptual importance of the periodicity in voiced speech that originally motivated the development of the waveform interpolation coding technique. It was first introduced by W. B. Kleijn [7] and the first version was called *Prototype Waveform Interpolation* (PWI). PWI encoded voiced segments only and therefore, it was used in combination with other schemes such as CELP for coding unvoiced segments.

PWI exploits the fact that pitch-cycle waveforms in a voiced segment evolve slowly with time. This slow evolution of the waveforms suggests that we do not have to transmit every pitch-cycle to the decoder; instead, we could transmit them at regular intervals. At the decoder, the non-transmitted pitch-cycle waveforms could then be derived by means of interpolation. In this way, the degree of voiced speech periodicity could be well controlled and consequently, very high quality reconstructed voiced speech could be obtained [9]. In PWI, the pitch-cycles that are selected to be transmitted are referred to as the *Prototype Waveforms*.

Although PWI works remarkably well with voiced segments, it has one inherent flaw — it is not applicable to unvoiced speech. In other words, it always has to work with another method of speech coding to handle unvoiced segments. Thus, the switching between coders becomes inevitable and significantly reduces the robustness of the coder. In 1994, PWI was further refined to become WI which is capable of encoding both voiced and unvoiced speech [29, 18]. Similar to the principles of PWI, WI represents a speech signal with a sequence of evolving waveforms. For

voiced speech, these waveforms are simply pitch-cycles. And for unvoiced speech and background noise, the waveforms are of varying lengths and contain mostly noise-like signals. Since the evolving waveforms are not limited to pitch-cycles anymore, it is not appropriate to use the terms pitch-cycle or prototype waveform to describe the evolving waveform. Instead, the term *Characteristic Waveform* is adopted, which will be abbreviated to CW from here on.

A key difference between WI and PWI is that the evolving waveforms in WI are being sampled at a much higher rate. However, an increase in waveform sampling rate comes at the expense of an increase in bit rate. To counter this problem, WI decomposes the CW into a smoothly evolving waveform (SEW) and a rapidly evolving waveform (REW). The SEW represents the quasi-periodic component of the speech signal while the REW represents the remaining non-periodic and noise components in the signal. Since the two waveforms have very different perceptual requirements, they can be quantized separately to enhance coding efficiency.

Before discussing any details or implementation of WI, a high-level description of the coder is given in the next section.

## 3.2 Overview of the WI Coder

Figure 3.1 presents a high-level schematic diagram[1] of the WI coder. It can be structurally divided into two layers: the *analysis-synthesis layer* and the *quantization layer*. In the former layer, the analysis block (processor **100**) first performs a LPC analysis on the incoming speech signal and obtains the corresponding residual signal. Then the pitch is estimated and the residual is decomposed into a series of CWs. These CWs are subsequently aligned and normalized in power so they can accurately represent a two-dimensional surface illustrating the evolution of the waveforms. The synthesis stage (processor **200**) does the reverse of the analysis side. The residual signal is reconstructed from the CWs and sent to a LP synthesis filter where the speech signal is finally reconstructed.

[1]For the purpose of clarity, each functional block (which will be referred to as a *processor* hereafter) in the WI schematic diagram is identified by a three-digit number. Each digit in the number corresponds to one level of embedding. For example, a processor labeled as **134** indicates that the processor is embedded inside another processor called **130**. And the processor **130** is in turn embedded inside processor **100**. Therefore, if a processor is labeled as **240**, it means that it has two levels of embedding where processor **200** contains processor **240**. This numbering convention will be adopted by all the subsequent WI schematic diagrams presented in this thesis.

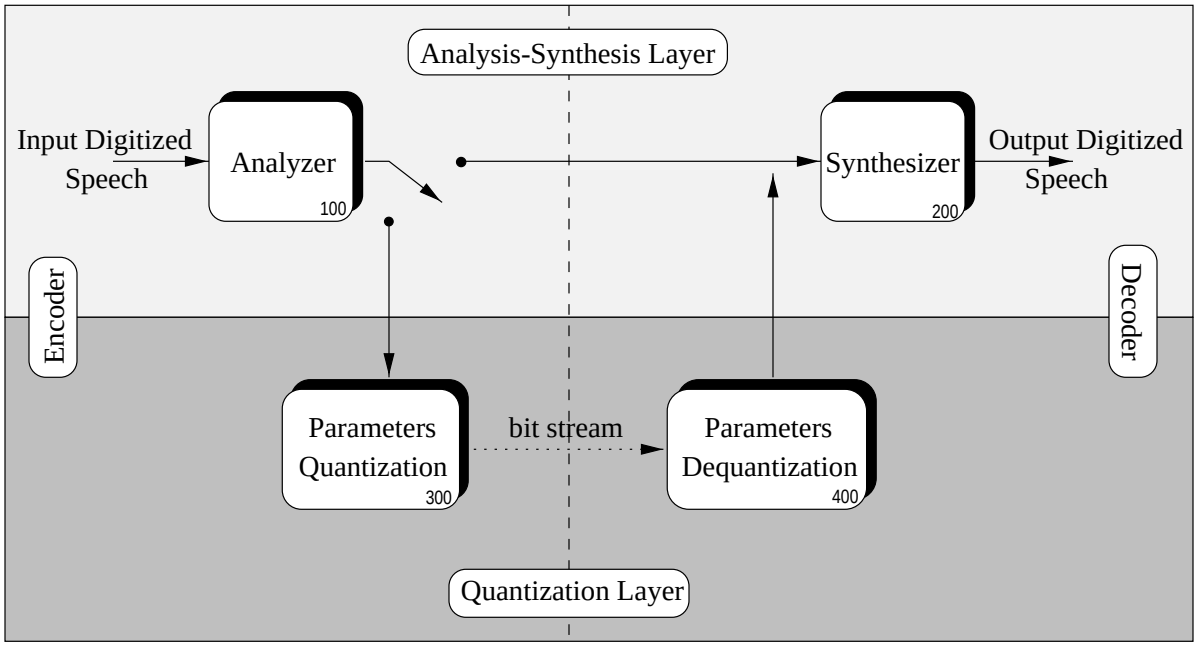


**Fig. 3.1** A block diagram of the WI speech coding system. The switch enables the coder to bypass the quantization layer and allows us to measure the performance of the analysis-synthesis layer. The schematic diagrams for processor **100** and **200** can be found in Figs. 3.3 and 3.14 respectively. Further, the schematics for processors **300** and **400** are shown in Fig. 4.1.

Processor **300** in the quantization layer carries out the SEW-REW decomposition and the parameter quantization. Processor **400** at the receiver dequantizes the parameters and reconstructs the CWs from the transmitted SEWs and REWs.

In this chapter, we will discuss the analysis-synthesis layer which encompasses most of the key WI elements including pitch extraction, CW extraction, CW alignment and CW interpolation. Our discussion is based largely on the seminal work on WI by Kleijn [30].

For each processor in the layer, implementation details along with relevant mathematical derivations will be given. Schematic diagrams of selected processors will be shown to facilitate the discussion. We will also provide the performance results of the analysis-synthesis layer and discuss how WI can be used to time-scale a reconstructed speech signal. Processors **300** and **400** in the quantization layer will be examined in the next chapter.

## 3.3 Representation of Characteristic Waveform

Before we dive into the details of any processors, we first begin by choosing an appropriate mathematical representation for the CWs. As we will learn later, a majority of the computations in WI are associated with the CWs, it is therefore crucial to have an appropriate CW representation so as to reduce the complexity of the coder.

The CWs are ultimately used to construct a two-dimensional surface describing the waveform evolution. Thus, the CW representation that we are seeking must have the ability to represent a two-dimensional signal.

A good start is to consider a single, one-dimensional CW. The CW is a discrete-time real sequence, one pitch period long. By denoting the CW as $s(m)$ and the pitch[2] as $P$, we can write:

$$s(m) \in \mathbf{R} \qquad m = 0, 1, \ldots, P-1 \tag{3.1}$$

A portion of the processing in WI is in the frequency domain. This implies that a frequency-domain representation would be favoured. Here, we have chosen the Discrete-Time Fourier Series (DTFS) representation where $s(m)$ can be expressed as:

$$s(m) = \sum_{k=0}^{\lfloor P/2 \rfloor} \left[ A_k \cos\left(\frac{2\pi km}{P}\right) + B_k \sin\left(\frac{2\pi km}{P}\right) \right] \qquad 0 \le m < P \tag{3.2}$$

where $\{A_k\}$ and $\{B_k\}$ are the DTFS coefficients and can be calculated using a set of transform equations. Specifically, when P is *even*:

$$\left.\begin{aligned} A_k &= \frac{2}{P} \sum_{m=0}^{P-1} \left[ s(m) \cos\left(\frac{2\pi km}{P}\right) \right] \\ B_k &= \frac{2}{P} \sum_{m=0}^{P-1} \left[ s(m) \sin\left(\frac{2\pi km}{P}\right) \right] \end{aligned}\right\} \quad \text{for} \quad k = 1, 2, \ldots, P/2 - 1$$

$$\left.\begin{aligned} A_k &= \frac{1}{P} \sum_{m=0}^{P-1} \left[ s(m) \cos\left(\frac{2\pi km}{P}\right) \right] \\ B_k &= \frac{1}{P} \sum_{m=0}^{P-1} \left[ s(m) \sin\left(\frac{2\pi km}{P}\right) \right] \end{aligned}\right\} \quad \text{for} \quad k = 0 \text{ and } P/2 \tag{3.3}$$

[2]For this thesis, the terms "pitch" and "pitch period" will be interchanged.

When P is *odd*:

$$\left.\begin{aligned} A_k &= \frac{2}{P}\sum_{m=0}^{P-1}\left[s(m)\cos\left(\frac{2\pi km}{P}\right)\right] \\ B_k &= \frac{2}{P}\sum_{m=0}^{P-1}\left[s(m)\sin\left(\frac{2\pi km}{P}\right)\right] \end{aligned}\right\} \quad \text{for} \quad k = 1, 2, \ldots, (P-1)/2$$

$$\left.\begin{aligned} A_k &= \frac{1}{P}\sum_{m=0}^{P-1}\left[s(m)\cos\left(\frac{2\pi km}{P}\right)\right] \\ B_k &= \frac{1}{P}\sum_{m=0}^{P-1}\left[s(m)\sin\left(\frac{2\pi km}{P}\right)\right] \end{aligned}\right\} \quad \text{for} \quad k = 0 \tag{3.4}$$

The shape of an individual CW can now be described by a set of DTFS coefficients $\{A_k, B_k\}$. Note that the index $m$ in (3.2) does not have to be an integer; it can be any real value within the range $0 \leq m < P$. In other words, values that fall between discrete time instants (e.g. $s(1.4)$) can now be calculated readily by (3.2) [3].

Having acquired the representation for a single CW, we are now ready to construct the two-dimensional representation for a sequence of CWs. In fact, this representation can be obtained by simply adding one modification to (3.2). That is, attach a discrete time index $n$ to all the parameters in (3.2) that may vary with time. These parameters are $\{A_k\}$, $\{B_k\}$ and $P$. Equation 3.2 can therefore be rewritten as:

$$s(n,m) = \sum_{k=1}^{\lfloor P(n)/2 \rfloor}\left[A_k(n)\cos\left(\frac{2\pi km}{P(n)}\right) + B_k(n)\sin\left(\frac{2\pi km}{P(n)}\right)\right] \qquad 0 \leq m < P(n) \tag{3.5}$$

where coefficients $\{A_k(n)\}$ and $\{B_k(n)\}$ are now time-varying and so is the pitch value $P(n)$. Note that we are ignoring the coefficients $A_0$ and $B_0$ in the equation (the index $k$ starts from $k = 1$ instead of $k = 0$). This is because the $B_0$ in (3.3) and (3.4) is a redundant coefficient ($\sin(0) = 0$). On the other hand, the $A_0$ represents the DC component of the signal and has no perceptual relevance. Consequently, both coefficients can be ignored.

Equation 3.5 is now a two-dimensional signal representation where $m$ and $n$ are the running variables. Individual CWs are displayed along the $m$ axis and the shape of the CWs evolves over time along the $n$ axis.

[3] This operation is in fact equivalent to bandlimited interpolation of a periodic sequence.

However, the length of the CW in (3.5) is dependent on the time-varying pitch $P(n)$; the CWs at different times may be of different length. It is generally more convenient to normalize all the CWs to a common length. This normalization can be accomplished by substituting

$$\phi = \phi(m) = \frac{2\pi m}{P(n)} \tag{3.6}$$

in (3.5) and we can obtain

$$s(n,\phi) = \sum_{k=1}^{\lfloor P(n)/2 \rfloor} [A_k(n)\cos(k\phi) + B_k(n)\sin(k\phi)] \qquad 0 \le \phi(\cdot) < 2\pi \tag{3.7}$$

In this way, all the CWs have the same length of $2\pi$. Figure 3.2 gives a pictorial description of the normalization and an example of a two-dimensional surface represented by (3.7).

*Remarks on the DTFS representation*

- At first sight, $B_{\frac{P}{2}}$ in (3.3) seems to be a redundant coefficient since $\sin(m\pi) = 0$ for all integers $m$. In fact, this is not entirely true. As we will see in Section 3.4.5, this particular coefficient would no longer be zero when the signal undergoes a fractional time shift in (3.20) in the alignment processor.

- Generally speaking, representing a signal by DTFS implies that the signal is being periodically extended. Likewise, representing a CW by DTFS means that the CW is being derived from a periodic signal.

- Other CW representations are also possible [31]. Time domain representations may actually reduce the coder complexity to a certain extent by avoiding the DTFS forward and inverse transforms. Nevertheless, they could be problematic when dealing with frequency-dependent processing [25].

## 3.4 The Analysis Stage

We will first concentrate on the analysis processor **100**. As stated previously, the primary goal of this processor is to decompose a speech signal into a series of evolving

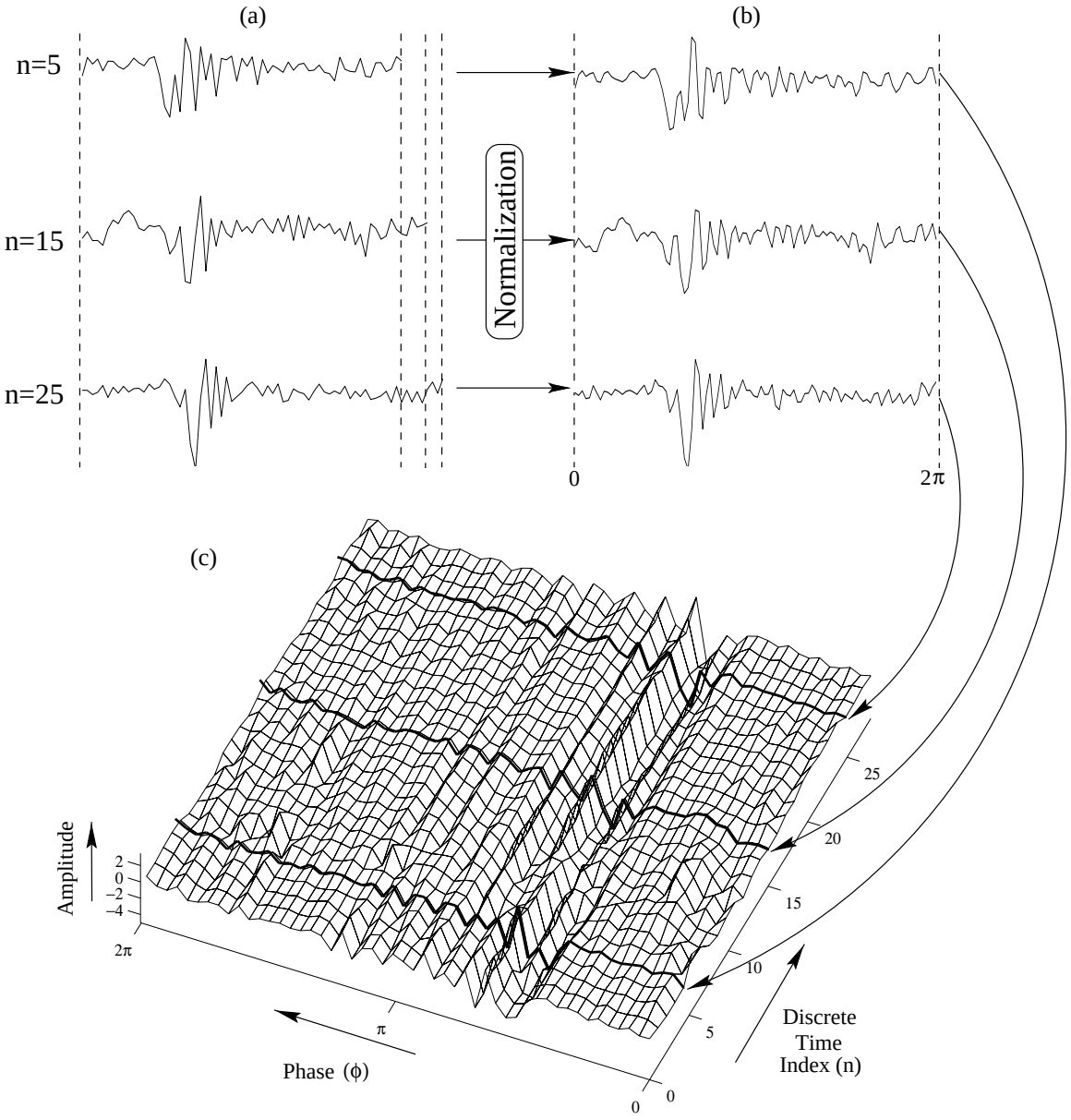


**Fig. 3.2** An example of a characteristic waveform surface. (a) CWs (pre-aligned) sampled at $n = 5, 15, 25$. Note that they are of different lengths. (b) The time normalized version of the CWs. (c) Formation of a two-dimensional evolving surface. The surface is constructed from a series of CWs including the ones shown in (b). The individual CWs are displayed along the $\phi$ axis and the evolution of the CW shape over time can be seen along the $n$ axis.

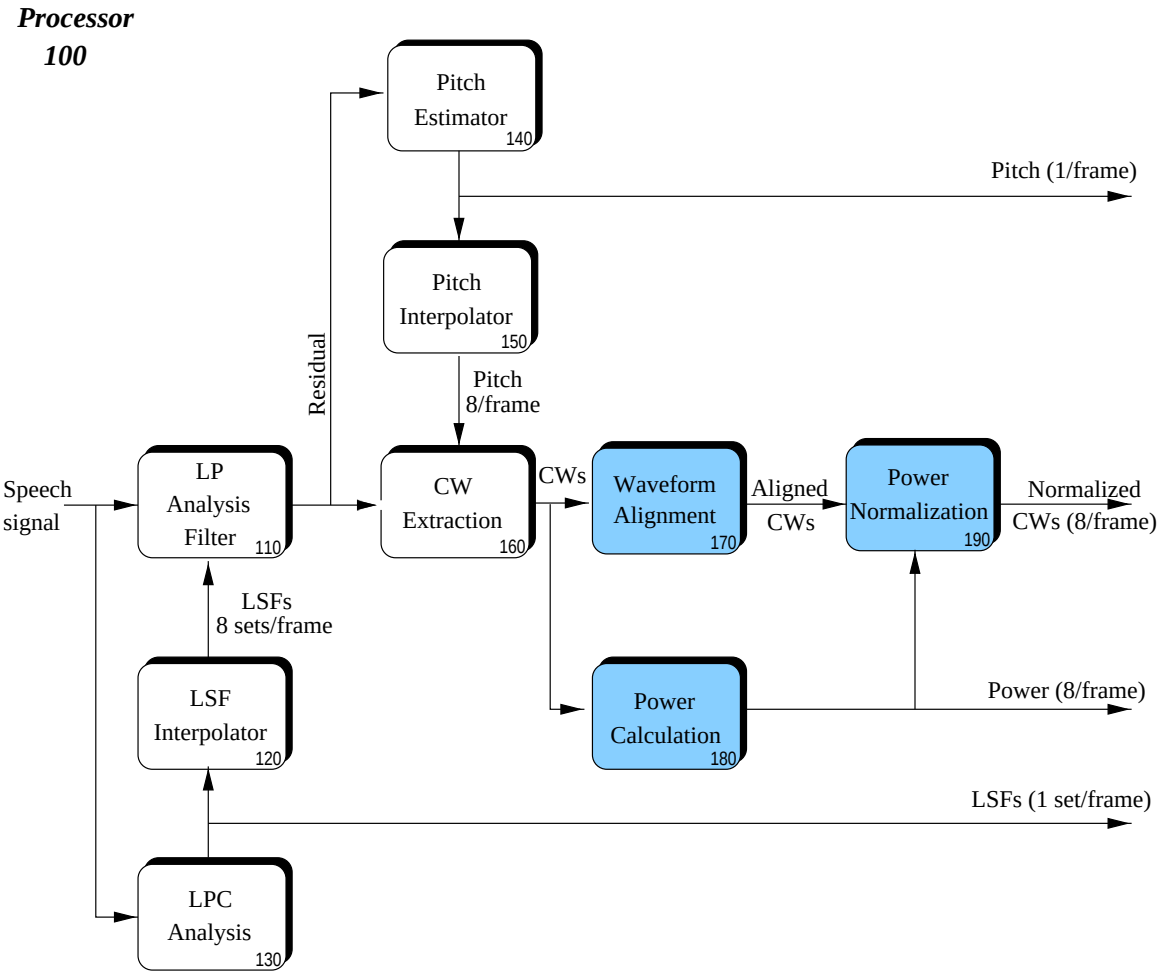


**Fig. 3.3** A block diagram of the WI analysis block (processor **100**). The shaded processors are executed on a per subframe basis while the non-shaded ones are processed once per frame.

CWs (a two-dimensional surface) as well as other orthogonal parameters including the LSFs, the power and the pitch. Figure 3.3 shows all the processors that comprise this analysis layer.

This thesis assumes that the input and the reconstructed speech are in a sampled, digital format and the sampling rate is 8000 kHz. Our frame size $L_f$ is 160 samples (20 ms) and the subframe length $L_{sf}$ is 20 samples [4]. Note that some processors are

[4] As we will learn in Section 3.4.4, the value of $L_{sf}$ in the coder is actually governed by the CW extraction rate, $R_{extr}$. Specifically, the distance between each two successive extraction points defines the length of a subframe. In our implementation, $R_{extr}$ is 8/frame. This implies that the length of the subframe is $L_{sf} = L_f \div R_{extr} = 20$ samples.

executed once per frame while others are executed once per subframe. The following subsections will discuss each of the processors shown in the Fig. 3.3

### 3.4.1 LP Analysis

Each incoming speech frame is first sent to processor **130** where a 10th order LP analysis is performed to derive a set of LP coefficients $\{a_k\}$. Specifically, the input speech is first pre-emphasized using $\alpha = 0.1$ in (2.15). This operation aims to compensate the loss of high-frequency energy due to the lowpass filtering during the A-to-D conversion. The pre-emphasized speech is then windowed using a Hamming window defined in (2.3) with $L_w = 240$. The center of the window lies at the right boundary of the current frame. In other words, the window covers 120 samples in the current frame and 120 samples in the future frame. These 120 future samples translate into an algorithmic delay of 15 ms. The autocorrelation method is performed on the windowed speech to generate the filter coefficients $\{a_k\}$. These $\{a_k\}$ are bandwidth expanded using $\gamma = 0.98829$ or equivalently 30 Hz expansion. The resulting coefficients are converted to the LSF domain and shipped to processor **120**. All of the above operations are repeated once per frame, so the update rate for the LP coefficients is 50 Hz in our implementation. This update rate can be justified by the rate of the vocal tract articulators [6].

In processor **120**, the LSFs between two successive frames are linearly interpolated into a subframe level to ensure a smooth transition is achieved. Each of the interpolated set of LSFs is then converted back into their $\{a_k\}$ before they are used by the LP analysis filter in processor **110** to compute the LP residual signal for the current frame. In addition, the last interpolated LP coefficients $\{a_k\}$ in the current frame will be used to compute 120 residual samples in the future frame. These future residual samples will be needed by the pitch estimator (processor **140**) and the CW extractor (processor **160**). Note that the transfer function of the LP analysis filter is the same as the one shown in Fig. 2.2.

### 3.4.2 Pitch Estimation

The computed residual samples (including the 120 look-ahead samples) are fed into processor **140** which is the pitch estimator. In the WI paradigm, the accuracy of this pitch estimator is very crucial to the performance of the coder. In particular, both

the extraction process (processor **160**) at the encoder and the interpolation process (processors **230** and **250**) at the decoder rely heavily on the pitch estimate.

There are a variety of pitch estimation procedures available. Some of them are based on locating "pitch markers" (the dominant spike in each pitch cycle of the residual signal) whereas some other ones are based on finding the delay which gives the maximum autocorrelation or prediction gain for a frame of samples. In our WI implementation, we adopt the pitch estimation algorithm from EVRC (Enhanced Variable Rate Codec) [32] which belongs to the latter category. A brief description of the algorithm is given as follows.

The pitch estimator provides an estimation once per frame. For each frame of the data, the estimator carries out the calculations independently over two overlapping windows. The first window comprises the entire current frame while the second window comprises the second half of the current frame and the first-half of the look-ahead frame. These look-ahead samples were previously computed in processor **110** using the last interpolated $\{a_k\}$ in the current frame. Thus, the pitch estimator introduces no additional delay to the coder.

Next, the computations of prediction gain over all desired delay values are carried out separately in each window. This prediction gain, denoted as $\beta$, is defined as:

$$\beta = \max\left\{0, \min\left\{\frac{\sum_{i=0}^{L_f-d-1} r(i)r(i+d)}{\sqrt{\sum_{j=0}^{L_f-d-1} r^2(j) \sum_{k=0}^{L_f-d-1} r^2(k+d)}}, 1.0\right\}\right\} \qquad P_{\min} \le d \le P_{\max} \tag{3.8}$$

in which $d$ is an integer value representing the delay and $r(\cdot)$ is the residual signal. The denominator acts as a normalization factor and the max and min functions keep $\beta$ bounded within $[0, 1]$. If the delay value $d$ corresponds to the true pitch period of the signal or an integer multiplier of that, the corresponding $\beta$ value would be close to 1.0. In contrast, $\beta$ tends to be considerably less than unity for all delays if the signal displays no periodic character (unvoiced speech). Therefore, in order to find the true pitch, we search for the delay that yields the maximum $\beta$. This delay will be referred to as the *optimal delay*.

After finding the optimal delay for each window, we can use the following thresh-

olds and logic to combine the optimal delays from the two windows to obtain a more reliable delay estimate for the current frame. If we let ($d_0$, $\beta_0$) be the optimal delay and the corresponding gain value found for the first window and ($d_1$, $\beta_1$) be the ones found for the second window, the final delay estimate $d_{opt}$ is obtained by (assumes 8000 Hz for the sampling rate):

```
if ( β0 > β1 + 0.4 )
{
        if ( |d0 - d1| > 15 )
                dopt = d0
        else
                dopt = ⌈ (d0 + d1) / 2.0 ⌉
}
else
        dopt = d1
```

$\beta_0$ and $\beta_1$ function as confidence values indicating how reliable the corresponding pitch estimates are ($d_0$ and $d_1$). For example, if $\beta_0$ is much larger than $\beta_1$, it indicates that the $d_0$ estimate is more reliable than $d_1$. Using this reliability information, the first *if* statement is constructed so as to give preferences to the look-ahead frame estimates ($d_1$, $\beta_1$). This will enable the coder to perform better at voicing onsets.

Note that the values of $d$ in (3.8) are integers. Hence, the pitch estimator described by the equation gives only integer pitch values. Indeed, integer pitch (with a resolution of one sample for a 8 kHz sampling rate) values are sufficient for the current WI implementation [30].

Equation 3.8 works with two parameters $P_{\min}$ and $P_{\max}$. They are the minimum and maximum allowable pitch values. In our implementation, they are set to 20 and 120 respectively. We could have set them to a larger range such as 20 and 147 since we are spending 7 bits anyways to quantize the pitch ($147 - 20 + 1 = 128$). However, a wider range of pitch values may lead to more pitch doubling/tripling occurrences. More about the pitch doubling/tripling will be written in Section 3.4.3 and the pitch coding issue will be discussed in Chapter 4.

*Remarks on the pitch estimator*

- The pitch estimator always supplies a pitch period no matter if the signal is periodic or not. In an unvoiced speech where the $\beta$ value is comparatively

low, the pitch period varies. In this case, the pitch value should be set to the minimum value, $P_{\min}$ in order to reduce the computational load of the coder. As we will see later in Section 3.4.4, this pitch estimate will be used to decide the length of extracted CWs in processor **160**. Shorter CWs allow WI to do less computations, especially in the DTFS transform and the alignment process.

- Calculations of the prediction gain over the entire delay range (from $P_{\min}$ to $P_{\max}$) is computationally burdensome. To save some computations, one can first get an initial pitch estimate on a decimated residual buffer. This estimate together with its neighborhood values can then be refined further in the non-decimated signal to obtain the final pitch estimate. In this way, the size of the delay range is significantly reduced which results in a lower computational cost.

### 3.4.3 Pitch Interpolation

As stated in Section 3.4.2, the pitch is estimated only once per frame. However, WI requires the pitch period at every extraction point in processor **160** to perform CW extraction. To solve this problem while maintaining the same level of computational complexity, we use a pitch interpolator (processor **150**) to calculate the intermediate pitch values. Although there are many existing pitch interpolation algorithms available [33], the conventional linear interpolation technique is sufficient for WI.

If we define $P(n_1)$ and $P(n_2)$ to be the pitch values at the boundaries of the present frame where $n_1 < n_2$, then the pitch can be linearly interpolated by:

$$P(n) = \frac{(n_2 - n)P(n_1) + (n - n_1)P(n_2)}{n_2 - n_1} \qquad n_1 \leq n \leq n_2 \tag{3.9}$$

where $n_2 - n_1 = L_f = 160$ samples in our implementation.

Nevertheless, in natural speech especially at the beginning and end of a voiced segment, the pitch value occasionally doubles/triples/halves [9]. In addition, pitch estimators suffer from frequent errors in which the estimated pitch is an integer multiple of the actual pitch. If no special attention is paid and the linear interpolation is performed across these changes, the resulting reconstructed speech would result in audible chirps [30].

To correct this problem, we can interpolate the pitch values in the following way.

For the case where $P(n_1) < P(n_2)$:

$$P(n) = \begin{cases} \dfrac{C(n_2 - n)P(n_1) + (n - n_1)P(n_2)}{C(n_2 - n_1)} & \text{for } n_1 \leq n < \frac{n_1+n_2}{2} \ , \\ \dfrac{C(n_2 - n)P(n_1) + (n - n_1)P(n_2)}{n_2 - n_1} & \text{for } \frac{n_1+n_2}{2} \leq n < n_2 \ . \end{cases} \tag{3.10}$$

where $C$ is defined to be the ratio of $P(n_2)$ to $P(n_1)$ rounded to the nearest integer. For $P(n_1) > P(n_2)$:

$$P(n) = \begin{cases} \dfrac{(n_2 - n)P(n_1) + C(n - n_1)P(n_2)}{n_2 - n_1} & \text{for } n_1 \leq n < \frac{n_1+n_2}{2} \ , \\ \dfrac{(n_2 - n)P(n_1) + C(n - n_1)P(n_2)}{C(n_2 - n_1)} & \text{for } \frac{n_1+n_2}{2} \leq n < n_2 \ . \end{cases} \tag{3.11}$$

where $C$ is the nearest integer ratio of $P(n_1)$ to $P(n_2)$.

The factor $C$ can be considered as an indicator showing whether the pitch has (sub)multiplied. When $C$ is unity, it indicates that there is no pitch doubling or tripling and the above two formulations become the linear interpolation equation (3.9). On the other hand, when $C$ is greater than one, it implies that the pitch has (sub)multiplied and the interpolation described by (3.10, 3.11) is performed in such a way that the pitch changes discontinuously at the midpoint by the factor C. Figure 3.4 illustrates an example of such interpolation in the case of pitch doubling and halving.

In addition, Figs. 3.5a and 3.5b show a speech and a residual segment respectively

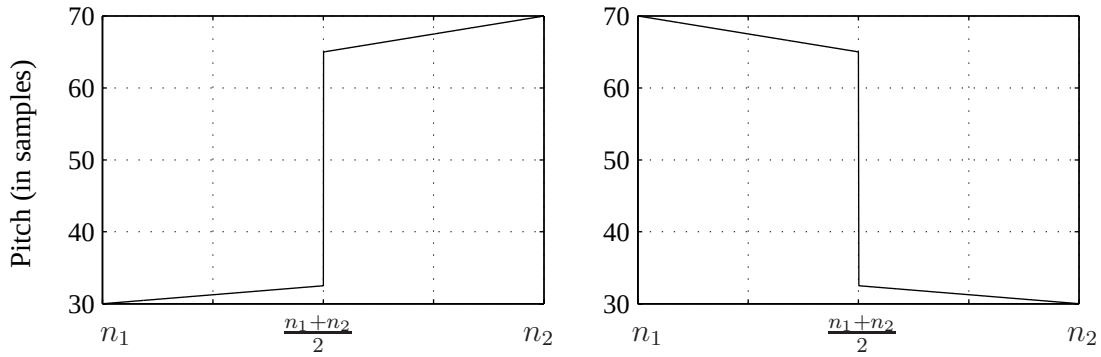


**Fig. 3.4** Interpolation of pitch in the case of pitch doubling. The left diagram interpolates between 30 and 70 using (3.10) and the right diagram does the vice-versa using (3.11).

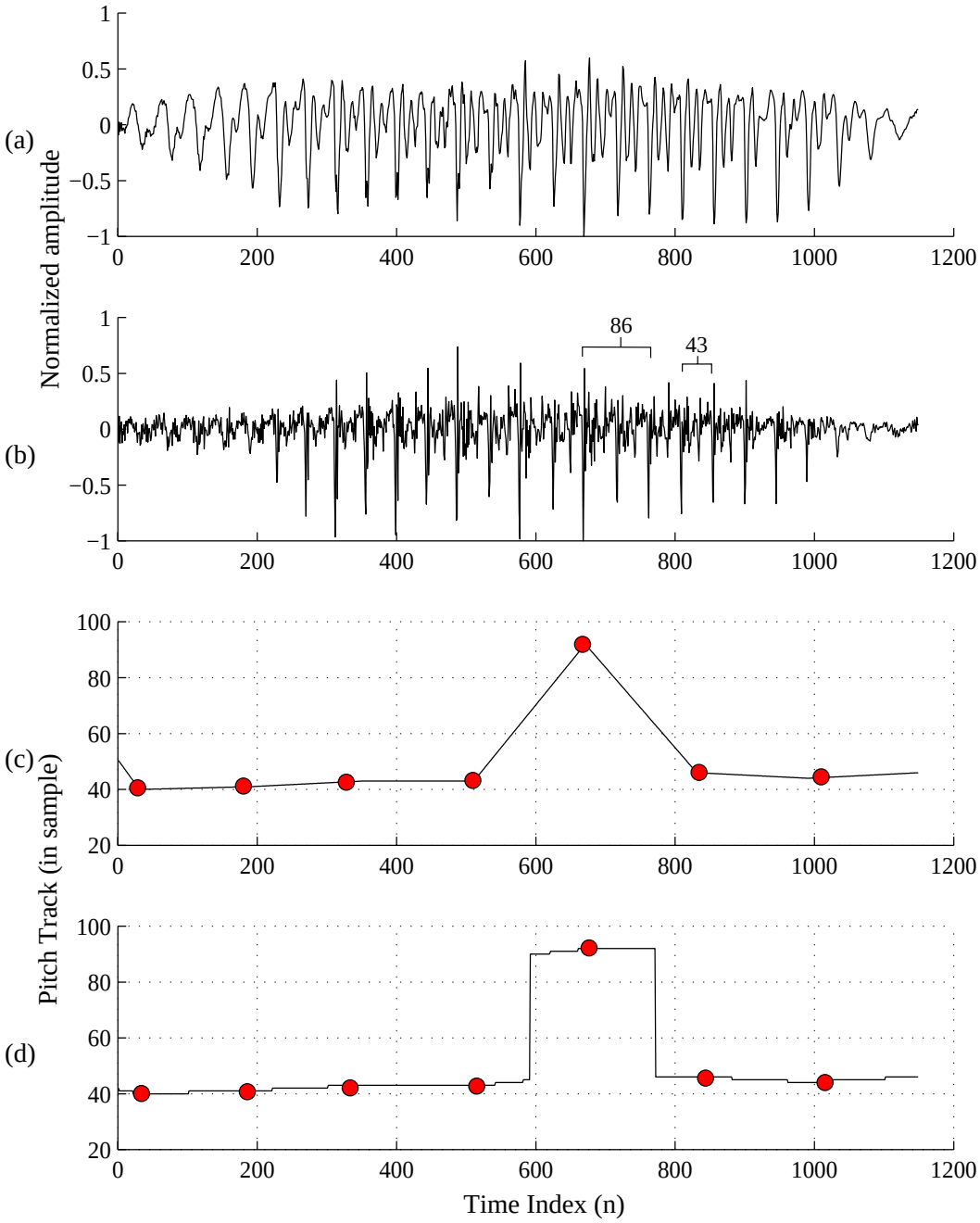


**Fig. 3.5** (a) and (b) show a pitch-doubling speech segment and its corresponding residual. (c) shows the interpolated pitch track resulting from the linear interpolation using (3.9). The linearly increasing and decreasing pitch track (from 46 to 83 and vice-versa) would cause an audible chirp in WI reconstructed speech. The dots are the original pitch estimates from processor **140**. (d) shows the resulting pitch track when using (3.10) and (3.11) for interpolation.

in which a pitch doubling actually occurs. The corresponding interpolated pitch tracks using (3.9) and (3.10, 3.11) are shown in Figs. 3.5c and 3.5d respectively for comparisons. As mentioned in Section 3.4.2, fractional pitch values are not necessary for WI. Therefore, the interpolated pitch values resulting from either (3.9), (3.10) or (3.11) should all be rounded to the nearest integers.

### 3.4.4 CW Extraction

After the pitch is estimated and interpolated, CWs are then extracted in processor **160**. This extraction process is performed on a framewise basis and the frequency of the extraction is determined by the extraction rate, denoted as $R_{extr}$. In our WI simulation, we have found that an extraction rate of 8/frame (400 Hz) is sufficient for the analysis-synthesis to produce excellent speech quality.

The extraction process begins by first partitioning the current frame into eight intervals of equal length. The endpoint of each time interval is marked by an *extraction point*, as illustrated in Fig. 3.6a. As a result, we would have 20 samples between adjacent extraction points. This interval defines the length of the subframe $L_{sf}$ in the coder. In other words, an extraction point is always situated at the endpoint of a subframe.

At each extraction point, we simply take the interpolated pitch period supplied from processor **150** and form an *extraction window* of that length. The extraction window is centered at the extraction point and the residual segment bounded within the window becomes the extracted CW. Thus, the extracted CW always has a length of one pitch period.

As mentioned in Section 3.3, the CW is periodically extended during the conversion to DTFS domain. Therefore, if no particular attention is paid to the boundaries of the CW during the extraction process, it can lead to large discontinuities in the periodic CW (where the left side of the CW meets the right side). Such discontinuities would in turn cause audible distortions in the output speech. To overcome this problem, we *relax* the positions of each extraction point to a certain extent. That is, the extraction point is allowed to move slightly from its original position by a value of $\epsilon$. The value of $\epsilon$ that yields the lowest signal energy around the extraction window boundaries is then chosen. Figure 3.6 shows an example of an unconstrained extraction point. In our implementation, this $\epsilon$ is allowed to vary between $-\epsilon_{\max}$ and

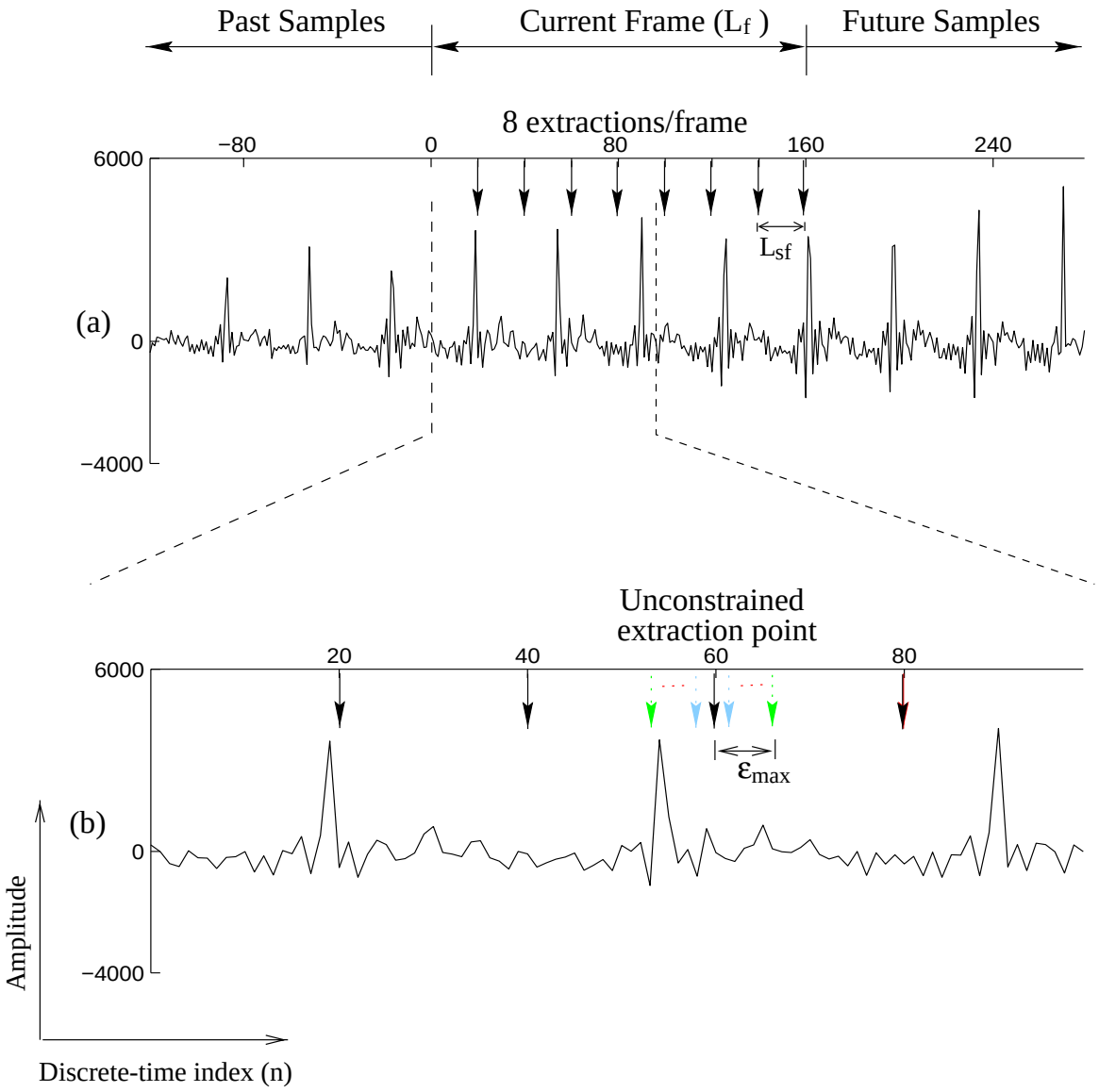


**Fig. 3.6** An example of an unconstrained extraction point. The diagram (a) shows the original locations of the eight equally spaced extraction points for a frame of residual signal. Each extraction point is allowed to move slightly in such a way that the extraction window boundaries are located in regions of relatively low power. The second diagram demonstrates such relaxation for the extraction point at $n = 60$.

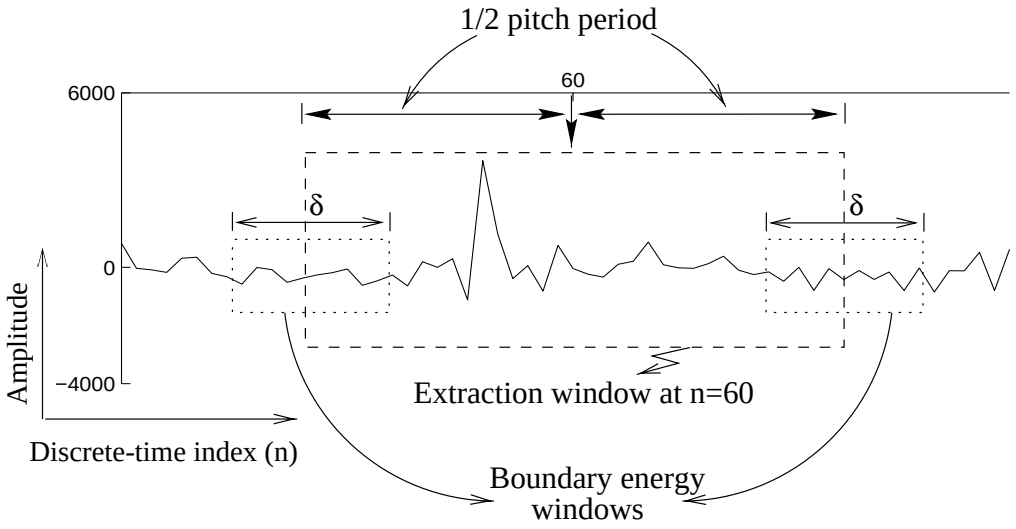


**Fig. 3.7** The extraction window corresponding to $n = 60$ in Fig. 3.6 is shown. Its two boundary windows are illustrated as well. Note that the extraction window is of a pitch-period long and the boundary energy windows each have a length of $\epsilon$.

$\epsilon_{\max}$. Informal experiments confirm that $\epsilon_{\max}$ can be as high as 16 samples without affecting perceptual quality of the reconstructed speech.

In order to efficiently calculate the boundary energy, we create another window called *boundary energy window* which centers at either side of the extraction window, as shown in Fig. 3.7. The boundary energy for a particular extraction point is the sum of the energies of the segments bounded by the left and right boundary windows. Here, we designate the length of the boundary energy window as $\delta$ and it is sufficient to set this value to 10 samples.

In addition to performing the extraction, processor **160** also transforms all the CWs into the DTFS domain using the transform equations (3.3) and (3.4). It should be recalled from Section 3.3 that the coefficients $A_0$ and $B_0$ can be ignored.

*Remarks on the extraction process*

- The extraction windows may sometimes go over the frame boundaries, so a number of past and future samples are required to perform the entire extraction process for the current frame. Since the longest possible length of a CW is $P_{max}$, the number of past samples required is at least $P_{max} \div 2 = 60$. The same applies

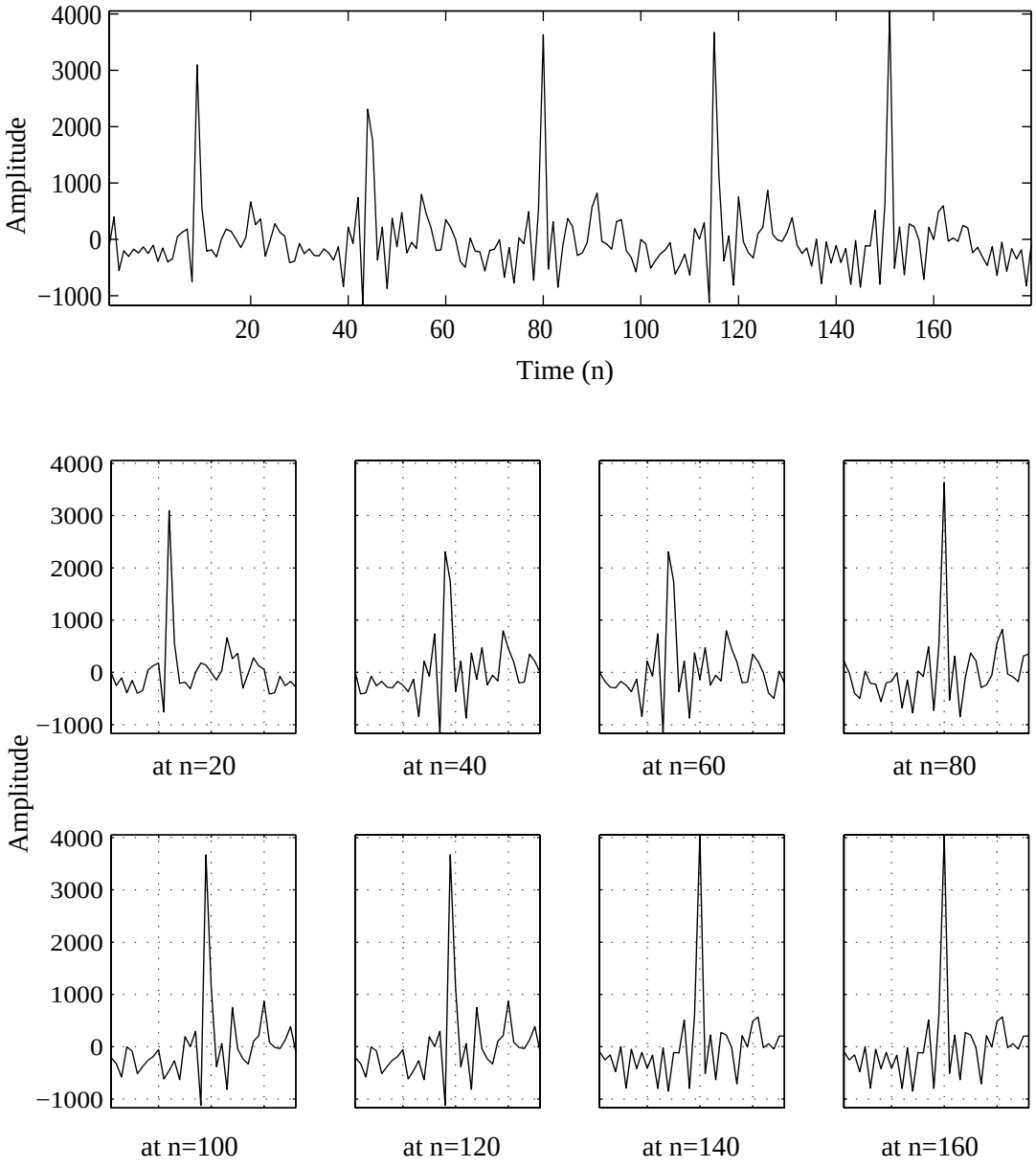


**Fig. 3.8** An example of the CWs extracted from a frame of residual signal. The top diagram illustrates the original residual signal and the extraction points are located at $n = 20, 40, \ldots, 160$. The bottom diagrams show the individual extracted CWs. Since the location of each extraction point is allowed to move by an offset of $\epsilon$ (between $-16$ and 16), the adjacent CWs may be extracted from the same residual segment. In this case, the CW extracted at $n = 100$ is identical to the one extracted at $n = 120$. The same happens to $n = 140$ and $n = 160$. Also note that the CWs at $n = 40$ and $n = 60$ are the same except for a small time shift.

to the number of future samples. These future samples are being provided by processor **110** and thus no additional algorithmic delay is required.

- The successive extraction windows overlap most of the time. In other words, two adjacent CWs may share the same residual segments. Further, because each extraction point is allowed to have an offset $\epsilon$ (between $-16$ and 16), so two adjacent CWs may even be identical. An example of this is given in Fig. 3.8 where the CWs extracted between $n = 100$ and 120 are the same. The same phenomenon happens between $n = 140$ and 160.

- For voiced speech, each extracted waveform can be interpreted as an individual pitch cycle. But for unvoiced speech, they are noise-like segments of varied lengths.

- In our implementation, a frame size has 160 samples. Since there are 8 extractions in a frame and each extraction has a minimum length of 20 samples ($P_{min}$), every sample in a frame should theoretically be included in at least one CW if $\epsilon$ is set to zero.

### 3.4.5 CW Alignment

The extraction procedure in processor **160** provides a DTFS description for every extracted CW. In general, these CWs are not in phase, that is, the main features in the waveforms are not time-aligned. In order to obtain an accurate description of the evolving CWs (like the one shown in Fig. 3.2c), an alignment of the CWs must be established.

In our implementation, this alignment is performed in processor **170** on a per subframe basis. Specifically, it works with two successive CWs — the current and the previous CWs. The processor aligns the current CW with the previous CW by introducing a circular time shift to the current one. Since the DTFS description allows us to regard the CW as a single cycle of a periodic signal (Section 3.3), this circular time shift is indeed equivalent to adding a linear phase to the DTFS coefficients.

Figure 3.9 shows a block diagram of the alignment processor **170**. In order to facilitate our understanding of this schematic, we will separate the discussion of the alignment process into three different scenarios. In the first scenario, we will assume that the alignment is performed between two CWs of the same dimension. Here, we

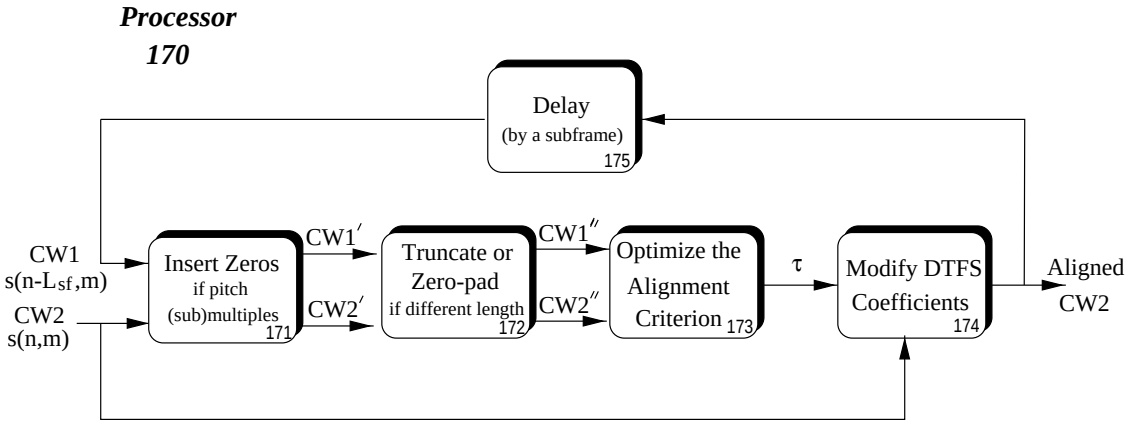


**Fig. 3.9** A block diagram of the alignment processor **170**

will discuss the alignment criterion (processor **173**) and the time shifter (processor **174**). The former determines the amount of time shift required by the current CW to align with the previous CW. The latter then circularly shifts the current CW by incorporating the time-shift resulting from processor **173** into its DTFS coefficients. Note that processor **171** and **172** are not needed in this scenario because of the equal dimension assumption.

In the second scenario, the current and the previous CWs will be assumed to have different dimensions (but with no pitch multiple or sub-multiple). Instead of deriving a new version of processor **173**, we add a preprocessor **172** before the **173** to accommodate the difference in dimensionality.

In the last scenario, we will discuss another preprocessor **171** which handles multiples and submultiples of the pitch.

### Scenario 1: Alignment with equal dimension

Let us begin with the first scenario where the previous and the current CWs are assumed to have the same length. Recalling from (3.5), the DTFS representation of a pair of successive CWs are:

$$\begin{aligned} s(n_0,m) &= \sum_{k=1}^{M}\left[A_k(n_0)\cos\left(\frac{2\pi km}{P}\right)+B_k(n_0)\sin\left(\frac{2\pi km}{P}\right)\right] \\ s(n_1,m) &= \sum_{k=1}^{M}\left[A_k(n_1)\cos\left(\frac{2\pi km}{P}\right)+B_k(n_1)\sin\left(\frac{2\pi km}{P}\right)\right] \end{aligned} \tag{3.12}$$

where $n_0$ and $n_1$ are the time indices of the previous and the current CWs respectively. Furthermore, for notational convenience,

$$\begin{aligned} P &= P(n) = P(n-1) \\ M &= \lfloor P(n)/2 \rfloor = \lfloor P(n-1)/2 \rfloor \end{aligned} \tag{3.13}$$

$P$ represents the length (pitch) of the CWs and $M$ is the number of spectral harmonics. In our implementation, because processor **170** operates on a per subframe basis, $n_1 - n_0 = L_{sf} = 20$.

Suppose now a circular time shift of $T$ samples (to the right) is applied to the current CW, $s(n_1, m)$ then becomes

$$s(n_1, m-T) = \sum_{k=1}^{M} \left[ A_k(n_1) \cos\left(\frac{2\pi k(m-T)}{P}\right) + B_k(n_1) \sin\left(\frac{2\pi k(m-T)}{P}\right) \right] \tag{3.14}$$

At this point, it is clear that a circular shift $T$ in the time-domain is equivalent to adding a linear phase $\frac{2\pi T}{P}$ in the DTFS domain. Next, in order to determine the amount of time shifting $T$ required to align $s(n_1, m-T)$ with $s(n_0, m)$, we can maximize the following criterion:

$$\begin{aligned} T = \arg\max_{0 \le T' < P} \sum_{k=1}^{M} \Bigg\{ & [A_k(n_0)A_k(n_1) + B_k(n_0)B_k(n_1)] \cos\left(\frac{2\pi k T'}{P}\right) + \\ & [B_k(n_0)A_k(n_1) - B_k(n_1)A_k(n_0)] \sin\left(\frac{2\pi k T'}{P}\right) \Bigg\} \end{aligned} \tag{3.15}$$

The right-hand side of (3.15) is the cross-correlation between the two CWs expressed in terms of the DTFS coefficients. A detailed proof of this can be found in [34]. Equation 3.15 can be also be expressed in terms of a normalized time shift $\tau$. By substituting:

$$\tau = \frac{2\pi T}{P} \tag{3.16}$$

into (3.15), we can obtain

$$\tau = \arg\max_{0\le\tau'<2\pi} \sum_{k=1}^{M} \{[A_k(n_0)A_k(n_1) + B_k(n_0)B_k(n_1)]\cos(k\tau') + [B_k(n_0)A_k(n_1) - B_k(n_1)A_k(n_0)]\sin(k\tau')\} \tag{3.17}$$

This equation will be referred to as the *alignment criterion* and it forms the basis for processor **173**.

One immediate advantage of performing the alignment in the DTFS domain is that it allows fractional alignment without any extra computations and the conventional upsampling and downsampling procedures are avoided. Also, this fractional alignment can be at any desired resolution ($\tau$ can be any real value between 0 and $2\pi$). In our WI implementation, we found that $\tau$ with a resolution of 1/4 of a sample (for a 8000 Hz sampling frequency) gives sufficiently good perceptual results.

The next step in the alignment processor is to incorporate the time shift $\tau$ into the DTFS coefficients of the current CW, $s(n_1, m)$. This can be done by expanding the sin and cos terms in (3.14) using fundamental trigonometric identities. By grouping the relevant terms together in the resulting expansion, we would have a new set of DTFS coefficients:

$$\left.\begin{aligned} A'_k(n_1) &= A_k(n_1)\cos\left(\frac{2\pi kT}{P}\right) - B_k(n_1)\sin\left(\frac{2\pi kT}{P}\right) \\ B'_k(n_1) &= A_k(n_1)\sin\left(\frac{2\pi kT}{P}\right) + B_k(n_1)\cos\left(\frac{2\pi kT}{P}\right) \end{aligned}\right\} \quad \text{for } k = 1, 2, \ldots, M \tag{3.18}$$

where[5]

$$s(n_1, m - T) = \sum_{k=1}^{M}\left[A'_k(n_1)\cos\left(\frac{2\pi km}{P}\right) + B'_k(n_1)\sin\left(\frac{2\pi km}{P}\right)\right] \tag{3.19}$$

$\{A'_k(n_1)\}$ and $\{B'_k(n_1)\}$ now represent the new DTFS coefficients for the CW time-shifted by $T$ samples to the right. Equation 3.18 can also be expressed in terms of

[5]All the DTFS coefficients will be assumed to be properly aligned after the discussion of this alignment Section 3.4.5 and therefore, the prime symbols will be omitted.

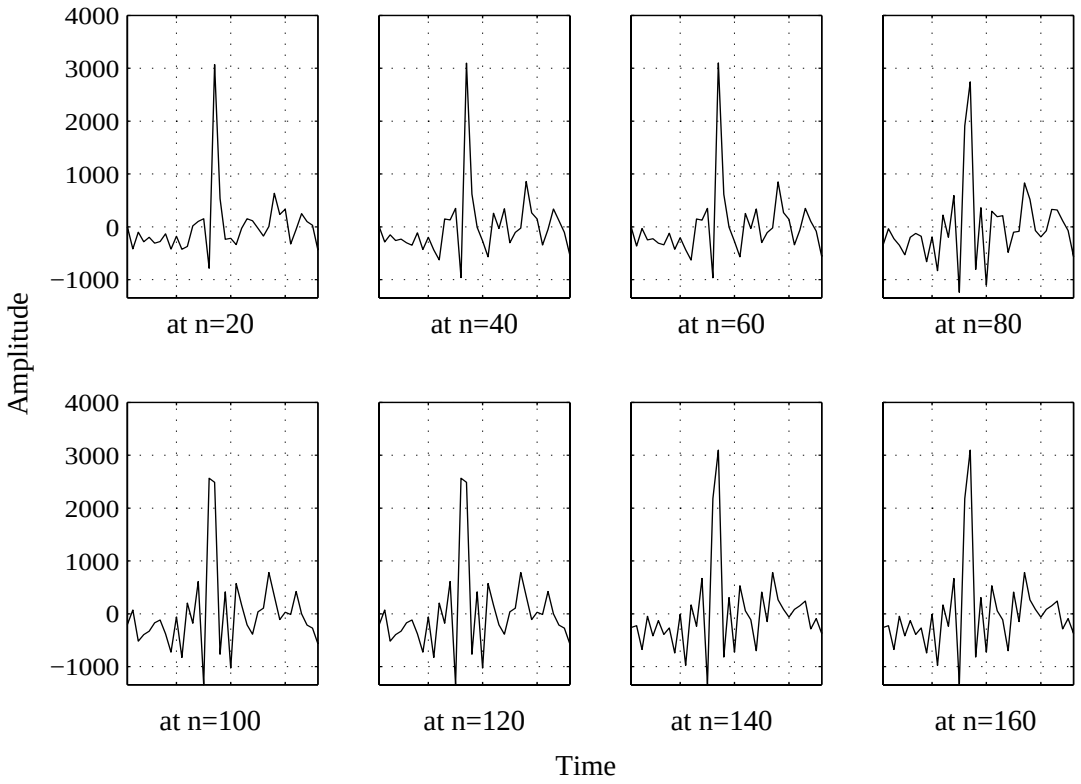


**Fig. 3.10** Illustration of the aligned CWs with an alignment resolution of 1/4 of a sample. The unaligned version of the same waveforms were previously shown in Fig. 3.8.

the normalized time shift $\tau$ using (3.16):

$$\left.\begin{aligned} A'_k(n_1) &= A_k(n_1)\cos(k\tau) - B_k(n_1)\sin(k\tau) \\ B'_k(n_1) &= A_k(n_1)\sin(k\tau) + B_k(n_1)\cos(k\tau) \end{aligned}\right\} \quad \text{for} \quad k = 1, 2, \ldots, M \tag{3.20}$$

In summary, processor **173** utilizes (3.17) to find the optimized $\tau$ and processor **174** then uses (3.20) to incorporate this $\tau$ into the DTFS coefficients. An example of a sequence of aligned CWs is demonstrated in Fig. 3.10.

### Scenario 2: Alignment with different dimensions

So far, we have assumed that $s(n_0, m)$ and $s(n_1, m)$ have the same length. But in general, they come in different dimensions. In other words, the alignment criterion

(3.17), which is based on the equal dimension assumption (3.13), is no longer directly applicable. To avoid deriving a new alignment criterion, we dedicate processor **172** to pre-process the CWs in either one of the following ways so that they can be in the same length prior to the alignment criterion.

1. spectrally truncate the longer CW to the length of the shorter one
2. spectrally pad zeros to the shorter CW to match the length of the longer CW

In the first approach, discarding the high frequency harmonics would result in a temporal shrinking of the CW. Although the CW may lose some temporal details in this process, the harmonics at the high end of the spectrum tend to have relatively

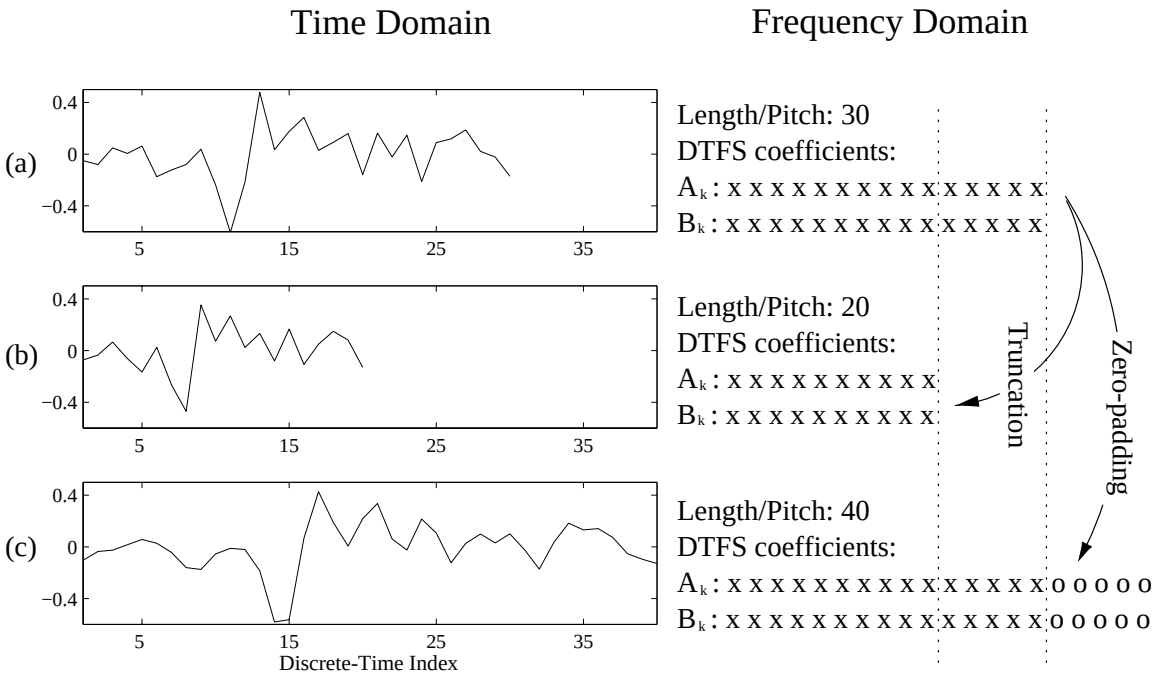


**Fig. 3.11** Time-Scaling of a CW. (a) A CW of length 30 samples and its corresponding DTFS coefficients, 15 $\{A_k\}$ + 15 $\{B_k\}$ (the DC component is not shown). (b) The time-contracted version of the waveform in (a) as a result of truncating the harmonics at the high frequency end. The approximate temporal shape of the CW is still preserved after the contraction but the details are mostly lost. (c) The time-stretched version of the same waveform after zeros have been appended to the spectral series. Such stretching introduces no extra information to the time-sequence, but it offers a higher resolution.

low energy. Consequently, the shape of the truncated CW generally approximates very well to that of the original.

In the second option, the zero-padding in the spectral domain causes the shorter CW to be temporally stretched to match the length of the longer one. Such time-stretching is indeed equivalent to the bandlimited interpolation of the CW in time domain. The interpolation itself does not introduce any new temporal information to the sequence, but it offers a higher resolution. Figure 3.11 shows an example of a CW being time-contracted and time-stretched.

Experiments confirm that the two approaches yield different numerical results (different $\tau$'s); however, they give indistinguishable perceptual quality in the reconstructed speech. Nevertheless, the first approach takes an upper-hand when considering the computational complexity. Truncating the sequence means that there are fewer coefficients left in the series and hence, fewer multiplications are involved with the alignment criterion.

**Scenario 3: Alignment with pitch (sub)multiple**

As discussed in Section 3.4.3, pitch doubling/tripling/halving occasionally happens in natural speech. Thus, multiple or sub-multiple pitch cycles can appear in an extracted CW. To prevent alignment problems, the shorter CW is periodically extended an integer number of times in processor **171**, such that it most closely matches the length of the longer CW, prior to the alignment criterion. In spectral domain, this is equivalent to inserting zero-amplitude harmonics between the original harmonics in the shorter CW. Figure 3.12 demonstrates how zeros are inserted between the DTFS coefficients $\{A_k, B_k\}$ and its corresponding time-domain effect.

To detect whether a pitch (sub)multiple has occurred, we can proceed in the same fashion as in Section 3.4.3 where we use the indicator $C$. If this indicator is greater than unity, then pitch multiplication or division is considered to have occurred. If $C = 2$, it indicates that the pitch has doubled and we insert one zero to the shorter CW to periodically extend it once (Fig. 3.12b). If $C = 3$, it suggests that the pitch has tripled, so we insert two zeros between harmonics of the shorter CW to extend it twice (Fig. 3.12c). Other multiplies can be dealt with similarly.

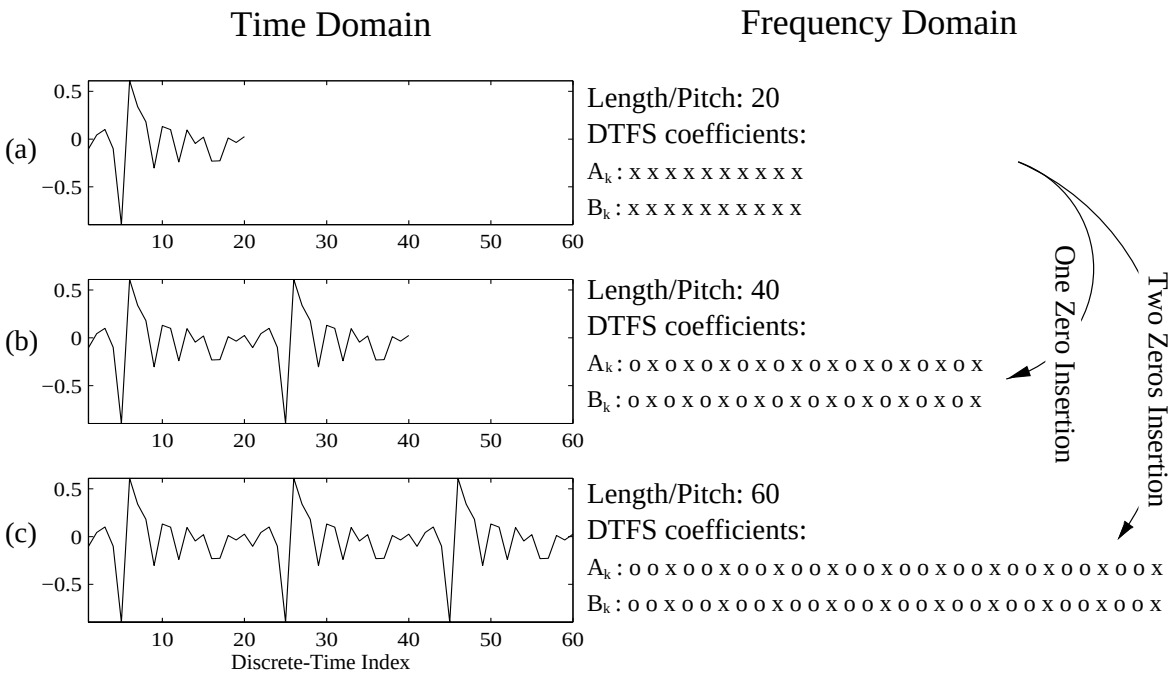


**Fig. 3.12** Illustration of the zero-insertion between spectral samples. (a) A CW of length 20 samples and its corresponding DTFS coefficients, $10\,\{A_k\} + 10\,\{B_k\}$ (the DC component is not shown). (b) The waveform in (a) is periodically extended once after one zero is inserted between adjacent harmonics. (c) The waveform in (a) is periodically extended twice after two zeros are inserted between adjacent harmonics

*Remarks on the alignment process*

- The alignment is performed between two successive CWs. The same rule applies at frame boundaries; the first extracted CW in the current frame is aligned with the last extracted CW in the previous frame.

- The time-scaling in processor **172** works under an assumption that the shape, as well as the length, of the CWs evolve continuously. However, a number of experiments were carried out in [33] and showed that time-scaling a LP residual signal might not be "a proper thing" to do. It suggested there that zero-padding in the time domain could be more appropriate than zero-padding in the spectral domain for a LP residual signal.

- In Fig. 3.11b, it is clear that the power of the signal (energy per sample) di-

minishes after it is spectrally truncated. In contrast, the zero-padding and the zero insertion procedures (Figs. 3.11b and 3.12) seem to preserve the power. The reason for this will become clear as we proceed to Section 3.4.6. For the time being, the power drop in the spectral truncation will reduce the cross-correlation values in (3.17) by a factor. However, this inaccuracy is transparent to the arg max operator in (3.17) and thus has no effect on the final value of $\tau$.

- Straightforward evaluation of the alignment criterion (3.17) could be very computationally expensive, particularly for long CWs. For instance, if both $s(n_0, \cdot)$ and $s(n_1, \cdot)$ are of 90 samples long, there will be $90 \times 4 = 360$ cross-correlations needed to be calculated (assuming that the alignment resolution is 1/4 of a sample). Each of these cross-correlation will take at least $90 \times 2 = 180$ multiplications according to (3.17). Thus, the total computational cost involved in the alignment criterion alone is about $360 \times 180 = 28800$ multiplications! However, one can make use of recursion to reduce such complexity. In particular, at the first step of the recursion process, one can first align the waveform with a very coarse alignment resolution but with a full alignment range (from 0 to $2\pi$). The $\tau$ resulting from this process will then be used by the next recursion call to narrow the alignment range and at the same time, the alignment is performed with a finer alignment resolution. Therefore, as the recursive call continues, the alignment range becomes narrower and the alignment resolution becomes finer. Finally, when it hits the desired resolution (1/4 of a sample), the recursion terminates. Such an alignment procedure not only reduces the complexity by an order of magnitude, but also ensures that the CWs are properly aligned during the periodic segments. In fact, the recursion procedure will most likely find a *local* maximum correlation rather than a *global* one. This is very advantageous to the coder since it reduces the amount of periodicity generated for an original signal which is not periodic. Such a feature is particularly desirable for non-periodic speech segments, since an excessive periodicity in an unvoiced speech would cause buzziness.

- As discussed in Section 3.4.4, the CWs are extracted in a way to avoid high boundary energies; however, due to the nature of circular shifting, the alignment process may result in a CW with significant boundary energy at its borders. Yet,

this will not cause any discontinuities in the reconstructed speech since the CWs have been periodically extended before the alignment process.

### 3.4.6 CW Power Computation and Normalization

After the CWs are extracted and aligned, their powers [6] are then normalized. Thus, the relationship between a normalized and unnormalized CW is expressed in terms of a power. The main motivation of this normalization is to separate the power and shape in CWs so that they can be quantized separately to achieve higher coding efficiency.

The approach here is to first use the power extraction processor **180** to compute the power of each CW. Then, this power together with the CW are passed to the power normalizer **190**. Since all CWs have been previously converted to their DTFS domain, the power computation and normalization are performed on the DTFS coefficients $\{A_k, B_k\}$. Therefore, we will first formulate the relationship between the power of a CW and its DTFS coefficients.

The average power of a CW at time $n$, denoted as $\Psi(n)$, can be expressed as:

$$\Psi(n) = \frac{1}{P(n)} \sum_{m=0}^{P(n)-1} |s(n,m)|^2 \tag{3.21}$$

where $P(n)$ is the length of the CW. Combining (3.5) and (3.21), we can obtain:

$$\begin{aligned}\Psi(n) &= \frac{1}{P(n)} \sum_{m=0}^{P(n)-1} s(n,m)s^*(n,m) \\ &= \frac{1}{P(n)} \sum_{m=0}^{P(n)-1} s(n,m) \sum_{k=1}^{\lfloor P(n)/2 \rfloor} \left[ A_k^*(n) \cos\left(\frac{2\pi km}{P(n)}\right) + B_k^*(n) \sin\left(\frac{2\pi km}{P(n)}\right) \right]\end{aligned}$$

Since we deal with real speech and residual samples, $\{A_k(n)\}$ and $\{B_k(n)\}$ will always be real which implies:

$$\begin{aligned} A_k(n) &= A_k^*(n) \\ B_k(n) &= B_k^*(n) \end{aligned} \tag{3.22}$$

[6] In this thesis, the power of a CW is defined to be the average energy per sample over exactly one pitch cycle.

Also, for the sake of simplicity, we will omit the index $n$ from the expression because we are dealing with one particular time index only. $\Psi(n)$ then becomes

$$\Psi = \frac{1}{P}\sum_{m=0}^{P-1} s(m) \sum_{k=1}^{\lfloor P/2 \rfloor} A_k \cos\left(\frac{2\pi km}{P}\right) + \frac{1}{P}\sum_{m=0}^{P-1} s(m) \sum_{k=1}^{\lfloor P/2 \rfloor} B_k \sin\left(\frac{2\pi km}{P}\right)$$

By interchanging the order of the two summations in each term, we obtain

$$\Psi = \frac{1}{P}\sum_{k=1}^{\lfloor P/2 \rfloor} A_k \sum_{m=0}^{P-1} s(m) \cos\left(\frac{2\pi km}{P}\right) + \frac{1}{P}\sum_{k=1}^{\lfloor P/2 \rfloor} B_k \sum_{m=0}^{P-1} s(m) \cos\left(\frac{2\pi km}{P}\right)$$

Now, we can make use of (3.3) and (3.4) and construct

$$\Psi = \begin{cases} \dfrac{1}{2}\displaystyle\sum_{k=1}^{P/2-1} \left(A_k^2 + B_k^2\right) + A_{P/2}^2 + B_{P/2}^2 & \text{for } P \text{ even} \ , \\ \dfrac{1}{2}\displaystyle\sum_{k=1}^{\lfloor P/2 \rfloor} \left(A_k^2 + B_k^2\right) & \text{for } P \text{ odd} \ . \end{cases} \tag{3.23}$$

Equation 3.23 is the formula that processor **180** requires to determine the CW power from its DTFS coefficients.

As for processor **190** which does the actual power normalization, it can be formulated as follows. Dividing (3.23) by $\Psi$, we can obtain a unit power on the left:

$$1.0 = \begin{cases} \dfrac{1}{2\Psi}\displaystyle\sum_{k=1}^{P/2-1} \left(A_k^2 + B_k^2\right) + \dfrac{A_{P/2}^2}{\Psi} + \dfrac{B_{P/2}^2}{\Psi} & \text{for } P \text{ even} \ , \\ \dfrac{1}{2\Psi}\displaystyle\sum_{k=1}^{\lfloor P/2 \rfloor} \left(A_k^2 + B_k^2\right) & \text{for } P \text{ odd} \ . \end{cases} \tag{3.24}$$

Incorporating the $\Psi$ into each of the DTFS coefficients, (3.24) becomes

$$1.0 = \begin{cases} \dfrac{1}{2}\displaystyle\sum_{k=1}^{P/2-1} \left[\left(\frac{A_k}{\sqrt{\Psi}}\right)^2 + \left(\frac{B_k}{\sqrt{\Psi}}\right)^2\right] + \left(\frac{A_{P/2}}{\sqrt{\Psi}}\right)^2 + \left(\frac{B_{P/2}}{\sqrt{\Psi}}\right)^2 & \text{for } P \text{ even} \ , \\ \dfrac{1}{2}\displaystyle\sum_{k=1}^{\lfloor P/2 \rfloor} \left[\left(\frac{A_k}{\sqrt{\Psi}}\right)^2 + \left(\frac{B_k}{\sqrt{\Psi}}\right)^2\right] & \text{for } P \text{ odd} \ . \end{cases} \tag{3.25}$$

Thus, the normalization procedure is simply to divide each DTFS coefficient by the

square root of the average power, $\sqrt{\Psi}$.

In summary, for every incoming CW, the power calculator **180** utilizes (3.23) to compute the power $\Psi$ from its DTFS coefficients. The normalizer **190** then divides each of its DTFS coefficients by a factor of $\sqrt{\Psi}$ according to (3.25) in order to obtain the unit-power description of the CW.

*Remarks on the power normalization*

- Equation 3.23 indicates that the average power of a CW is roughly proportional to the sum of the energies of all harmonic components. This should explain why adding zero-amplitude harmonics to a CW spectrum preserves its original power (Figs. 3.11c, 3.12b and 3.12c). This also explains the power reduction resulting from the harmonic truncation in Fig. 3.11b.

- It makes no difference if the alignment processor **170** is placed after or before the power normalizer **180**. In other words, it does not matter if the alignment is performed on the normalized or unnormalized CWs.

- The power normalizer **180** should be disabled when the CW power (computed in processor **170**) is significantly small which is a strong indication of a silent segment. Otherwise, a computational overflow may easily occur during the normalization procedure.

### 3.4.7 Output of the Analysis Layer

In summary, the analysis layer decomposes a speech segment into four parameters — pitch, LSFs, power and CWs. The first two have an update rate of once per frame while the last two are being computed once every subframe (eight times per frame). It should be emphasized that these CWs form a two-dimensional evolving waveform surface. Figure 3.13 illustrates an example of such surface for a voicing onset segment. The original residual signal is also shown.

Normally, the four parameters are sent to processors **300** and **400** for quantization and de-quantization, which will be discussed in Chapter 4. However, if the coder is set to run on the analysis-synthesis layer only, these parameters will be directly sent to processor **200** (Fig. 3.1).

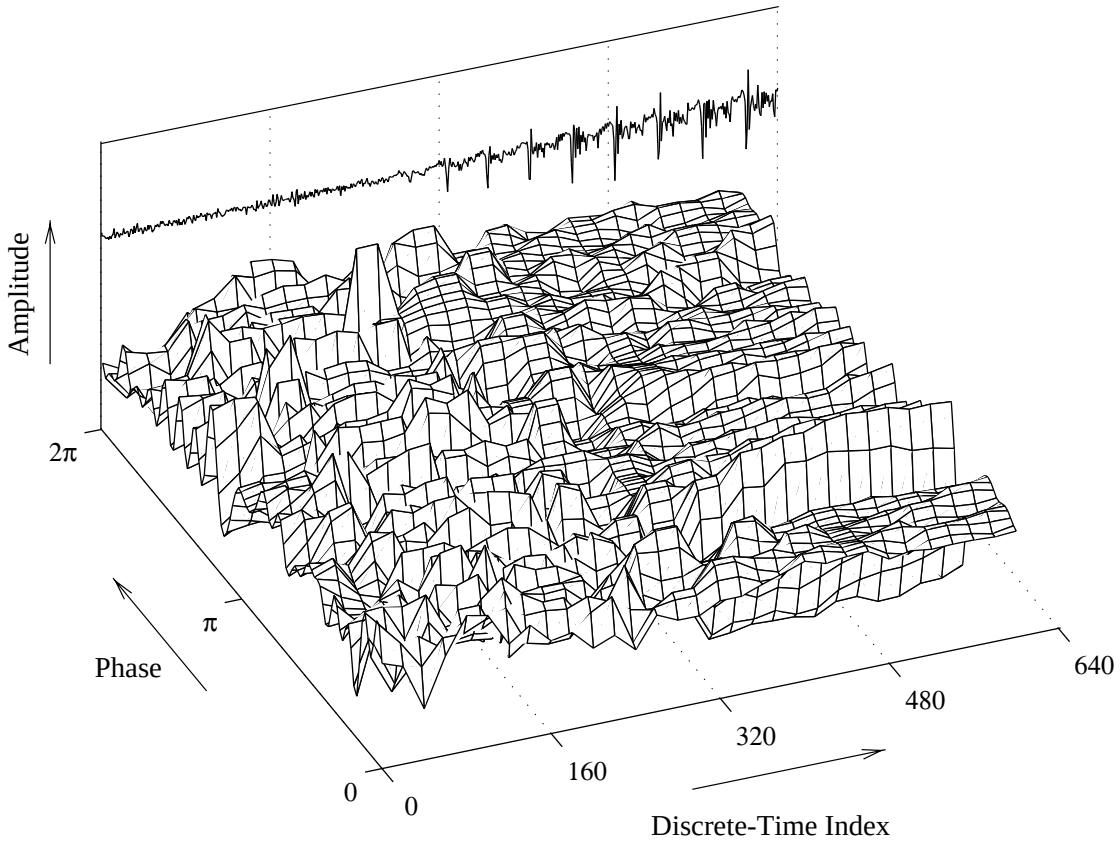


**Fig. 3.13** Decomposition of a residual signal into a CW evolving surface. The CWs are extracted eight times per frame which is equivalent to one extraction per 20 samples. The power of each CW is normalized and its length is normalized to a period of $2\pi$ as well.

## 3.5 The Synthesis Stage

From the LSFs, pitch, power and the normalized CWs, the speech signal can be reconstructed in the synthesis processor **200**. Note that when the coder is operating with the quantization layer, the synthesis block receives the quantized version of these parameters.

A schematic of the synthesis processor is shown in Fig. 3.14. Similar to the processors at the encoder, the frequency of execution varies from processor to processor in the synthesis layer. Some processors operate on a frame-by-frame basis, while others are executed once per subframe or even once per sample.

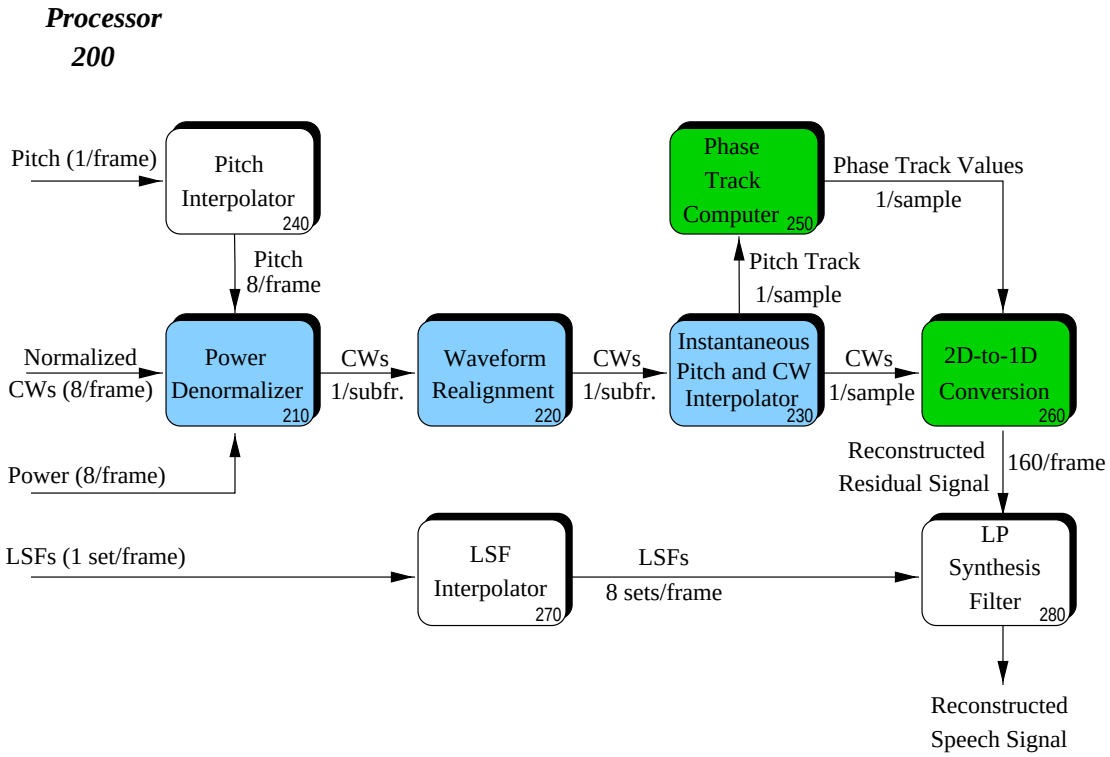


**Fig. 3.14** A block diagram of the WI decoder which is an expanded version of processor **200** in Fig. 3.1. The lighter-shaded processors are executed once per subframe while the darker-shaded ones are executed once per sample. The non-shaded ones process once per frame.

Since processors **240** and **270** are identical to processors **150** and **120** respectively, their functional descriptions will not be repeated in this section.

### 3.5.1 CW Power Denormalization and Realignment

As a first step, the power of each incoming CW is denormalized in processor **210**. This denormalizer uses the pitch information supplied by the interpolator **240** to determine the length of the CW. To restore the power in the CW, we reverse the process that we did in (3.25) by multiplying each of the DTFS coefficients by $\sqrt{\Psi}$.

After the CWs have been denormalized, they are sent to the CW realigner **220**. If the coder is operating without quantization, this realigner is unnecessary because the CWs have already been aligned previously at the encoder by processor **170**. On the other hand, if the coder parameters are being quantized, the successive CWs may

no longer be exactly aligned after they are decoded.

### 3.5.2 Instantaneous Pitch and CW Generation

We now have a fully reconstructed, aligned CW and its pitch length at every subframe interval. In the WI paradigm, a CW and a pitch length at every sample point are required to reconstruct the one-dimensional residual signal. These instantaneous[7] CWs and pitch values are generated in processor **230**.

Linear interpolation can be used to upsample the CWs. When the upsampling is performed between two CWs of the same dimension, straightforward interpolation can be applied. However, if the CWs are of different dimensions or if the length of one is a (sub)multiple of the other, extra processing is required to ensure a smooth interpolation.

As for pitch interpolation, it will work differently from processor **150** in the analysis layer. The interpolation will still be linear but it will not employ the pitch (sub)multiple equations (3.10) and (3.11). It should be emphasized that the instantaneous pitch values generated from this interpolation should always correspond to the lengths of the instantaneous CWs.

Figure 3.15 shows a schematic of the interpolator **230** which is capable of handling the CW and the pitch interpolations in three different scenarios: (i) equal dimension, (ii) different dimensions and (iii) pitch (sub)multiples.

#### Scenario 1: Interpolation with equal dimension

First, we assume that the interpolation is performed between two CWs of the same length $P$. As a result of this assumption, processors **231**, **232** and **234** will not be executed. If we denote $n_0$ and $n_1$ to be the time instants at the boundaries of an interpolation interval, then the instantaneous CW, $s(n, m)$, at index $n$ can be computed by interpolating between $s(n_0, m)$ and $s(n_1, m)$. In time domain, this interpolation can be expressed as:

$$s(n,m) = \left(\frac{n_1 - n}{n_1 - n_0}\right) s(n_0, m) + \left(\frac{n - n_0}{n_1 - n_0}\right) s(n_1, m) \qquad n_0 \leq n \leq n_1,\, 0 \leq m < P \tag{3.26}$$

[7] In this thesis, we use the word *instantaneous* to describe a coder parameter that has the same update rate as the sampling rate of the input/output speech signal (i.e., 8000 Hz).

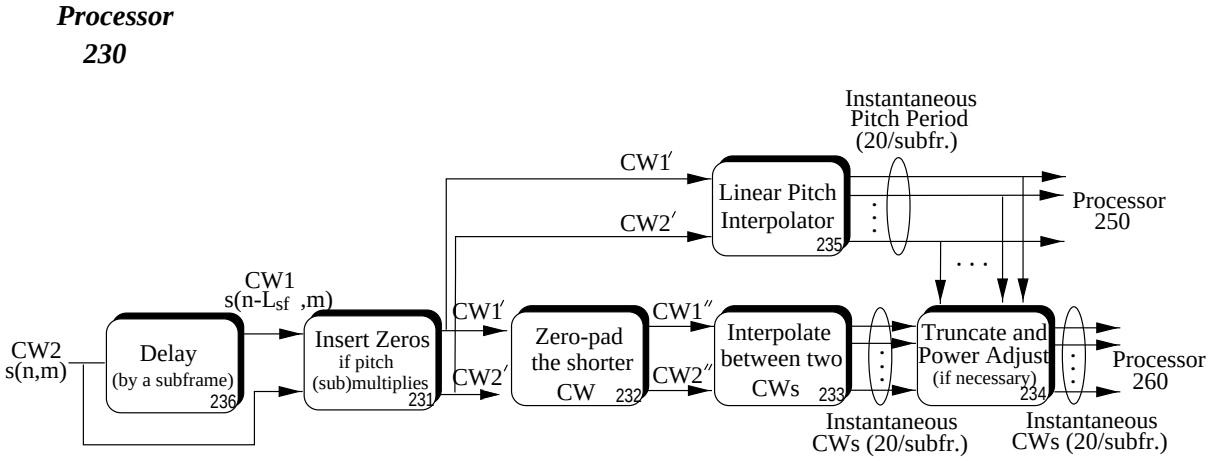


**Fig. 3.15** A block diagram of the interpolator processor

Substituting (3.5) in (3.26), one can obtain

$$\left.\begin{aligned} A_k(n) &= \left(\frac{n_1-n}{n_1-n_0}\right)A_k(n_0) + \left(\frac{n-n_0}{n_1-n_0}\right)A_k(n_1) \\ B_k(n) &= \left(\frac{n_1-n}{n_1-n_0}\right)B_k(n_0) + \left(\frac{n-n_0}{n_1-n_0}\right)B_k(n_1) \end{aligned}\right\} \text{ for } \quad k = 1, 2, \ldots, \lfloor P/2 \rfloor \quad (3.27)$$

In other words, the linear interpolation between the two CWs in the time domain is equivalent to the linear interpolation of their DTFS coefficients. The interpolation is performed on a per-subframe basis, therefore $n_1 - n_0 = L_{sf} = 20$.

Since the CWs at the boundaries are of the same length, the interpolated CWs in between will be of the same length as well. As a result, a constant pitch contour will result in processor **235**.

### Scenario 2: Interpolation with different dimension

In general, the pitch will change over the interval and the CWs at the boundaries will be of different lengths (different number of coefficients $\{A_k, B_k\}$). To facilitate the interpolation in this case, one can time-scale the shorter CW to the length of the longer one prior to the interpolation. As stated in Section 3.4.5, such time-scale operation is equivalent to padding zero harmonics to its DTFS description. Therefore, processor **232** is set up to first pad zeros to the shorter CW until it matches the length of the longer CW. Afterwards, processor **233** can interpolate the CWs in the same fashion as in Scenario 1 to obtain the instantaneous CWs. Processor **231** remains

inactive.

In processor **235**, the conventional linear interpolation equation (3.9) can be used to upsample the pitch. Yet, the upsampled pitch values resulting from this process may not be consistent with the pitch lengths of the interpolated CWs (output of **233**), as a result of the zero-padding in **232**. To avoid this inconsistency, we set up processor **234** to time-contract each of these CWs to match the length specified by the interpolated pitch contour. An example of such operation is shown in Fig. 3.16 for the $\{A_k\}$ coefficients where the interpolation is done between two CWs of lengths 26 and 34. The same operation is applied to the $\{B_k\}$ coefficients. We should recall from Section 3.4.5 that the temporal contraction of a CW can be achieved by truncating the trailing coefficients of its DTFS representation. Also, as discussed in Section 3.4.6, the truncation of DTFS coefficients may result in a power drop in the CWs. To maintain the same power level after the time-contraction, we need to scale the DTFS coefficients by an appropriate factor.

**Scenario 3: Interpolation with pitch (sub)multiple**

If the current CW is significantly longer or shorter than the previous CW, it indicates that pitch (sub)multiple has most likely occurred. Processor **231** is set up to accommodate these occurrences. Similar to the alignment processor **170**, this processor uses the same indicator $C$ as a pitch (sub)multiple decision criterion. If pitch has indeed multiplied or submultiplied ($C > 1$) over the subframe interval, the processor repeats the shorter CW an integer number of times such that it closely matches the length of the longer CW. This procedure is equivalent to inserting zero-amplitude harmonics between the original harmonics in the DTFS representation. After zeros are inserted, the CWs are then sent to processor **232** and processed in the same way as in Scenario 2.

Since the zeros inserted by processor **231** will be left in the CWs which will subsequently be sent to processor **260**, processor **235** interpolates the pitch length *after* the zeros are inserted (Fig. 3.15) in the CWs. In this way, the instantaneous pitch length that goes processor **250** will accurately reflect the length of the instantaneous CW that goes to processor **260**.

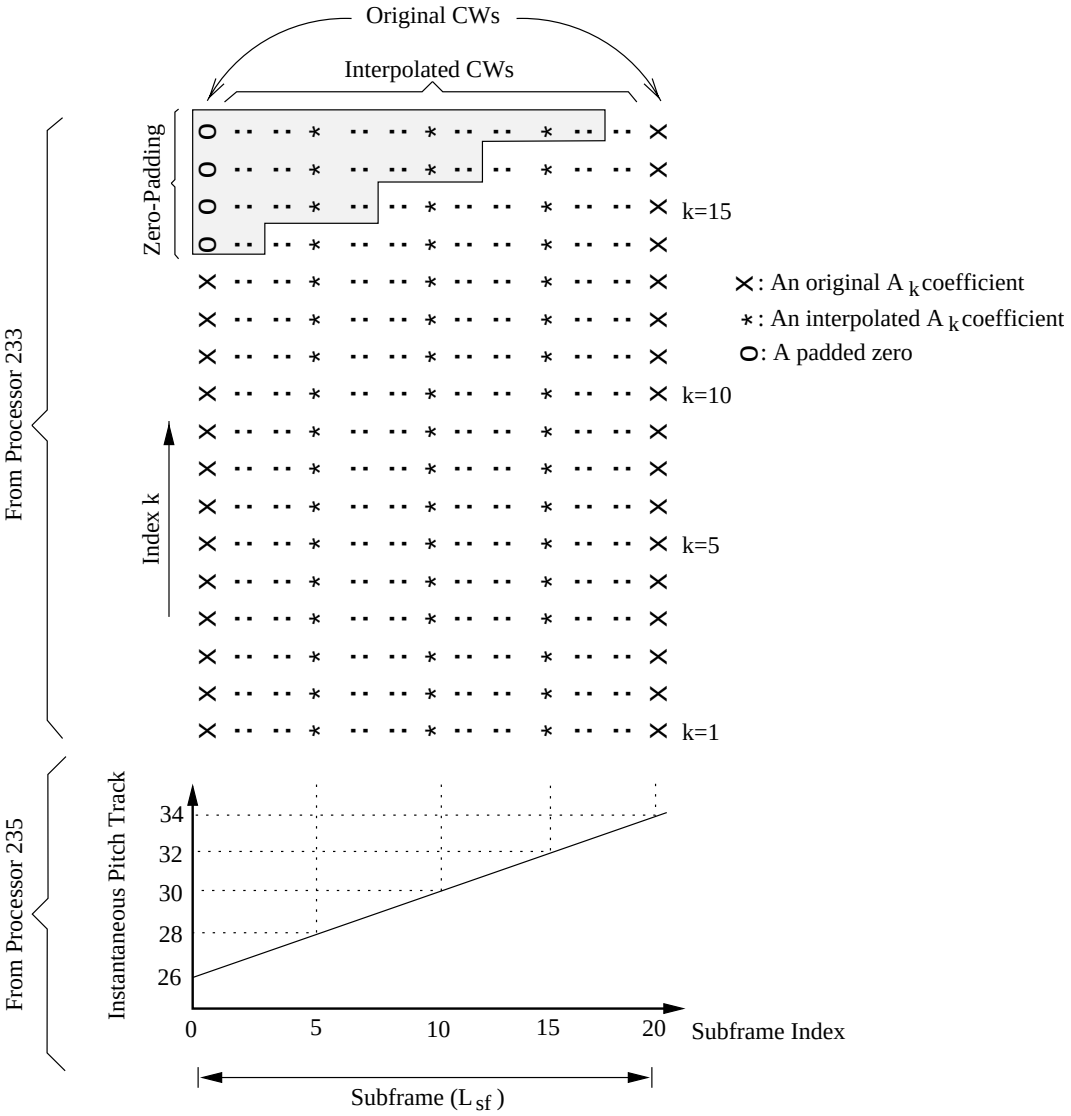


**Fig. 3.16** An example of the CW interpolation over a subframe interval. Suppose the interpolation is performed between two successive CWs over a subframe of $L_{sf}$. The CW at the left subframe boundary is of length 26 and it has 13 harmonics. The CW at the right is of length 34 and it has 17 harmonics. The original as well as the interpolated $\{A_k\}$ coefficients are illustrated in the top diagram. The bottom graph shows the corresponding pitch track supplied by processor **235**. As one can easily see, because of the zero-padding, all the interpolated CWs would have an immediate length of 17 $\{A_k\}$ coefficients (same length as the longer CW) which is inconsistent with the pitch length specified by the pitch track. To cure this problem, we time-contract each interpolated CW so as to match the pitch track description. As a result, the coefficients within the shaded area need to be trimmed.

*Remarks on the CW and pitch interpolation*

- At a frame boundary, the interpolation is done between the last CW in the previous frame and the first CW in the current frame.
- The instantaneous pitch values that are going to processor **250** are not rounded to integers so as to enhance the accuracy of the phase track.

### 3.5.3 Phase Track Estimation

In this section, we are concerned with processor **250** whose objective is to convert the pitch values into a phase track. This phase track will be used by processor **260** to transform the two-dimensional CW surface back into the one-dimensional residual signal. Since we already have a pitch value at every sample point, the phase track can be computed by incrementally summing the area under the frequency track curve $F(n)$. The relationship between $F(n)$ and $P(n)$ can be expressed as:

$$P(n) = \frac{1}{F(n)}\,, \tag{3.28}$$

Now, if we designate $\phi(\cdot)$ as the phase track, the phase contour at each sample point can be updated on a per-sample basis by

$$\phi(n) = \phi(n-1) + \int_{n-1}^{n} \frac{2\pi}{P(n')} dn' \tag{3.29}$$

where $\phi(n)$ and $\phi(n-1)$ are the current and the previous phase values. The integral corresponds to the incremental area between the interval $n-1$ and $n$. Assuming the pitch evolves linearly over this integration interval, (3.29) can be expanded into:

$$\phi(n) = \phi(n-1) + \int_{n-1}^{n} \frac{2\pi}{(n-n')P(n-1) + (n'-n+1)P(n)} dn' \tag{3.30}$$

and straightforward evaluation of this integral leads to

$$\phi(n) = \begin{cases} \phi(n-1) + \dfrac{2\pi}{P(n) - P(n-1)} \ln\left[\dfrac{P(n)}{P(n-1)}\right] & \text{for } P(n) \neq P(n-1), \\ \phi(n-1) + \dfrac{2\pi}{P(n)} & \text{for } P(n) = P(n-1). \end{cases} \tag{3.31}$$

By executing (3.31) on a sample-by-sample basis, processor **250** can convert the pitch track $P(\cdot)$ into the phase track $\phi(\cdot)$. It should be emphasized that $\phi(n)$ is an increasing function since the integral in (3.30) always results in a positive value. Therefore, special attention should be paid to prevent $\phi(n)$ from overflowing by subtracting off a multiple of $2\pi$ when appropriate.

The $\ln(\cdot)$ operator in (3.31) is a rather computationally expensive operator considering that it is executed once per sample. To reduce the complexity, a technique similar to the Riemann sum can be adopted to approximate the integral in (3.29). Specifically, the integral $\int_{n-1}^{n} \frac{dn'}{P(n')}$ can be approximated by a rectangular polygon which has a height of

$$\frac{1}{2}\left(\frac{1}{P(n-1)} + \frac{1}{P(n)}\right) \tag{3.32}$$

and a width of one sample long. Note that (3.32) represents the average of the two successive frequency values — $F(n-1)$ and $F(n)$. As a result of this approximation, (3.29) can be rewritten as:

$$\phi(n) \approx \phi(n-1) + \pi\left(\frac{1}{P(n-1)} + \frac{1}{P(n)}\right) \tag{3.33}$$

Figure 3.17 shows the difference graphically between (3.31) and (3.33). The difference is quite subtle and also, since a slow wandering of the phase of the speech signal waveform does not make a perceptual difference [9], (3.33) is therefore a viable approximation to (3.31) in a practical implementation.

Note that at the beginning of a speech signal, the value of $\phi(0)$ can be set to an arbitrary number. The initial phase offset value does not affect the perceptual quality of the reconstructed speech [30].

### 3.5.4 2D-to-1D Transformation

Processor **260** converts the two-dimensional CW surface $\{A_k(\cdot), B_k(\cdot)\}$ to the one-dimensional residual signal $r(\cdot)$. This conversion operation is done on a sample-by-sample basis and is best illustrated by an example: Figure 3.18 demonstrates the reconstruction process for a residual segment from a female speaker. The interpolated phase track corresponding to the segment (from processor **250**) is illustrated

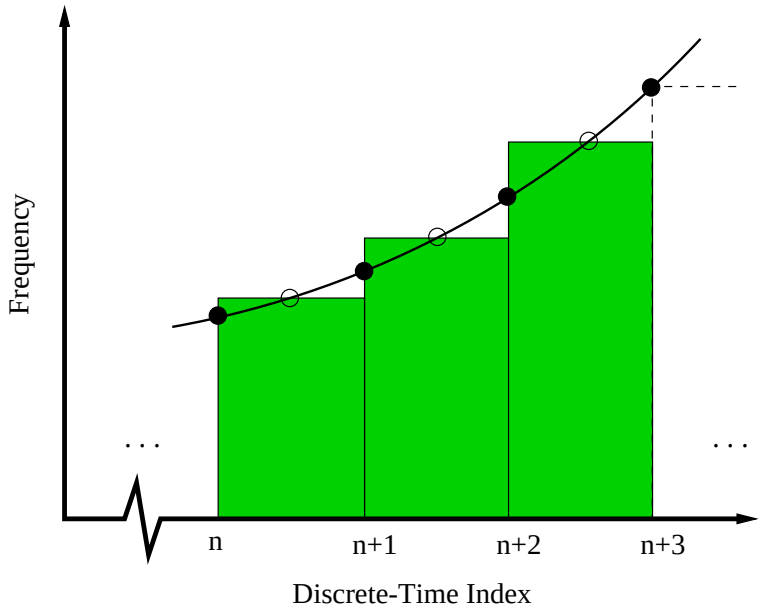


**Fig. 3.17** Comparisons between the two phase track computation methods. A solid dot denotes the frequency value at a discrete-time instant while a hollow dot denotes the average of the two adjacent frequency values. The solid curve corresponds to the interpolated frequency track and the area under this curve corresponds to the phase track calculated by (3.31). On the other hand, the area bounded within the shaded rectangular polygons corresponds to the phase track computed by (3.33).

in Fig. 3.18a. Figure 3.18b shows the interpolated CW surface (from processor **230**) where each CW is normalized to a length of $2\pi$. To perform the transformation, we superimpose the two graphs and the projection of the intersection (where the phase track meets the CW surface) onto the plane perpendicular to the phase axis is $r(n)$.

The transformation can be implemented as an inverse DTFS operation. More precisely, $r(n)$ can be determined by using (3.7) where $\phi$ is a function of $n$:

$$r(n) = s(n, \phi(n)) = \sum_{k=1}^{\lfloor P(n)/2 \rfloor} [A_k(n) \cos(k\phi(n)) + B_k(n) \sin(k\phi(n))] \qquad 0 \le \phi(\cdot) < 2\pi \tag{3.34}$$

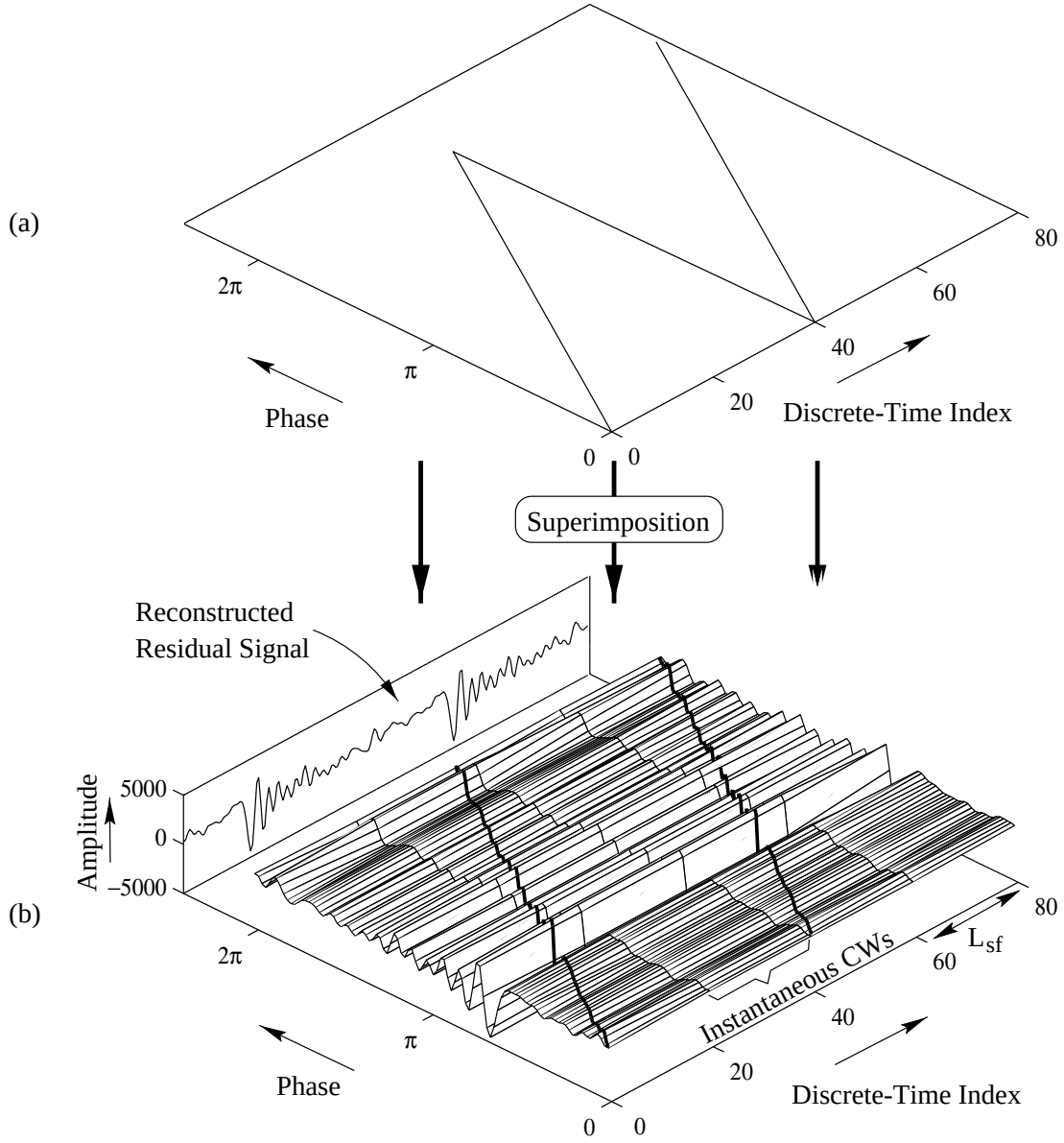


**Fig. 3.18** Transformation from a CW surface to a residual signal. (a) An interpolated (instantaneous) phase track for a voiced segment which has a constant pitch at 40. (b) The interpolated pitch track superimposed on the interpolated CW surface. The contours in planes perpendicular to the time axis represent the CWs. The intersection of the two specifies the reconstructed residual signal. Note that the resulting residual has a pitch of 40.

### 3.5.5 LP Synthesis

The reconstructed residual signal is used to excite the LP synthesis filter in processor **280** to obtain the final speech signal. The transfer function of this filter is equivalent to the one shown in Fig. 2.1 and the filter coefficients are computed as a result of the LSF interpolation in processor **270**. The reconstructed speech is de-emphasized with the same value of $\alpha$ as used in pre-emphasis in processor **130**.

## 3.6 Performance of the Analysis-Synthesis Layer

We simulated the analysis-synthesis (unquantized) layer which is capable of handling pitch multiple and pitch submultiple occurrences. In this section, we will outline several performance issues associated with this system. Subjective evaluation results will be presented as well.

### 3.6.1 Time Asynchrony

The WI method described in this chapter generally does not maintain the time synchrony between the original and the reconstructed speech. This is mainly due to the inexactness of the pitch track estimation and the time-scaling process during the waveform interpolation (processor **230**). Nevertheless, perfect reconstruction in WI is still possible if the following conditions are fulfilled:

1. Perform the CW extraction once per sample point instead of once per subframe (eliminating the CW interpolation and hence, the time-scaling process)
2. Attain an exact phase track $\phi(n)$
3. Obtain an exact value for the initial phase offset $\phi(0)$
4. Constrain the extraction points, i.e., $\epsilon = 0.0$
5. Preserve the DC component in each CW

It should be noted that the requirement of having an exact pitch value at every sample point is very difficult to achieve in practice.

As a result of the time asynchrony in WI, the SNR measurement between the original and the coded speech cannot be used as a fidelity criterion; subjective testing is therefore necessary to evaluate the reconstructed speech quality of WI.

### 3.6.2 Subjective Quality Evaluation

It has been reported that near-transparent speech can be generated from the unquantized WI scheme [19]. In order to verify the accuracy of our analysis-synthesis system, we ran an A-B listening test comparing the speech quality of our implementation with that of the ITU-T 32 kbps G.726 (ADPCM). The speech materials consisted of 23 sentences recorded in clean environment, uttered by 7 male and 6 female speakers. The average length of each sentence was about 2.5 seconds.

For the comparison test, each sentence was processed by our WI unquantized model and the ADPCM. The resulting sentence pairs were randomized and presented to eight non-expert listeners. The overall results are tabulated in Table 3.1.

**Table 3.1** Paired comparison test results between the WI analysis-synthesis layer and the 32 kbps ADPCM

| Preference | Number of votes | Percentage of votes |
|---|---|---|
| Prefer WI | 65 | 39% |
| No preference | 88 | 52% |
| Prefer ADPCM | 15 | 9% |
| Total: | 168 | 100% |

The results shown in Table 3.1 indicated that our unquantized WI implementation outperformed the 32 kbps ADPCM. Whenever a preference was expressed, more than 80% of the time it went to WI. ADPCM was favoured only for 9% of the test utterances. In fact, most WI sentences were even indistinguishable from the originals. These results not only confirmed the accuracy of our implementation, but also showed that the speech quality of WI is not constrained by the model itself.

The above tests were performed at $R_{extr} = 8/frame$. When $R_{extr}$ was set to a higher value (at the expense of higher complexity), there was only a minor improvement in the resulting output quality. However, when a lower $R_{extr}$ value was used, the output speech began to suffer from buzziness which was an indication of an excessive periodicity.

### 3.6.3 Temporal Envelope Variations

As mentioned in Section 1.4, the time envelope of a speech signal is of perceptual importance. It helps to maintain the naturalness and intelligibility of a speech signal.

Nonetheless, we have found in our simulations that WI falls a little short in preserving this envelope, even with unquantized parameters. Undesired envelope variations sometimes occur in reconstructed speech. An example of such occurrences can be seen in Fig. 3.19. In the case of high-energy input speech signals, such envelope variations may cause clipping in the reconstructed speech.

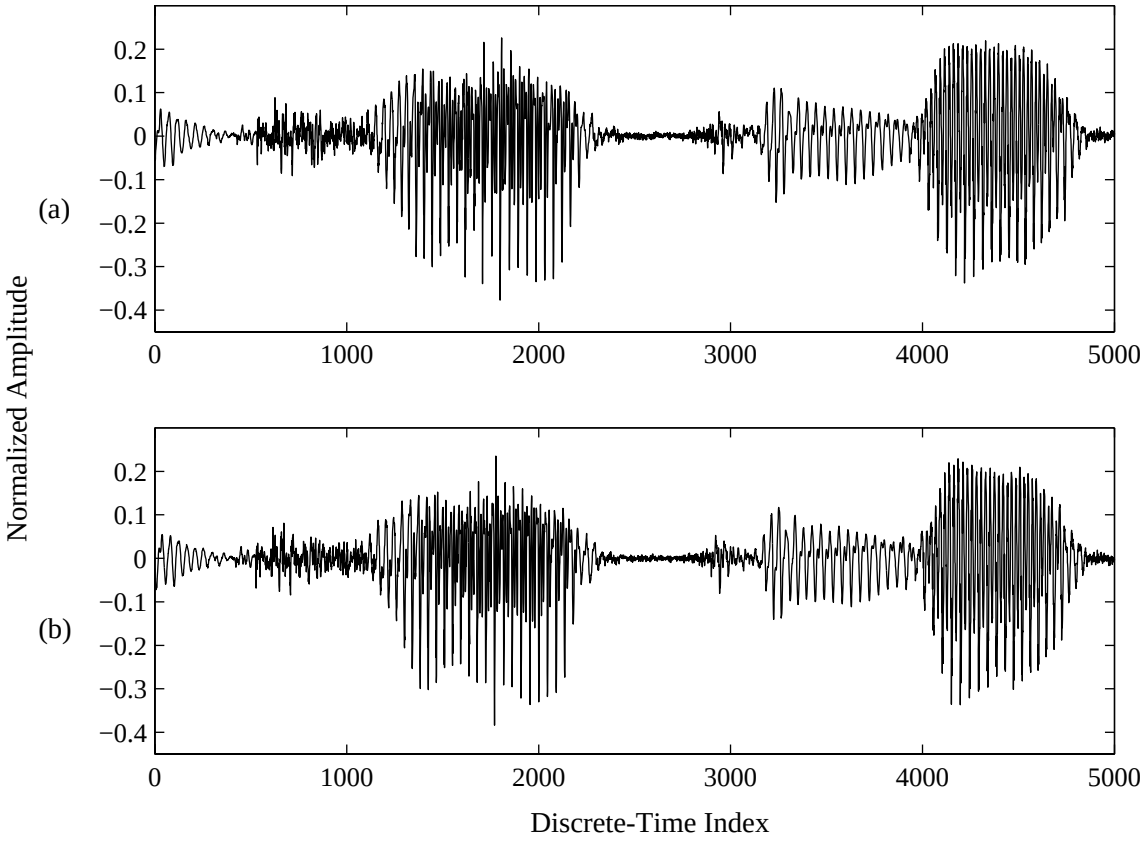


**Fig. 3.19** An example of the time envelope variation caused by the WI method. (a) A speech segment spoken by a female speaker. (b) The same speech segment reconstructed by WI without quantization.

In fact, the time envelope variation problem was first identified in PWI and was found to be caused by the time-varying nature of the LP filter [35]. In other words, linear interpolation in the residual domain does not necessarily correspond to linear interpolation in the speech domain and hence, a smooth time envelope is not guaranteed to be reconstructed.

It was reported in [34] that such envelope variations result in audible warble in

PWI. Nevertheless, this warble distortion seems to be absent in the output of the WI model in spite of the occasional presence of the erratic temporal envelope. Such perceptual improvement of WI over PWI is mainly attributed by the relatively high extraction rate in WI [8]. As more CWs are extracted per frame, the interpolation interval gets shorter which in turn reduces the non-linear effect of the LP filter on the interpolated residual signal.

## 3.7 Variants of the WI Scheme

The conceptual introduction of WI has initiated the development of various WI-based coding schemes; many WI variants (derivatives) have been devised in recent years. In this section, we will discuss two of the most popular derivatives which are (i) the speech-domain analysis/synthesis scheme and (ii) the residual-domain analysis/speech-domain synthesis scheme [9]. Other WI variants will be briefly mentioned at the end.

### 3.7.1 Analysis in Speech + Synthesis in Speech

Our WI scheme carries out the coding in the residual domain. It has been well documented [9, 35, 36, 37] that the WI principles can also be applied directly on speech. In fact, performing the extraction and the interpolation in the speech domain may eliminate the potential envelope variations in the reconstructed speech [35] caused by the time-varying effects of the LP filter (see Section 3.6.3). It may also lead to more efficient coding of the CWs [36, 37].

We simulated this particular scheme by removing all the LP-related processors from our current implementation. They included **110**, **120**, **130**, **270** and **280**. However, preliminary listening experiments showed that the speech quality of the resulting setup contained some roughness (a noisy character or hoarseness), even without quantization. Such distortion was later diagnosed to be caused by the high boundary energy in the extracted CWs, which directly led to audible discontinuities when the CWs were periodically extended in processor **160**. It is possible that a better pitch

[8] The extraction rate in PWI is usually once a frame (50 Hz) which is eight times lower than that of WI.

[9] For the purpose of comparisons, the WI scheme previously discussed in this chapter does both the analysis and synthesis in the residual domain.

resolution (fractional pitch estimation) and/or a better pitch interpolation scheme may mitigate these effects.

One important point to note here is that these discontinuities are not perceptually significant in our original WI scheme. The reasons for this are twofold. First, our CWs are extracted in the residual domain which generally has clear, well-defined pitch pulses and low-power regions in between. Therefore, a CW with low boundary energy can be easily found in the neighborhood of an extraction point in spite of some minor inaccuracies in the CW pitch lengths, inaccuracies which are caused by the non-fractional pitch estimations or the approximations in the linear pitch interpolation. Second, any discontinuities in the reconstructed residual tend to be smoothened out by the LP synthesis filter (processor **280**) when synthesizing the final output. This synthesis filter can act as a lowpass filter attenuating high-frequency noise from the residual signal.

One straightforward solution in enhancing the pitch precision is to upsample the input signal prior to the WI encoding, and to downsample the decoded speech back to obtain the true reconstructed speech. Such upsampling procedure increases the resolution of the speech signal as well as the precision of the pitch estimates and consequently, reduces the boundary energy of the extracted CWs. However, this procedure is associated with a commensurate increase in the coder complexity because the lengths of the CWs are elongated by the upsampling process. Recall from Section 3.4.2 that the amount of computational efforts in WI increases with the CW lengths.

A more efficient way to address this problem is suggested by [9] in which a new extraction procedure is proposed using the maximum-prediction-gain criterion. This procedure not only accurately extracts speech-domain CWs with low boundary energies but also yields a precise (fractional) pitch value for each extraction point and thus alleviates the approximation made in the linear pitch interpolation procedure.

### 3.7.2 Analysis in Residual + Synthesis in Speech

Recall from Fig. 3.14, the spectral envelope represented by the LP coefficients is added to the reconstructed residual signal through the LP synthesis filter in processor **280**. Indeed, it is also possible to add this spectral envelope to the individual CW [9, 19, 30], by transforming the residual-domain CWs to the speech-domain CWs before the waveform interpolation takes place in processor **230**. Figure 3.20 shows a

schematic of such a setup.

Note that Fig. 3.20 is very similar to the decoder structure presented earlier in Fig. 3.14. The key difference is that processors **230** and **260** now operate on the speech-domain CWs. Furthermore, a new processor **290** is created to transform the CWs to the speech domain by adding the formant structures to them. Unlike processor **280** which uses the linear convolution, processor **290** accomplishes this addition by circularly convolving the CW $\{A_k, B_k\}$ with the LP coefficients $\{a_k\}$. The formulations required for this circular convolution are:

$$\begin{aligned} C_k(n) &= \frac{A_k(n)\sum_{m=0}^{N} a_m \cos\left(\frac{2\pi km}{P(n)}\right) + B_k(n)\sum_{m=0}^{N} a_m \sin\left(\frac{2\pi km}{P(n)}\right)}{\left[\sum_{m=0}^{N} a_m \cos\left(\frac{2\pi km}{P(n)}\right)\right]^2 + \left[\sum_{m=0}^{N} a_m \sin\left(\frac{2\pi km}{P(n)}\right)\right]^2} \\ D_k(n) &= \frac{-A_k(n)\sum_{m=0}^{N} a_m \sin\left(\frac{2\pi km}{P(n)}\right) + B_k(n)\sum_{m=0}^{N} a_m \cos\left(\frac{2\pi km}{P(n)}\right)}{\left[\sum_{m=0}^{N} a_m \cos\left(\frac{2\pi km}{P(n)}\right)\right]^2 + \left[\sum_{m=0}^{N} a_m \sin\left(\frac{2\pi km}{P(n)}\right)\right]^2} \end{aligned} \tag{3.35}$$

where $k = 1, 2, \ldots, \lfloor P(n)/2 \rfloor$ and $\{C_k(n), D_k(n)\}$ represent the DTFS coefficients of the speech-domain CW at time instant $n$. $\{a_0, a_1, \ldots, a_N\}$ are the interpolated LP coefficients supplied by processor **270**. Note that $a_0$ is always 1 (see Fig. 2.1).

Although it is not the focus of this section, it is important to know that the inverse of (3.35) also exists:

$$\begin{aligned} A_k(n) &= C_k(n)\sum_{m=0}^{N} a_m \cos\left(\frac{2\pi km}{P(n)}\right) - D_k(n)\sum_{m=0}^{N} a_m \sin\left(\frac{2\pi km}{P(n)}\right) \\ B_k(n) &= C_k(n)\sum_{m=0}^{N} a_m \sin\left(\frac{2\pi km}{P(n)}\right) + D_k(n)\sum_{m=0}^{N} a_m \cos\left(\frac{2\pi km}{P(n)}\right) \end{aligned} \tag{3.36}$$

where $k = 1, 2, \ldots, \lfloor P(n)/2 \rfloor$. A detailed proof of (3.35) and (3.36) can be found in [34]. Thus, (3.35) and (3.36) can be used to respectively add and remove the formant structures from the CWs. They will be particularly useful when dealing with perceptual-weighting processing, as we will see in Section 4.4.3.

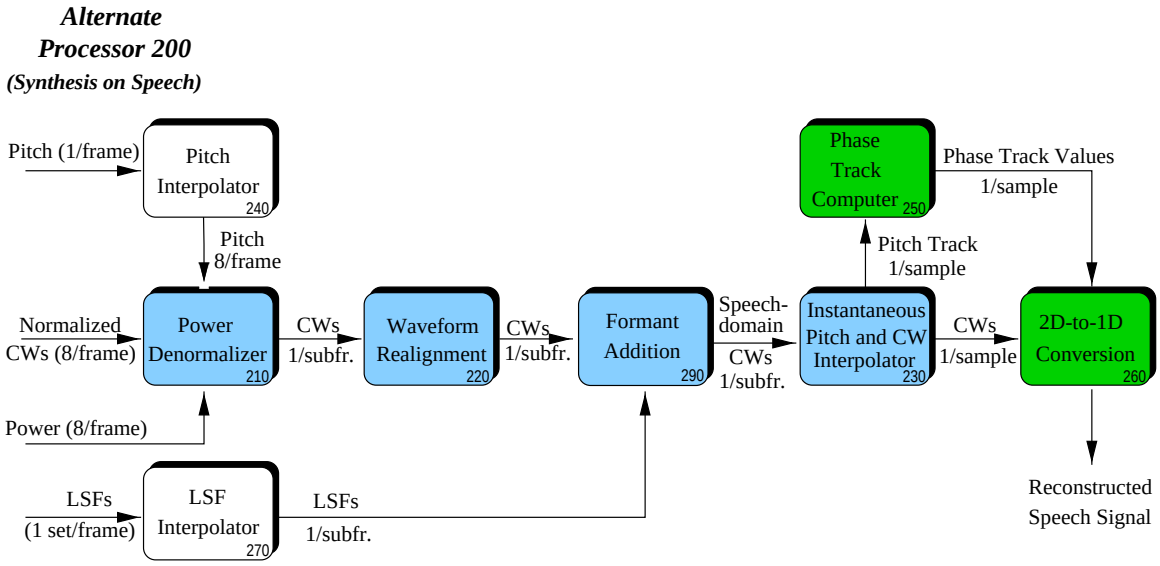


**Fig. 3.20** An alternate WI decoder which does the waveform interpolation and the 2D-to-1D conversion on the speech-domain CWs instead of the residual-domain CWs. The lighter-shaded processors are executed once per subframe while the darker-shaded ones are executed once per sample. The non-shaded ones are computed once per frame.

### Linear Convolution versus Circular Convolution

Experiments were performed to find out how well the results of the circular convolution approximated that of the linear convolution. Specifically, we constructed the following two analysis-synthesis configurations (without quantization):

(A): Using Fig. 3.3 for analysis and Fig. 3.14 for synthesis

(B): Using Fig. 3.3 for analysis and Fig. 3.20 for synthesis

System A is our original WI analysis-synthesis which uses the linear convolution in the LP synthesis. On the other hand, System B is based on the circular convolution. The outputs generated from these two configurations at various extraction rates $R_{extr}$ were compared, both subjectively and objectively. Our test speech was of 25 seconds long and composed of three male and three female speakers.

Table 3.2 tabulates the SNR and the segmental SNR values between the reconstructed speech of the two systems. When $R_{extr} = 8$, both systems yielded almost the same perceptual quality despite slight numerical differences (SNR $\approx$ 18 dB). However,

this SNR decreased slowly as $R_{extr}$ increased. More importantly, as $R_{extr}$ reached beyond 20/frame, a noisy character ("static" noise) started to be perceivable in the output of System B and its quality continued to deteriorate as $R_{extr}$ was increased further. Nonetheless, the speech quality of System A improved progressively with the increasing extraction rate.

**Table 3.2** The SNR measures between the linear and circular convolution for a 25-second speech segment

| Number of | Measured in dB | |
|---|---|---|
| extractions per frame | SNR | Segmental SNR |
| 8 | 18.2 | 17.4 |
| 10 | 18.3 | 17.3 |
| 20 | 17.2 | 16.3 |
| 40 | 16.6 | 15.6 |
| 80 | 16.0 | 14.9 |
| 160 | 15.6 | 14.5 |

In the WI paradigm, the reconstructed speech quality should theoretically increase with the sampling rate of the CW. System A seems to agree with this principle but System B clearly violates it. To investigate the cause of the distortion in System B, we plot the differences between the outputs of the two systems in Fig. 3.21. Note that from Fig. 3.21c that most discrepancies concentrate around the non-periodic regions such as the transient and unvoiced segments. To understand why this phenomenon is happening, it is important to first realize that the key difference between the linear and circular convolutions lies in the initial filter memories. In the circular convolution operation, the filter obtains its initial memory values by effectively filtering the periodic extension (into the past) of the present CW. Whereas in the linear convolution, these memory values are determined by the past filter output samples. Therefore, the majority of the discrepancies between the two convolutions happen in the non-periodic segments where their initial filter memories would differ significantly.

The number of the circular convolutions performed per frame is equal to the number of extractions per frame. Therefore, as the extraction rate increases, the total errors increases and eventually leads to a noisy character. Such distortion is clearly reflected in Fig. 3.21d and Fig. 3.21e for $R_{extr} = 40$ and 160.

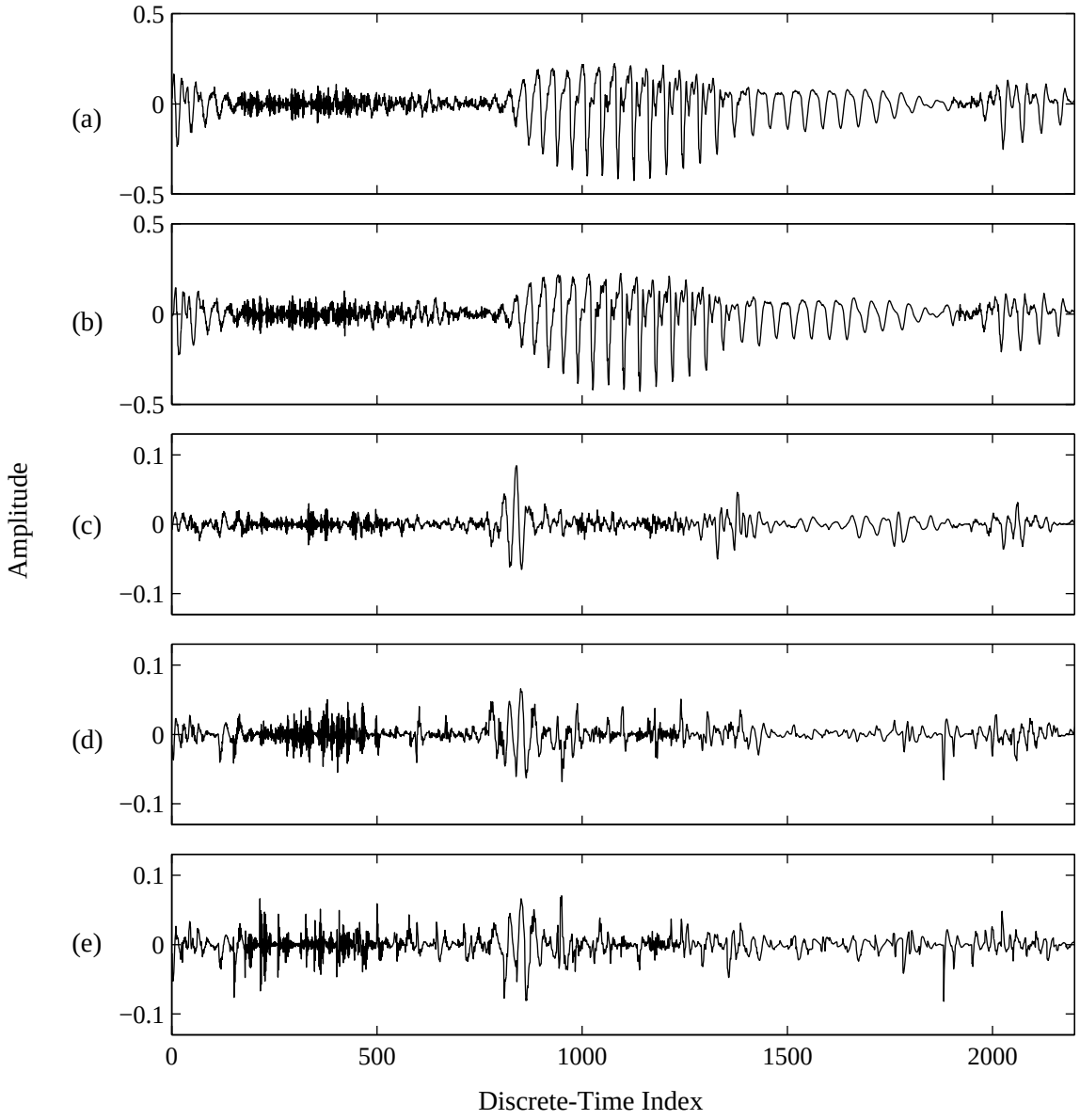


**Fig. 3.21** The discrepancy between the linear and the circular convolutions. (a) A speech segment using the linear convolution at $R_{extr} = 8$. (b) The same segment generated using the linear convolution as well, but at $R_{extr} = 160$. (c) The difference between the outputs of the linear and the circular convolutions. Both operate at $R_{extr} = 8$. (d) Same as above except that the convolutions perform at $R_{extr} = 40$. (e) Same as above except that $R_{extr} = 160$. Note that there is only a subtle difference between (a) and (b) which reinforces the accuracy of our WI scheme based on the linear convolution. The graph (c) confirms that the circular convolution approximates very well to the linear convolution (except at the transitions) when $R_{extr}$ is low. However, the deteriorating effect is evident in (d) and (e) as $R_{extr}$ increases.

### 3.7.3 Other WI Derivatives

In addition to the two derivatives introduced above, there are numerous and varied other WI-based schemes. One of which is the *pitch pulse evolution* model [38, 33] which yields a very high quality of speech at low bit rates. It was based on a combination of the WI and the generalized analysis-by-synthesis paradigms. A hybrid of WI and MBE [39] was developed in [36]. Its speech quality operating at 2.55 kbps reportedly outperformed those of FS1016 and IMBE [40]. Some other WI-based schemes can be found in [41, 42, 43, 44, 22].

## 3.8 Importance of Bandwidth Expansion in WI

Bandwidth expansion can benefit the LP operation by enhancing the stability of the LP filters and reducing the number of cross-overs of the quantized LSFs. In our simulation, we observe that the bandwidth expansion can also be used to improve other aspects of the WI coder. In this section, we will first introduce a phenomenon in LP residual signals known as the *pitch pulse disappearance* [45]. We will then examine how this phenomenon may impact the WI performance and how the bandwidth expansion can help to ease the problems.

#### Pitch pulse disappearance

It is observed that some human speech segments take on sinusoidal form. This happens more often in nasalized sounds since the spectral zeros in these sounds tend to cancel out the energies in the second and third formants. Consequently, there may only be one or two dominant harmonics left in the signal spectrum and this leads to the sinusoidal-like waveforms in the speech signal. These sinusoidal waveforms, which have high short-term correlations, result in high prediction gains in the LP analysis. The high prediction gains, in turn, causes the residual signal to have low energy and the pitch pulses start to disappear. Figure 3.22 illustrates an example of such disappearance.

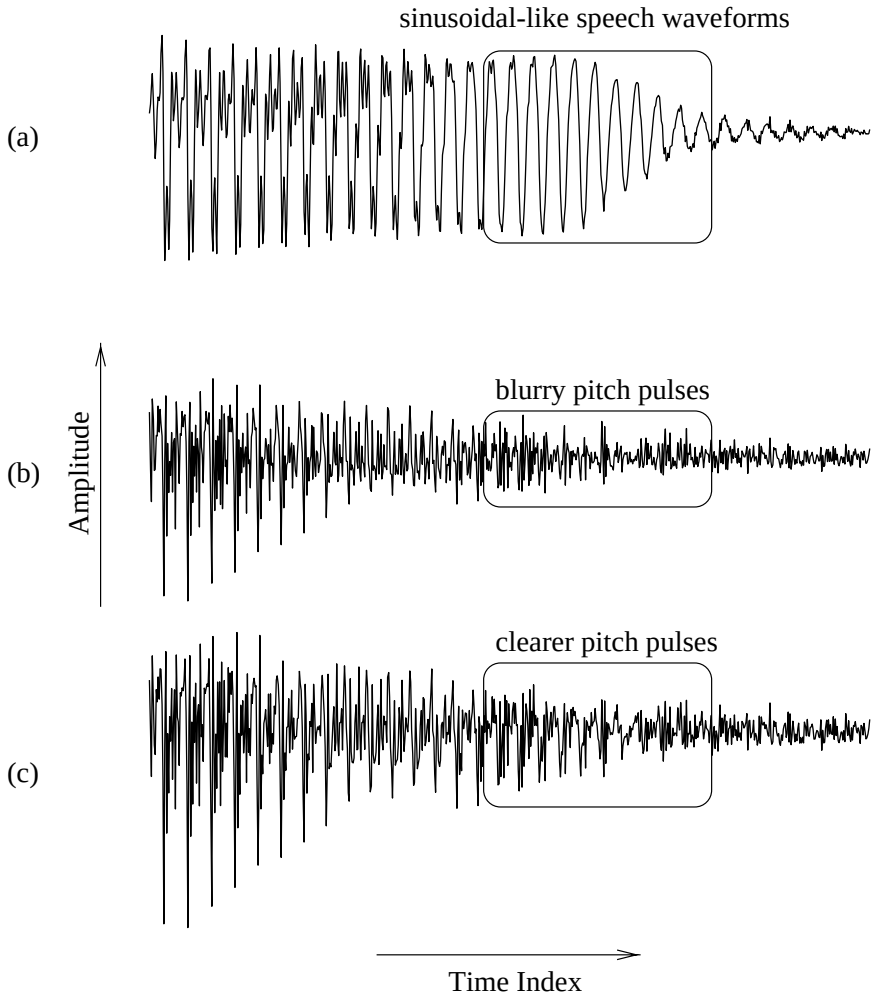


**Fig. 3.22** Illustration of the pitch pulse disappearance. (a) A speech segment that takes on sinusoidal-like shape. (b) The resulting LP residual signal using BW=1.0. The pitch pulses that correspond to the sinusoidal-like speech waveforms are blurry. (c) The resulting LP residual using BW=0.9. The pitch pulses in this case become clearer than the ones in (b).

### Impact of pitch pulse disappearance on WI

The fact that the residual has blurry pitch pulses may actually affect the performance of pitch estimator and also increase the chances of CW misalignment. One solution to this problem is to adjust the bandwidth expansion factor $\gamma$ in the LP analysis. By lowering $\gamma$, the pitch pulses start to reappear in the residual signal. Figure 3.22c shows an example of this reappearance for $\gamma = 0.9$ [10]. Therefore, setting up the correct bandwidth expansion factor is crucial to the performance of WI coder, particularly the CW alignment and the pitch estimation procedures.

Thus, as $\gamma$ is lowered further from unity, the pitch pulses in the residual become clearer. Nonetheless, there is an disadvantage to this. As $\gamma$ decreases, the energy of the residual signal increases which would in turn cause the CW quantizers to operate less efficiently.

## 3.9 Time-Scale Modification Using WI

Although the primary goal of the WI analysis-synthesis model is for speech coding, it can also be readily used to time-scale a speech signal [11].

Time-scale modification in speech signals is a process whereby speech segments are expanded or compressed along the time-axis in a way that the original frequency characteristics are preserved. Thus, a time-scaled speech would be perceived as changes in the rate of articulation while maintaining other speaker-dependent features such as pitch and vocal tract information.

The fact that we need to modify time independently of frequency is a challenging problem; the two dimensions cannot be easily decoupled [46]. However, in the WI analysis-synthesis system, this can be easily accomplished since we have decomposed speech into four independent elements — CWs, the powers, the pitch track and the LPC parameters. To obtain the reconstructed speech on a new time scale, we only have to alter the length of the interpolation interval over which the CWs, the pitch and the LSFs are interpolated (processors **230**, **250** and **270** respectively).

Figure 3.23 shows a speech segment which is speeded up and slowed down using

[10] In practice, $\gamma$ is rarely set to such a low value. A more typical value of $\gamma$ ranges between 0.97671 (60 Hz expansion) to 0.99413 (15 Hz expansion).

[11] WI can also be used to pitch-scale a speech signal although the procedure is not as trivial as in time-scaling [30].

the WI analysis-synthesis layer. Generally, very good performance can be obtained for the WI time-scaling except when the interpolation interval is significantly increased. Especially for low-pitch segments, buzziness starts to be perceivable when the speech is slowed down by more than 50%. This is due to the excessive periodicity caused by the prolonged interpolation interval. Reference [30] suggests to add random phase pitch-synchronously to the REW spectrum to decrease the level of periodicity. The concept of REW will be introduced in the next chapter.

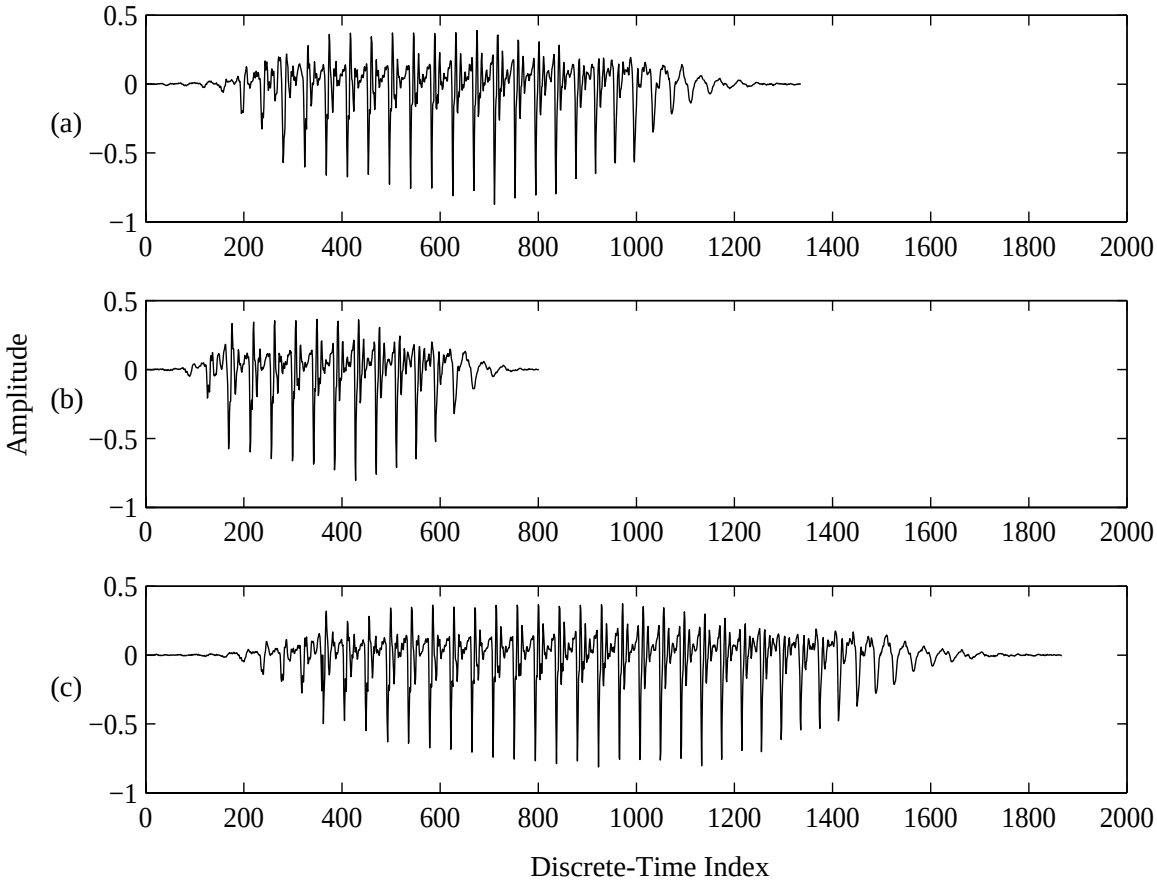


**Fig. 3.23** Time scale modification of a speech segment using WI analysis-synthesis layer. (a) A WI reconstructed speech segment with no time-scaling. (b) The resulting speech segment after the interpolation interval in the synthesis layer is shrunk by a factor of 0.6 (67% increase in the articulation rate). Note that the time-scaling maintains the original pitch track. (c) The resulting speech segment after it is slowed down by 40%. Waveform periodicity becomes quite prominent in this case.

# Chapter 4

# Quantization of the Coder Parameters

The WI analysis-synthesis layer (in the absence of quantization) has been proven to yield virtually transparent speech quality and thus, it provides an excellent foundation for the development of a speech coder. In this chapter, we turn our attention to the quantization layer and present an initial design of a 4 kbps speech coder.

There are four parameters to be transmitted in the WI scheme — LP parameters (LSFs), pitch, power and characteristic waveform. Figure 4.1 shows the processors involved in coding the parameters. It should be noted that these processors are part of processors **300** and **400** [1] shown in Fig. 3.1.

We will first begin this chapter by discussing the coding of the LSFs and the pitch. Next, we will proceed to the quantizations of the power and CWs which require further processing prior to coding. In the last section, we will compare the subjective quality of the WI coder with that of G.729.

## 4.1 LSF Quantization

In our bit budget for the 4 kbps WI coder, we allocate 30 bits in quantizing each set of LSFs and their transmission rate is 50 Hz (one set per frame). The split vector

[1] The analysis-synthesis layer discussed in Chapter 3 is capable of handling pitch multiple and submultiple occurrences. However, we are not yet able to implement such a feature for the quantization layer. As we will see later, this is mainly due to the limitations imposed by low-bit-rate quantization schemes.

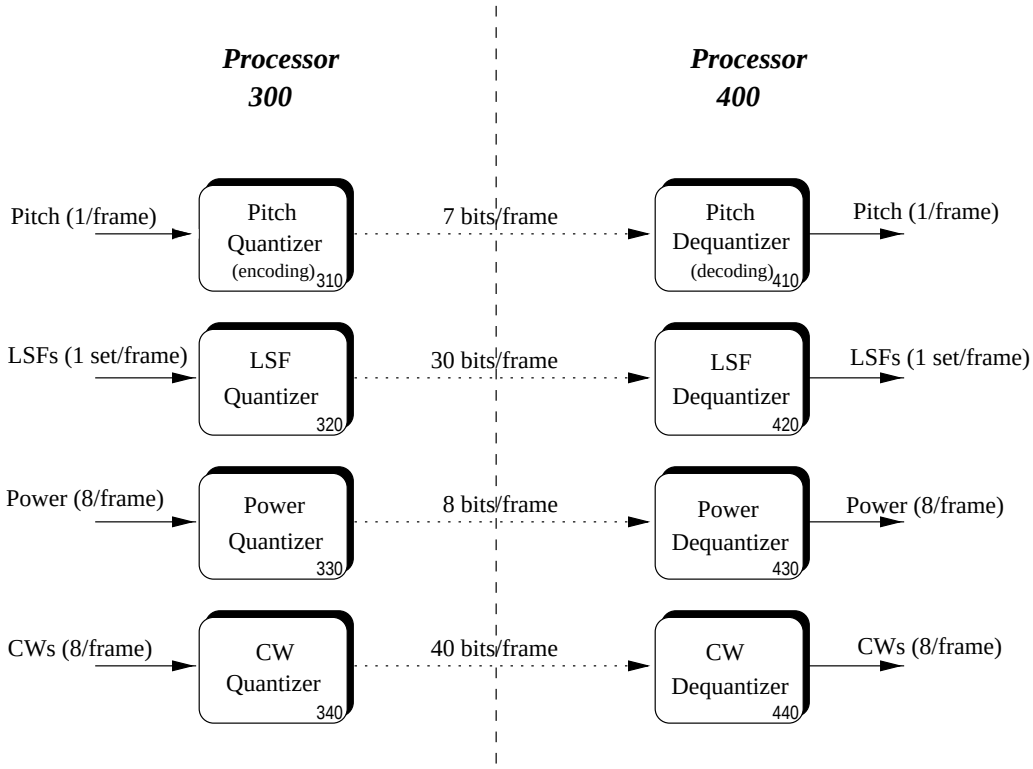


**Fig. 4.1** A block diagram of the WI quantizer. The dotted arrows represent bit-streams. The quantization schemes for the power and the CW are further shown in Fig. 4.2.

quantization technique is employed where a vector of 10 LSFs is divided into three subvectors of dimensions 3, 3 and 4. These subvectors are quantized separately using 10 bits each. The best codebook entry for each subvector is selected based on the minimum weighted distortion measure specified by [23]. In this particular distortion measure, the weight assigned to a given LSF is proportional to the spectral sensitivity and the value of the spectral envelope at that LSF. The codebooks are trained using the conventional Generalized Llyod Algorithm with the Mean-Square-Error (MSE) criterion as the distortion measure. To arrive at the optimal quantized sub-vectors, a sequential search is conducted and only those codewords in the second and the third codebooks that do not result in a cross-over are considered. In this way, the ordering property in the quantized LSFs is preserved and hence, the stability of the LP filter can be guaranteed.

## 4.2 Pitch Quantization (Coding)

The transmission rate for the pitch is 50 Hz (once per frame). Since the pitch estimator **140** yields only integer pitch values, we have a total of 101 possible pitch values ($P_{max} - P_{min} + 1$) which can be encoded with 7 bits. No quantization error is incurred for the pitch [2].

## 4.3 Power Quantization

The quantization and the dequantization of the power are carried out in processors **330** and **430** respectively. Unlike the LSFs, the power requires extra processing prior to its quantization. Figure 4.2 shows the schematic diagrams of **330** and **430**.

As is well known, the logarithm of the signal power is perceptually more relevant than the signal power itself. Therefore, the incoming power values are first transformed to the logarithmic domain. They are then lowpass filtered in **332** and downsampled from a rate of 400 Hz to 100 Hz (twice per frame) [29]. The sampled values are encoded with a non-adaptive differential scalar quantizer using a four-bit codebook. At the receiver, the signal power is decoded and upsampled to a rate of 400 Hz by interpolation in processor **433**. This interpolation is linear and is performed directly on the logarithmic power values. Once the log power contour is upsampled, the signal power can be obtained by the exponential operation.

### 4.3.1 Design of the Lowpass Filter

The aim of the lowpass filter in processor **332** is to prevent spectral aliasing in the downsampling procedure in **333**. Since the downsampling factor in this case is 4 (from 400 Hz to 100 Hz), the cutoff frequency of the filter is required to be 50 Hz, or equivalently 0.25 on a normalized frequency scale. In our implementation, this anti-aliasing filter is a 17-taps linear-phase non-causal FIR filter. Its impulse response, denoted as $h_{Gain}(m)$, is computed by windowing the impulse response of an ideal lowpass filter (cutoff at 50 Hz) with a hamming window of length 17 samples. Finally,

[2] In our implementation, we have chosen not to quantize the pitch. But for extreme low-bit-rate coding, it is possible to quantize the pitch.

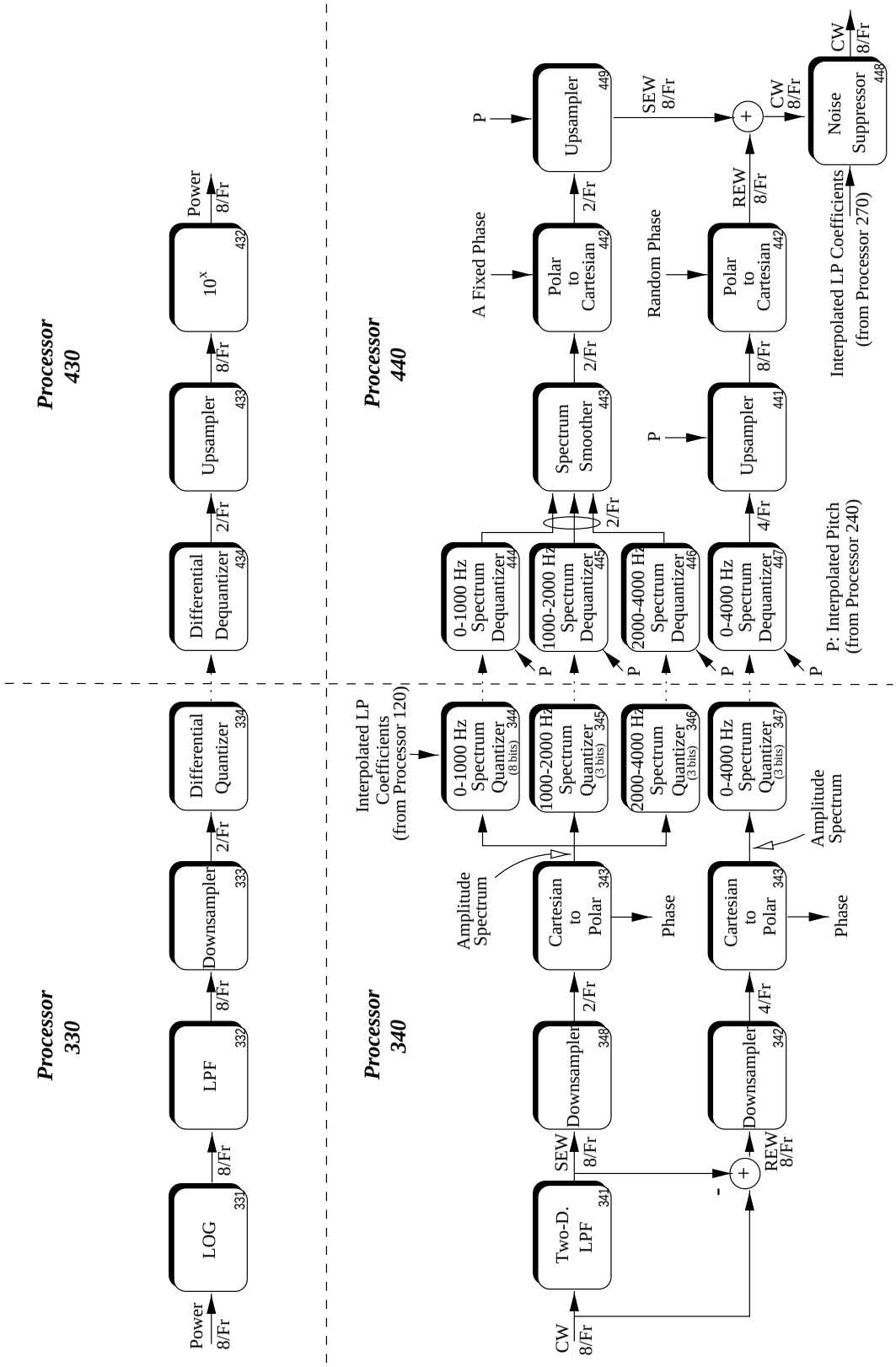


**Fig. 4.2** The schematic diagrams of the quantizers and dequantizers for the power and the CW. The dotted arrows represent the bit-streams.

$h_{Gain}(m)$ can be obtained by normalizing the windowed response such that

$$\sum_{i=-8}^{8} h_{Gain}(i) = 1 \tag{4.1}$$

The magnitude response of $h_{Gain}$ along with its impulse response are plotted in Fig. 4.3.

In processor **332**, the lowpass filtering procedure is performed in the time-domain by linear convolution which can be expressed as:

$$\widetilde{\Psi}_{\log}(kL_{sf}) = \sum_{i=-8}^{8} \Psi_{\log}(kL_{sf} - iL_{sf}) h_{Gain}(i) \qquad k = 0, 1, 2, \ldots \tag{4.2}$$

where $\Psi_{\log}(\cdot)$ and $\widetilde{\Psi}_{\log}(\cdot)$ denote the logarithmic power contour and its lowpass-filtered

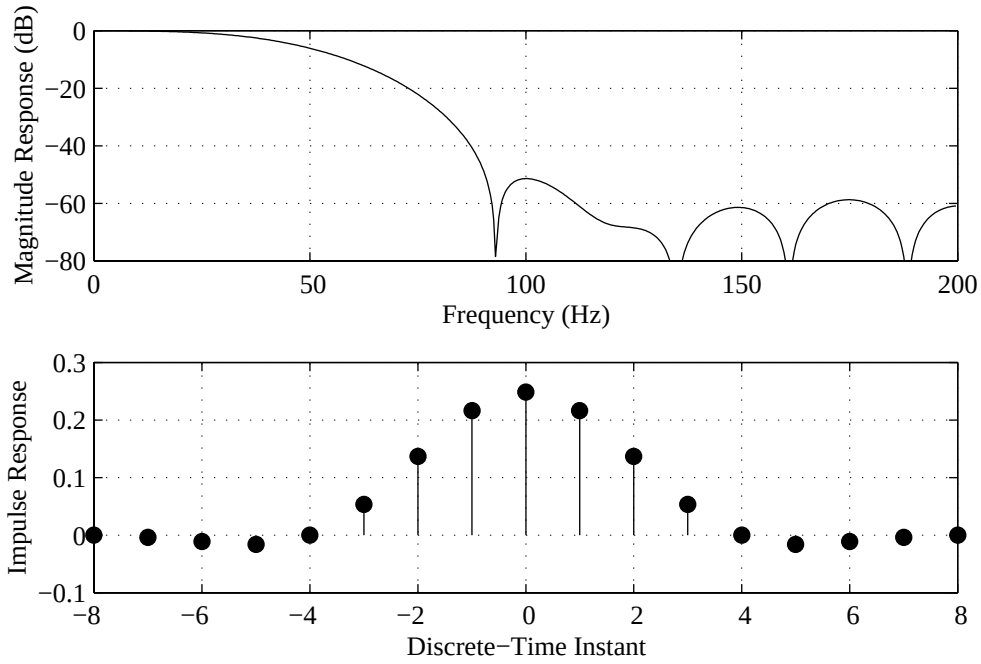


**Fig. 4.3** The characteristics of the anti-aliasing filter used before the power downsampling process (processor **332**). The top diagram shows the magnitude response of the filter. Its impulse response $h_{Gain}(m)$ is shown at the bottom. The filter is designed to have a cut-off frequency at 0.25 based on the normalized frequency scale.

version respectively. Note that the interval $L_{sf}$ is used to index $\Psi_{\log}(\cdot)$ and $\widetilde{\Psi}_{\log}(\cdot)$. This is consistent with the fact that the power values arrive at the quantization layer at every subframe interval.

The non-casual nature of the filter suggests that some look-ahead samples are required for the convolution process. If one examines (4.2) closely, the convolution actually requires eight samples from the future frame and eight samples from the past frame, in order to calculate the filtered values $\widetilde{\Psi}_{\log}$ in the current frame. This translates into one frame of algorithmic delay (20 ms) in the coder. The necessities of the look-past and look-ahead samples are further depicted in Fig. 4.4.

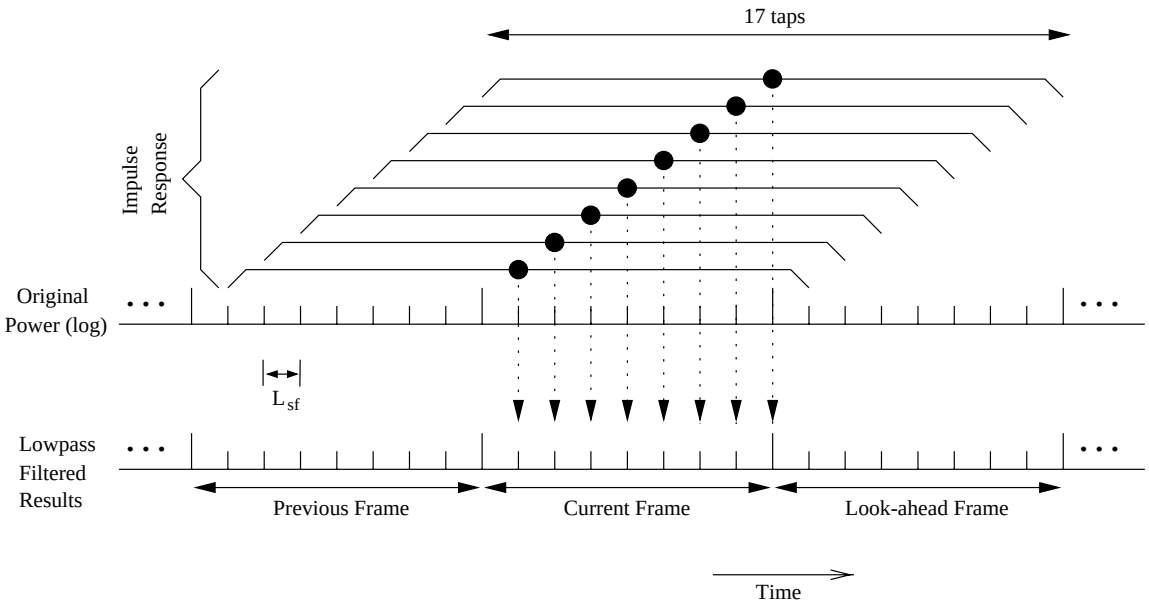


**Fig. 4.4** The convolution procedure for the lowpass filtering of the power contour. Each mark on the first time scale represents a logarithmic power value $\Psi_{\log}$. The marks on the bottom scale represent the filtered values $\widetilde{\Psi}_{\log}$. The diagram brackets the power values required to compute each filtering output $\widetilde{\Psi}_{\log}$ in the current frame.

*Remarks on the power quantization*

- As can be seen in Fig. 4.4, if a more accurate FIR filter is employed (i.e., more taps), more look-ahead samples will be required as well. This becomes a trade-off between the filter accuracy and the coder delay.

- If the extraction rate of the coder is allowed to increase, $\Psi_{\log}(\cdot)$ will have a higher time resolution. Hence, a filter with more taps can be used without incurring extra coder delay. Since an increase in the extraction rate is associated with an increase in the coder complexity, the filter accuracy and the coder complexity form a trade-off.

- Because of the downsampling process in **333**, only two $\widetilde{\Psi}_{\log}$ values are actually transmitted in each frame. In other words, processor **332** is required to compute only two filtered values per frame instead of eight, resulting in computational savings.

## 4.4 CW Quantization

In this section, we will discuss the CW quantizer and the dequantizer which correspond to processors **340** and **440** respectively. Figure 4.2 illustrates the schematics of these two processors. Similar to the power, the CW requires extra processing prior to its quantization. Specifically, each CW is decomposed into two separate waveforms which are quantized separately. The motivation and the details of this decomposition will be examined in the following subsections.

### 4.4.1 SEW-REW Decomposition

At first sight, it appears that an accurate representation of the CWs (an evolving surface) may require a very high transmission rate, particularly for the unvoiced segments which possess a very high information rate. Fortunately, not all of the information contained in the surface is perceptually relevant to human ears. As was revealed in Section 1.4, the human perception of voiced and unvoiced speech differs substantially. This suggests that it is also possible to exploit such differences in the CWs and to quantize them in a perceptually accurate fashion.

Rather than adopting a conventional voiced/unvoiced classifier, [29, 18, 19] propose a novel decomposition technique in which each CW is separated into two components prior to quantization. They are a slowly evolving waveform (SEW) and a rapidly evolving waveform (REW), representing the periodic and noise components of the signal respectively. By taking advantage of the differences in human perception

of these two waveforms, a high coding efficiency can be achieved by quantizing them separately.

The SEW is formed by lowpass filtering the CW evolving surface along the time axis in processor **341** and the REW can be found by subtracting the SEW from the CW. For voiced speech, the SEW and REW represent respectively a shaped pulse-like waveform and a noise component. Due to the presence of the periodicity in voiced regions, the SEW generally has a much higher energy level than the REW. Conversely, for unvoiced speech where the signal evolves quite rapidly and exhibits no apparent periodicity, the decomposition distributes most of the CW energy to the REW. Figure 4.5 illustrates an example of a SEW and a REW surfaces decomposed from the CW surface previously shown in Fig. 3.13.

To avoid introducing extra coder delay, the lowpass filter in processor **341** uses the same number of taps as in $h_{Gain}$ — 17 taps. It is also linear-phase and non-causal. However, this filter requires a much sharper cutoff frequency — 25 Hz or equivalently, 0.125 on the normalized scale. Its impulse response, denoted as $h_{CW}(m)$, is derived in the same way as in $h_{Gain}$. Figure 4.6 plots the magnitude response of $h_{CW}(m)$ as well as its impulse response. Note that the frequency response has a very gradual roll-off (a long transition band). This is mainly because the FIR filter has only 17 coefficients. One can increase the number of taps to achieve a more accurate filter, but at the expense of introducing additional algorithmic delay.

Because the DTFS operation is a linear transformation, lowpass filtering the CWs in the time-domain is equivalent to lowpass filtering their DTFS coefficients. For this reason, processor **341** performs the filtering directly on the $A_k$ and $B_k$ coefficients (for all $k$). Specifically, to compute the lowpass filtered CW at index $n$, we can use the following formulae:

$$\left.\begin{aligned} \widetilde{A_k}(n) &= \sum_{i=-8}^{8} A_k(n - iL_{sf})h_{CW}(i) \\ \widetilde{B_k}(n) &= \sum_{i=-8}^{8} B_k(n - iL_{sf})h_{CW}(i) \end{aligned}\right\} \quad \text{for} \quad k = 1, 2, \ldots, \lfloor P(n)/2 \rfloor \tag{4.3}$$

Since the CWs are at a rate of 400 Hz (every subframe interval), time index $n$ must be a multiple of $L_{sf}$. The $\widetilde{A_k}(n)$ and $\widetilde{B_k}(n)$ are the DTFS coefficients of the lowpass filtered CW (SEW) at index $n$.

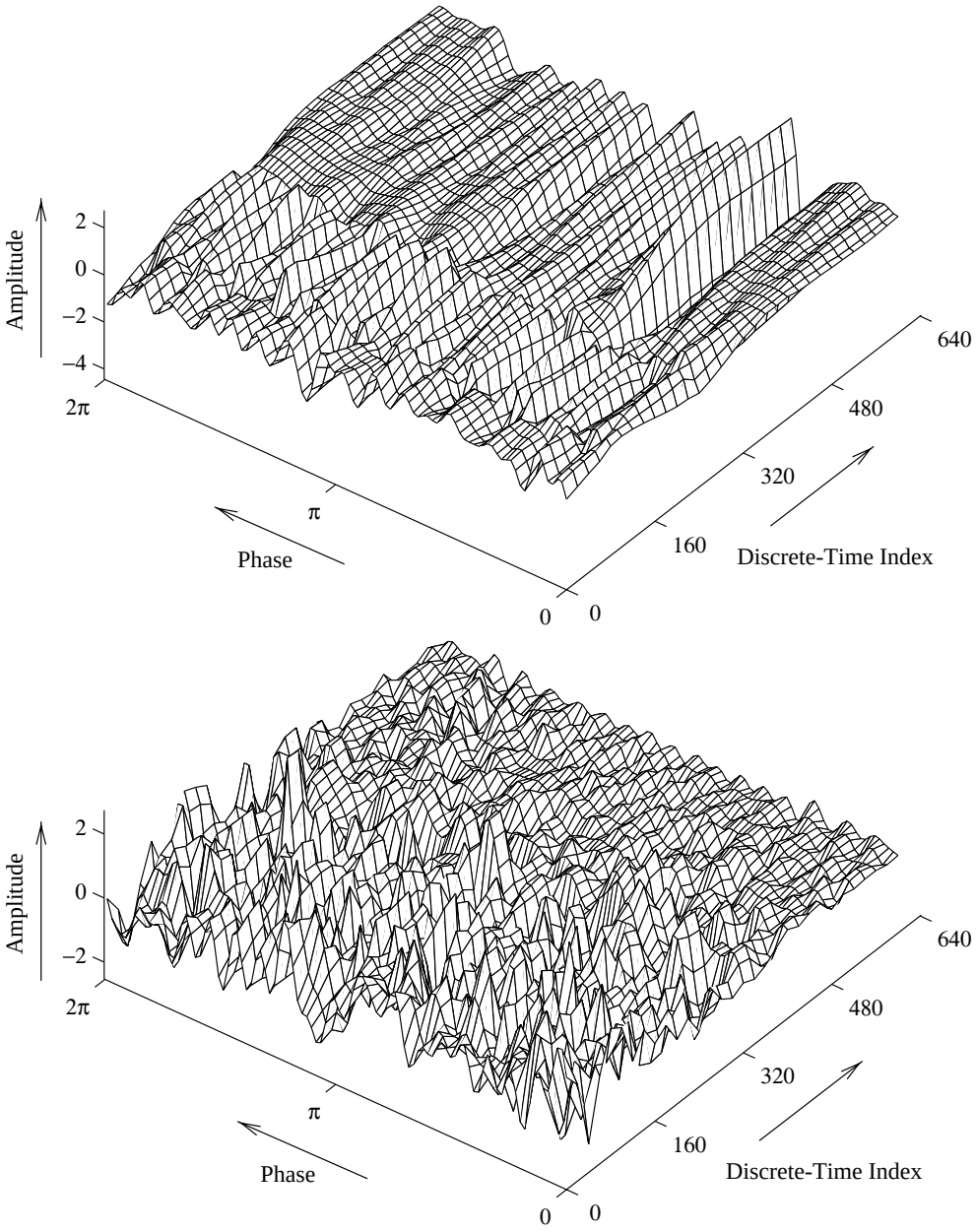


**Fig. 4.5** The SEW (top) and the REW surfaces (bottom) for the segment shown in Fig. 3.13. The cutoff frequency of the lowpass filter is about 25 Hz. The dominance of the REW during unvoiced region and the dominance of the SEW during voiced region are clearly visible.

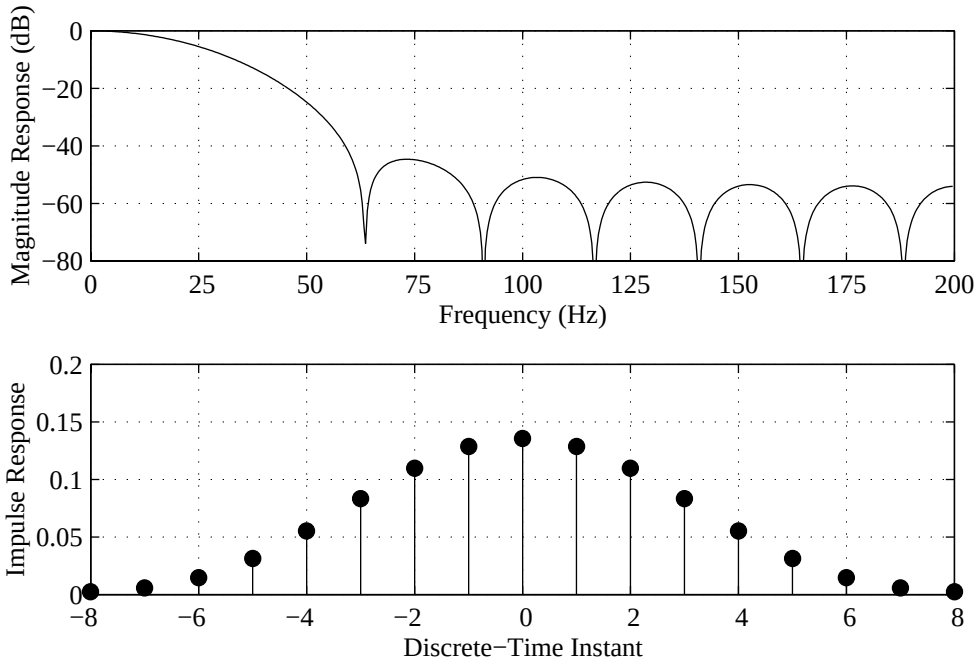


**Fig. 4.6** The characteristics of the lowpass filter used in the SEW-REW decomposition. The top diagram shows the magnitude response of the filter $h_{CW}(m)$ and its impulse response is shown at the bottom. The filter is designed to have a cut-off frequency at 0.125 on a normalized frequency scale.

Nevertheless, the dimension of the CW varies with the pitch. To facilitate filtering, the same techniques as in Section 3.4.5 can be used to time-stretch or time-contract the CWs so that all the CWs within the filtering window can have the same length prior to the filtering operation. Recall that the former operation corresponds to zero-padding in the DTFS domain and the latter corresponds to spectral truncation followed by a power adjustment (to compensate for power loss). The entire filtering operation along with these time-scaling operations is best illustrated by an example: Figure 4.7 shows the filtering operation for the $A_k$ coefficients for a single time instant. The same filtering procedures are applied to the $B_k$ coefficients.

*Remarks on the SEW-REW decomposition*

- As illustrated in Fig. 4.5, the energy of the REW clearly dominates in the unvoiced region whereas the SEW dominates in the voiced region. In fact, one

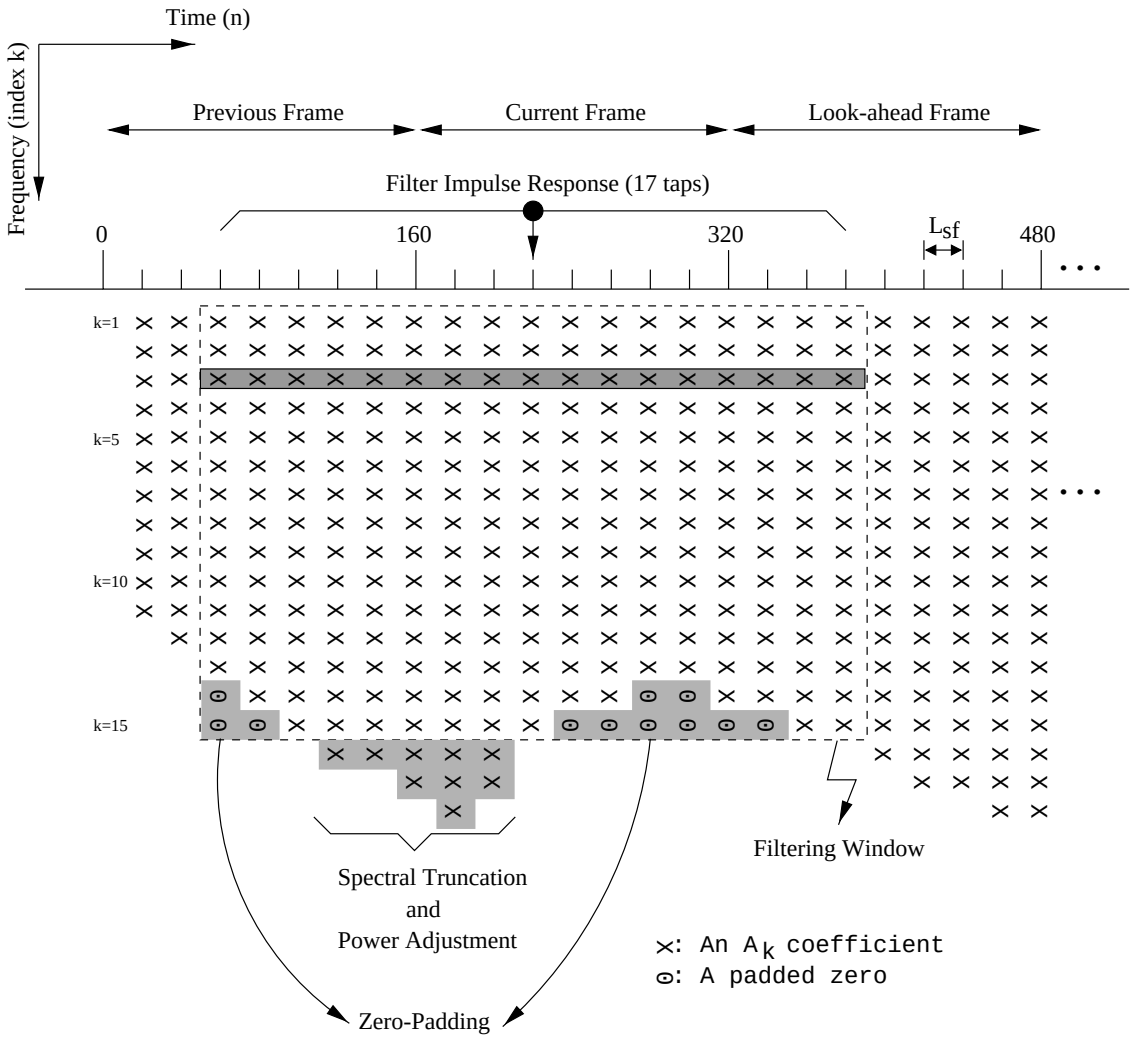


**Fig. 4.7** The lowpass filtering operation for the SEW-REW decomposition. The diagram illustrates the $A_k$ coefficients for 24 successive CWs spanning three frames. It focuses on computing the filtered CW at $n = 220$. Since the original CW at $n = 220$ has a length $k = 15$, all the other CWs within the filtering window (shown in dashed lines) are required to have this length before the filtering can begin. The shorter CWs that have lengths $< 15$ are extended by zero-padding. On the other hand, the longer ones ($> 15$) are shrunk by spectral truncation followed by power adjustment. Afterwards, the filtering procedure can begin and proceed on a $k$-by-$k$ fashion (row-by-row). The horizontal shaded strip shows the coefficients that are involved in the filtering for $k = 3$ and the resulting lowpass value from this particular filtering will be $\widetilde{A_3}(220)$.

can make use of this feature to obtain a rough estimate on voicing information (voiced/unvoiced detector). The degree of voicing should be roughly proportional to the SEW/CW energy ratio (or inversely proportional to the REW/CW ratio).

### 4.4.2 REW Quantization

In this section, we focus on quantizing the REWs in a perceptually accurate manner. We first begin by listing three important conclusions from the experiments conducted in [18]:

1. Little degradation in speech quality is heard if the phase spectrum of the REW is replaced by a random phase spectrum.

2. No deterioration is noticeable in the resulting speech if each REW amplitude spectrum is severely smoothened by a square window of 1000 Hz.

3. Very little audible distortion is produced if the REW amplitude spectrum is averaged over all REWs within a 5 ms interval.

The first finding concludes that the phase spectrum of the REW carries little perceptual information and should not be transmitted at low bit rates. The second and the third imply that the time resolution of the REW amplitude spectrum is far more important than its frequency resolution. In other words, the REW requires a high update rate but a coarse quantization technique.

To exploit these findings, processor **342** downsamples the incoming REW to a rate of 200 Hz which agrees with the time resolution suggested by the third finding (5 ms interval). Each downsampled REW is then converted to its polar notation where the phase spectrum is completely discarded. The amplitude spectrum is vector quantized using a codebook of eight entries. Such a small codebook is used because a rough description of the REW spectrum is sufficient to produce a good quality according to the second finding. The resulting vector index is transmitted and the overall bit consumption by REW is $200 \times 3 = 600$ bits/s $= 12$ bits/frame.

At the receiver, the REW spectra are decoded and upsampled by a factor of 2 in **441**, from a rate of 200 Hz to 400 Hz (making its sampling rate identical to that of the CW). This means that a new spectrum is inserted after every received

spectrum. These new spectra can be filled by linear interpolating the adjacent spectra or by choosing the previously received spectrum. Informal experiments indicated that there was no perceptual difference in the outputs of the two methods. Finally, each of the upsampled REW amplitude spectrum is combined with a random phase spectrum and then transformed back to its rectangular coordinates. The values in the random phase spectra are independent and uniformly distributed in $[0, 2\pi)$. Note that the random phase spectra are added to the REWs on a per-subframe basis.

### REW spectrum codebook design and search

The dimension of the REW amplitude spectrum is proportional to the pitch period. Consequently, the spectrum has a variable dimension and it must be described with an appropriate variable dimension vector quantizer (VDVQ). Note that in our implementation, the pitch is allowed to vary from 20 to 120 resulting in 10 to 60 harmonics in the REW spectrum (the DC component is excluded).

To tackle this VDVQ issue, the REW codebook is designed by a method called dimension conversion vector quantization (DCVQ) [5]. It is based on the assumption that the generation of a variable-dimension vector is as a result of a uniform sampling of another vector with a fixed and large dimension. This technique works as follows.

Prior to the codebook training, each REW spectrum in the training set is first bandlimited interpolated to a fixed dimension vector. A natural choice for the dimension of this vector is the maximum number of harmonics in the spectrum (60 harmonics in our case). After all the training vectors are converted to the same dimension, the conventional GLA technique is applied to train the codebook. Thus, the resulting trained codebook will have a uniform dimension of 60.

When encoding a spectrum in **347**, the codebook is first subsampled to match the length of the given spectrum. Then the best match entry is found based on the minimum MSE criterion and its codebook index is transmitted to the receiver. Note that one can also use the perceptually-weighted version of this error criterion to search for the best entry (more on the perceptual weighting will be seen in Section 4.4.3). However, this would hardly make a difference in the final codeword selection because the size of the REW codebook is extremely small.

Upon receipt of the codebook index in processor **447**, the REW spectrum is reconstructed by subsampling the quantized spectrum. The subsampling factor depends

on the pitch information provided by processor **240**; the pitch determines the number of harmonics in the spectrum.

As mentioned before, the size of the REW codebook is only eight. The spectral shapes in the trained codebook are therefore severely smoothened and can be well approximated by low-order polynomials [20]. Such polynomial representations would considerably reduce the amount of memory consumed by the codebook since only a few polynomial coefficients instead of an entire spectrum (60 harmonics) are stored for each codeword. The fitting of a spectrum to a polynomial is accomplished in a least-square sense and it is found that a fifth order polynomial is adequate for representing a REW spectrum.

### More on the random phase addition

As described, the phase of the REW spectrum is being reset to a new random phase spectrum once per subframe interval (400 Hz). In the WI paradigm, the purpose of this random phase is to remove the correlation between successive REWs. An excessive correlation between the REWs would lead to buzziness in the output speech, particularly for unvoiced segments.

In fact, it is theoretically more appropriate to add this random phase to the REW on a pitch-synchronous basis rather than on a per-subframe basis. Recall from the extraction procedure in processor **160**, for non-periodic signals, the correlation between two successive CWs vanishes when they are separated by one CW length or more (at shorter separations they overlap). By adding the random phase pitch-synchronously to the REW, we ensure that this correlation vanishes "at the right time" (not too often and not too seldom). Our experiments confirmed that if the random phase is being added too frequently (e.g., 8000 Hz), it would introduce a noisy character in the reconstructed speech. Conversely, if the random phase is being added too infrequently (e.g., 50 Hz), the resulting speech will sound buzzy.

To examine how much perceptual improvement can this idea bring to our coder, we modified our existing implementation such that the random phase is added to the REW pitch-synchronously. The improvements were surprisingly minor. However, such a technique can be useful in the time-scale modifications since it can eliminate the buzzy quality when the speech is significantly elongated (see Section 3.9).

### 4.4.3 SEW Quantization

Now, we turn our focus to the quantization of the SEW. Because the cutoff frequency of the decomposition filter is only about 25 Hz, the SEW has a very small evolution bandwidth. This suggests that the signal can be downsampled from 400 Hz to about 50 Hz (the Nyquist rate). However, it is advantageous to downsample it to a higher rate — 100 Hz (two SEWs per frame). This is to compensate the inaccuracy of the decomposition filter in processor **341**. As mentioned in Section 4.4.1, the filter has only 17 taps to handle such a sharp cut-off frequency.

Each downsampled SEW is converted to its polar notation where the phase spectrum is discarded. The amplitude spectrum is split into three non-overlapping subbands: 0–1000 Hz, 1000–2000 Hz and 2000–4000 Hz. The subbands are quantized separately where the baseband is quantized using 8 bits and the remaining two subbands use 3 bits each. Such bit-allocation scheme is to accommodate the better resolving capability of the human ear for lower frequencies [21]. The overall bit consumption for each SEW is therefore 14 bits and the transmission rate is $100 \times 14 = 1400$ bits/s $= 28$ bits/frame.

At the receiver, after the subbands are decoded and combined, linear interpolation technique is employed to smoothen the combined spectrum at the subband boundaries (i.e., at 1000 Hz and 2000 Hz) in processor **443**. An abrupt change or a significant discontinuity in the spectrum may lead to reverberation in the reconstructed output.

After the amplitude spectrum is reconstructed and smoothened, it is attached to a fixed phase spectrum and converted back to the rectangular coordinates. This fixed phase spectrum is drawn from a voiced segment generated by a high-pitched (more harmonics) male speaker which can offer more harmonics than a low-pitch speaker.

Afterwards, the SEWs are upsampled in processor **449** from a rate of 100 Hz to 400 Hz (same rate as the reconstructed REWs). Since the SEWs may be of different dimensions (different number of $a_k$ and $b_k$ coefficients), the procedures outlined in Scenario 2 in Section 3.5.2 can be adopted for this upsampling process.

#### SEW codebook design and search

Like the REW, the dimension of the SEW varies with the pitch. Therefore, the SEW codebook design and the search procedures are very similar to those of the REW.

The SEW amplitude spectra in the training set are first bandlimited interpolated

to a length of 60 which is the maximum number of harmonics that a SEW can have ($P_{max} \div 2 = 60$). Each training vector is then split into three subbands. The first 25% of the harmonics (15 harmonics) go to the baseband. The next 25% go to the second subband and the remaining harmonics belong to the last subband which corresponds to the frequency range 2000–4000 Hz. The GLA technique is applied to separately train the subband codebooks.

As for the codebook searching procedures, they are identical to the REW's except that the searching for the first subband is based on the perceptually weighted error criterion. This criterion is widely used in CELP-based coders and will be further discussed in the next subsection.

### Perceptually Weighted Error Criterion

In processor **344**, the quantized baseband SEW spectrum is selected from the codebook by minimizing the perceptually weighted error between the original and the quantized spectra. The perceptual weighting is derived from the formant structure of the speech signal in such a way that more quantization noise is allowed in the formant regions than in the valleys between formants. This is to exploit the spectral masking property in our human auditory system as previously described in Section 1.4. Since our CW is defined in the residual domain, the weighting can be absorbed into the synthesis filter [25]:

$$H_w(z) = \frac{1}{1 - \sum_{k=1}^{N} a_k {\gamma_w}^k z^{-k}} \qquad 0 < \gamma_w \leq 1 \tag{4.4}$$

where $\gamma_w$ is the weighting factor and is typically set to 0.8. Further, the $a_k$ are the unquantized interpolated LP coefficients from processor **120**.

Thus, the perceptually weighted error spectrum in **344** can be obtained by multiplying the amplitude response $|H_w(z)|$ by the error spectrum, which is the square of the difference between the original and the quantized spectra.

*Remarks on the SEW quantization*

- Figure 4.8 illustrates three examples on how the SEW amplitude quantization and the fixed phase addition affect the shapes of the SEWs. It is clear the

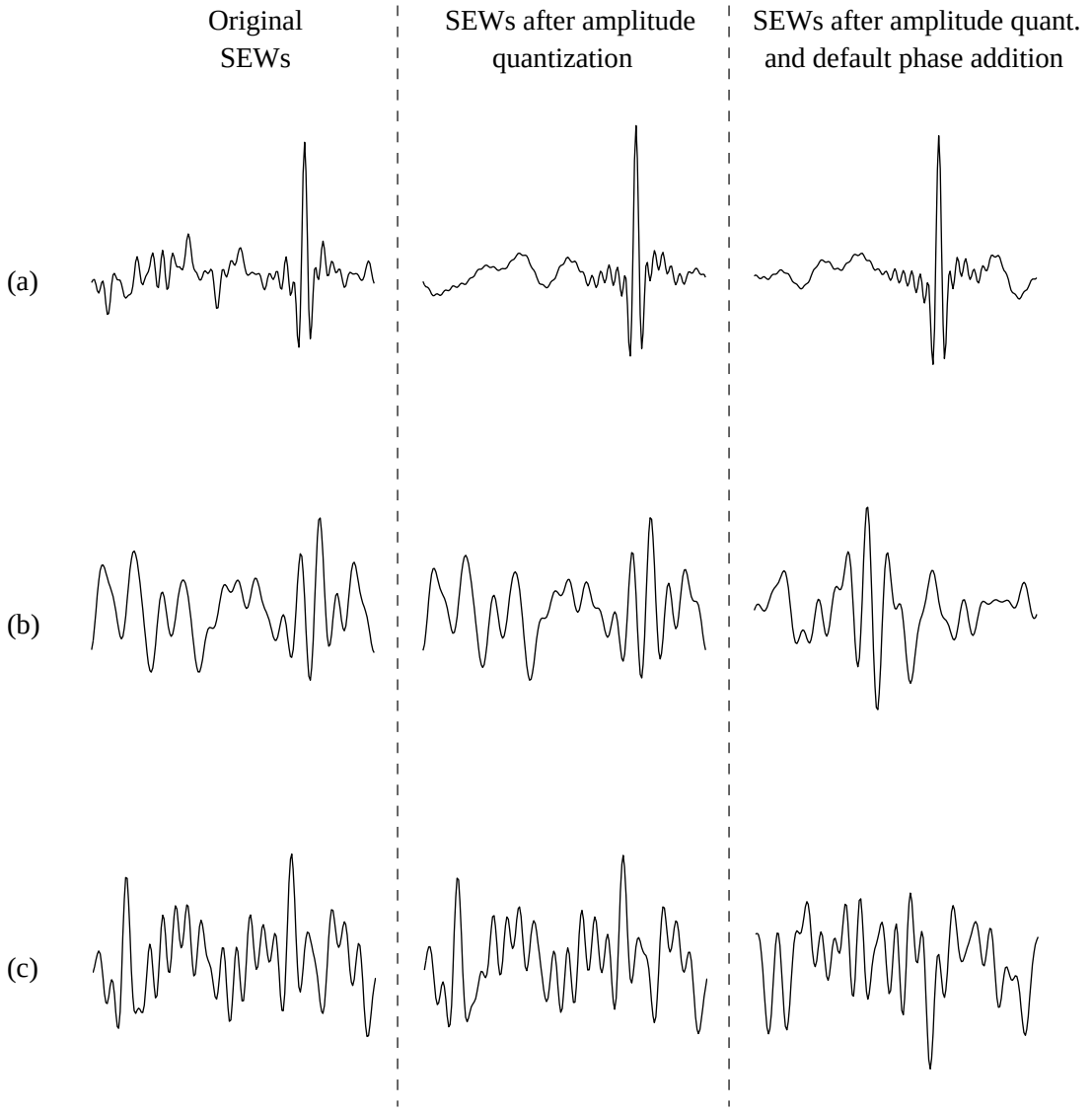


**Fig. 4.8** Quantization of the SEWs. (a) A SEW with a distinct pitch pulse. (b) A SEW extracted from an unvoiced segment. (c) A SEW extracted from a pitch doubling segment; it has two pitch cycles.

default phase may shift the pitch pulse location; therefore, a re-alignment procedure is necessary to ensure that the reconstructed CWs are properly aligned (processor **220**). Also note that the default phase tends to convert the SEWs of different lengths into one distinct pitch pulse (with a variable amount of phase dispersion). This is further depicted in Fig. 4.8b where a noise-like SEW extracted from an unvoiced segment is transformed into a single pitch pulse. This feature, however, creates distortions in pitch doubling or tripling segments since these SEWs have more than one pulse. An example of which is shown in Fig. 4.8c. In fact, it is this default phase that hinders the implementation of the pitch (sub)multiple feature in the quantization layer.

- To further enhance the perceptual quality of the coder, SEWs from the silent and unvoiced regions should be excluded from the quantization training set. This is because SEWs are dedicated to represent the slowly evolving components of the speech signal which are mainly in the voiced regions.

### 4.4.4 CW Reconstruction and Coding Noise Suppression

After both the SEWs and the REWs are upsampled to a rate of 400 Hz, the reconstructed CWs can be obtained by simply adding the two together. This is immediately followed by a noise suppression process [30] in **448**. Similar to a conventional formant postfilter [25], this process enhances the subjective quality of the reconstructed speech signal contaminated by the coding noise created by quantizing the SEWs and the REWs. It emphasizes the important frequency components of the noisy CW spectrum and attenuates the others. The goal is to carry this out without introducing unacceptable spectral distortion to the final speech quality.

The noise suppressor is implemented by filtering the CW with an all-pole and an all-zero filters in cascade. The all-pole filter is the same as the LP synthesis filter shown in Fig. 2.1 except the coefficients are being bandwidth-expanded by a factor of $\gamma_p$. On the other hand, the all-zero filter is defined to be the same as the LP analysis filter (Fig. 2.2) with the coefficients bandwidth-expanded by a factor of $\gamma_z$. So, the

transfer function of this pole-zero filter can be written as:

$$H_{nr}(z) = \frac{1 - \sum_{k=1}^{N} a_k {\gamma_z}^k z^{-k}}{1 - \sum_{k=1}^{N} a_k {\gamma_p}^k z^{-k}} \qquad 0 < \gamma_z < \gamma_p < 1 \tag{4.5}$$

Since $\gamma_z < \gamma_p$ and both parameters are between 0 and 1, the poles of the LP synthesis filter are being radially shifted inward in $H_{nr}(z)$ and the zeros of the LP analysis filter are also being radially shifted inward but further. This effectively suppresses the noise in the valleys of the spectrum and hence the overall coding noise level can be reduced. However, special attention should be paid when choosing these two parameter values. Our experience is that as $\gamma_z$ decreases further from $\gamma_p$, the coding noise in the reconstructed speech becomes less perceivable but at the expense of increasingly muffling. The values of $\gamma_p = 0.9$ and $\gamma_z = 0.85$ seem to provide a good trade-off.

Since each CW can be regarded as a periodic waveform, the pole-zero filtering can be conveniently carried out by means of circular convolution (see Section 3.7.2). Specifically, the output of the all-pole filter can be derived from (3.35), with $\{a_k\}$ substituted by $\{a_k \gamma_p^k\}$. Similarly for the output of the all-zero filter, we can employ (3.36) with $\{a_k\}$ replaced by $\{a_k \gamma_z^k\}$. Note that $\{a_k\}$ are supplied by processor **270**. Although the numerical difference between the linear and the circular convolutions has not yet been studied in this context, preliminary listening results indicate that this difference is barely audible.

Generally speaking, the noise suppressor has a subtle but positive effect on the perceived speech quality. With the correct choices of $\gamma_p$ and $\gamma_z$, the noise suppressor never adds any distortion, but in many cases can reduce the perceptual impact of the REW and the SEW coding noise. Since the improvement comes at no bit rate penalty, the inclusion of the noise suppressor is certainly an asset to the WI coder.

The resulting CW may no longer have a unit power after the noise suppression process. This however would not affect the final power of the speech signal since the power denormalizer **210** in the synthesis stage ensures that the CWs have the correct powers before they are interpolated and synthesized.

## 4.5 Performance Evaluations

A WI coder operating at 4.25 kbps has been designed and simulated. Table 4.1 gives the bit-allocation scheme of the coder. In the following subsections, we will provide the subjective test results of the coder and outline some of the potential problems associated with our quantization schemes. The overall coder delay will also be discussed.

**Table 4.1** Bit allocation for the 4.25 kbps WI coder

| WI Parameters | Transmission Rate | | | |
|---|---|---|---|---|
| | Bits/update | Update Rate | Bits/Frame | Bit/s |
| Pitch | 7 | 50 Hz | 7 | 350 |
| Power | 4 | 100 Hz | 8 | 400 |
| LSFs | 30 | 50 Hz | 30 | 1500 |
| SEW (amplitude) | 14 | 100 Hz | 28 | 1400 |
| REW (amplitude) | 3 | 200 Hz | 12 | 1200 |
| SEW (phase) | 0 | 100 Hz | 0 | 0 |
| REW (phase) | 0 | 400 Hz | 0 | 0 |
| | | Overall Bit Rate: | 85 | 4250 |

### 4.5.1 Subjective Speech Quality

To evaluate the reconstructed speech quality, the coder was subjected to a blind A-B comparison test. The test sentences were the same as the ones employed in the previous subjective test in Section 3.6.2. It should be emphasized that these sentences were not used during the development of the coder.

Each test utterance was processed by the 4.25 kbps WI coder and the 8 kbps G.729 coder[3]. The resulting pair was presented in random order to seven non-expert listeners. The outcome is summarized in Table 4.2.

The statistics show that the speech quality of the WI coder is comparable to (or better than) that of G.729 for 45% of the utterances. Though the toll-quality goal has not been reached, the coder yields high-quality coded speech which is very intelligible, natural and free of artifacts. No significant discontinuity can be heard and during

[3]G.729 is the ITU-T toll-quality 8 kbps standard.

**Table 4.2** Paired comparison test results between the 4.25 kbps WI and the 8 kbps G.729

| Preference | Number of votes | Percentage of votes |
|---|---|---|
| Prefer WI | 7 | 5% |
| No preference | 59 | 40% |
| Prefer G.729 | 80 | 55% |
| Total: | 146 | 100% |

voicing, the periodicity is generally well maintained. However, a buzzy and a noisy characters are slightly audible in a small portion of the test segments. Moreover, the voicing of some low-frequency segments tends to be perceived as being different from that of the original; the speaker identity is altered by the coder.

Further investigations have led us to believe that these distortions are attributable to the use of the default phase and the inability to maintain a proper SEW/REW energy ratio after the quantization process. These two causes will be described more precisely in the following.

### The default phase in SEW

The use of the default phase in the SEW is based on the assumption that our ears are relatively insensitive to phase information. Apparently, this assumption is not always true in the WI context. In an additional listening test, the coder has been proven to produce natural sounding speech for various speakers when using the original SEW phase in the 4.25 kbps coder. However, after the default phase is inserted into the SEWs, the output speech seems to lose a certain degree of speaker recognizability. This effect is more pronounced for low-frequency speakers (approx. $< 120$ Hz). Although no experiments were devised to measure how much speaker recognizability is affected, we believe that the SEW phase may hold certain information necessary to distinguish low-frequency speakers from one another. Yet, we still do not have an explanation as to why the default phase works well with high-frequency segments only. Further experiments are required to understand and resolve this question conclusively.

In fact, the default phase also contributes to the buzziness problem. As mentioned in Section 4.4.3, the default phase converts all SEWs of different dimensions into one distinct pitch pulse. This inevitably increases the similarities between the shapes of

the SEWs, which in turn causes unwarranted periodicity in the output speech.

#### The SEW/REW energy ratio

We have noticed that the SEW/REW energy ratio is occasionally altered by the quantization of the amplitude spectra (processors **344**, **345**, **346** and **347**). In fact, this energy ratio is of great importance to the performance of the coder. Our experiments confirmed that if the SEW is over-emphasized (or the REW is under-emphasized), the reconstructed speech would suffer from buzziness. On the other hand, over-emphasizing the REWs (or under-emphasizing the SEWs) may lead to a noisier output.

Our current coding scheme makes no attempts to control this energy ratio. As a result, the SEW/REW energy ratio before the quantization could be significantly different from the one after the quantization. This causes the output speech to have the slightly buzzy and noisy characteristics.

In summary, in order to bring the coder closer to the toll-quality goal, the default phase can still be employed but only limited to the high-frequency segments. Whereas for low-frequency segments, some phase quantizations may be necessary to maintain a high accuracy of talker characteristics. A mechanism must also be in place to ensure the default phase does not introduce extra periodicity in the output speech. On the other hand, it is also imperative to maintain a proper balance balance between the SEW and the REW energy levels in the reconstructed speech. An imbalance of the SEW/REW energy ratio causes the output to sound buzzy and noisy.

### 4.5.2 Algorithmic Delay

Our 4.25 kbps WI coder encodes speech with 20 ms frames. There is a look-ahead of 15 ms contributed by the LP filtering, the pitch estimation and the CW extraction procedures. The filtering operation required for the SEW and the REW decomposition adds an additional delay of 20 ms, resulting in a total algorithmic delay of 55 ms. This amount of delay is significantly higher than that of G.729 which has a delay of 15 ms.

# Chapter 5

# Concluding Remarks

A waveform interpolation speech coder operating at 4.25 kbps has been simulated using the C language. The resulting speech quality has been assessed subjectively. In this last chapter, we will summarize our work and review a few of the attractive features offered by the WI coding scheme. Also, we will discuss some of the remaining issues associated with the current implementation of the coder. A list of potential solutions will be outlined as part of the future work.

## 5.1 Summary of Our Work

In the first chapter, we provided some background information about speech coding which encompassed the properties of speech signals and the basic attributes of speech coders. The motivation and the scope of our research were outlined. The main goal was to develop a WI-based speech coder, with the intention of delivering near toll quality at rates around 4 kbps.

Chapter 2 gave a brief overview of the short-term linear predictive coding analysis. The concepts of the line spectral frequency, bandwidth expansion and pre-emphasis were introduced.

Chapter 3 began to introduce the background and concept of the WI coding scheme. We provided an extensive coverage on the implementation of the algorithm, with an emphasis on the analysis-synthesis layer (i.e., the unquantized model). The mathematical derivations for each processor were formulated.

The analysis layer decomposes a speech signal into four parameters — the LSFs,

pitch, power and the CWs. The synthesis layer reconstructs the speech signal from these four parameters. Both layers were constructed in such a way that they are robust to pitch multiple and sub-multiple occurrences.

The accuracy of the analysis-synthesis layer was verified by the subjective evaluation tests; the unquantized coder outperformed the 32 kbps ADPCM under clean input speech conditions. Despite the high performance of the model, we noticed that the reconstructed speech sometimes has undesired envelope variations. This is mainly caused by the time-varying effects introduced by the LP filters. Nevertheless, this distortion is barely audible.

Next, we discussed two of the most popular WI derivatives. In the first, both the analysis and synthesis layers are executed directly on the speech domain rather than on the residual domain. Preliminary experiments concluded that a more precise pitch estimation (i.e., fractional estimation) or a more sophiscated extraction procedure (e.g., perform the pitch estimation and the extraction simultaneously) is necessary to ensure that the extracted speech-domain CWs have low boundary energies. Otherwise, audible discontinuities can be noticeable in the output speech.

The second WI derivative performs analysis on the residual domain but synthesis on the speech domain. In such a configuration, the residual-domain CWs are transformed to the speech-domain CWs before the 2D-to-1D conversion takes place. This transformation is accomplished by circularly convolving the CWs with the impulse response represented by the LP coefficients. Our experiments indicated that the output of the circular convolution provides an excellent approximation to that of the linear convolution when the CW extraction rate is below 10/frame. On the other hand, as the extraction rate increases beyond $\geq$ 20/frame, this approximation seems inappropriate. The circular convolution starts to deviate numerically as well as perceptually from the linear convolution. In other words, this circular convolution configuration works well only when the extraction rate is not too high.

Also in Chapter 3, we discussed how bandwidth expansion can increase the accuracy of the pitch estimation and the waveform alignment procedures. We also described how the WI analysis-synthesis system can be used to do time-scale modifications on a speech signal.

In Chapter 4, we focused on developing a WI quantization scheme targeting at rates around 4 kbps. The coding scheme for each parameter of the coder was de-

scribed. The motivation and the implementation of the SEW-REW decomposition were also given.

The final WI coder operates at 4.25 kbps and it was subjected to A-B preference tests, with the ITU G.729 as the reference coder. The test results indicated that the WI coder was judged to be equivalent to or better than G.729 for 45% of the test utterances. Though the toll-quality goal is not yet achieved, the reconstructed speech is highly intelligible and natural. However, it can be noticed that the coded speech sometimes has a slightly buzzy and noisy characteristics. Moreover, the speaker identity is altered to a small extent, especially for low-frequency male speakers.

Investigations concluded that these distortions are attributable to the use of default phase in SEWs and the inability to maintain the SEW/REW energy ratio after quantizations. It is anticipated that the coder would be much closer to the toll-quality benchmark if these two problems are addressed.

## 5.2 Strength of the WI Scheme

Comparatively speaking, the WI coder offers many desirable features which are not common in most conventional low-bit-rate speech coders. Some of these features are listed as follows:

- The success of the WI coding scheme is in large part due to its inherent capability of producing an accurate level of periodicity for voiced speech, even at extremely low bit rates. This contrasts with most CELP-based coders which fail to maintain an appropriate periodicity when operating at about the same rates. In addition, WI provides an excellent framework to efficiently analyze, control and regulate the periodicity of voiced speech.

- The WI model (unquantized) has been proven to produce almost transparent speech quality. In other words, the performance of the WI coder is limited by quantizers, not by the model.

- One important advantage that WI offers is that it decomposes speech into relatively uncoupled parameters — the LP coefficients, power, pitch, SEW and the REW. Such independence allows the parameters to be quantized more efficiently. It also allows the parameters to be manipulated and controlled sepa-

rately. In fact, it is such independence that makes the time-scaling modifications possible in WI (Section 3.9).

- WI avoids the use of any binary classifiers such as a voiced-unvoiced (V/UV) detector. The avoidance of such classifications facilitates robustness to noisy environments and to channel errors. It also removes the inherent jumps introduced by switching between different modes or coding schemes. Generally, if this switching is present in a coder, the speech quality does not normally converge to that of the original signal with increasing bit rates.

- In practice, not all speech segments are completely voiced or completely unvoiced; some contain a mixture of both. The traditional low-bit-rate coders which employ V/UV classifiers are not able to cope with these situations. However, the SEW-REW decomposition procedure gives WI the capability to handle such segments. As a result, the coder behaves much more robustly, particularly when input speech signals are corrupted by acoustic background noise.

- Although it is not the main purpose of the coder, WI can also provide voicing information of a speech signal (i.e., a voicing detector). It does so by computing the energy ratio between the REW (or SEW) and the CW.

## 5.3 Future Research Directions

In this section, we list the remaining issues that are related to the current implementation of the coder and that we believe would be interesting to further investigate. Some potential solutions to the issues will also be described.

- Our investigations concluded that the SEW/REW energy ratio requires more attention. A quantization scheme for the SEW/REW energy ratio needs to be devised.

- Further research is required to understand why the default phase does not work well with low-frequency male speakers. How much improvement will we get if we transmit at least partial phase information for the low-frequency components? How much speaker recognizability information does the phase spectra of the SEWs hold? These questions are yet to be answered by experiments.

- Our experimental evidence confirmed that the default phase added to the SEWs can lead to unwarranted periodicity and hence, buzziness in the output speech. One possible solution to this problem is as follows. Instead of using one default phase for all SEWs, a phase selected from a small family of phase responses could be used in the SEW construction. By choosing the default phase from a family of phase contours, the similarities between the CWs can be substantially reduced. Direct phase quantization can also be a solution to this problem [47, 48].

- In our present work, a non-differential quantization scheme is used for coding the SEW so as to make the coder more robust to channel errors. However, in a noise free environment, it is advantageous to use a differential coding scheme because the spectra of successive SEWs can be very similar from time to time.

- The present REW and SEW magnitude quantizations are based on open-loop schemes. There are also many other closed-loop quantization schemes. For example, [49, 50, 51] successfully incorporate the analysis-by-synthesis technique into the WI coding.

- It is well known that our human auditory system has a frequency resolution which decreases rapidly with increasing frequency. In a recent paper [21], this knowledge is exploited in the SEW magnitude quantization and a substantial improvement in performance has been reported. It is worthwhile to investigate how much improvement this quantization technique can bring to our coder.

- WI relies heavily on the pitch estimate to generate an accurate periodicity in the output speech. As a result, when the input speech signal is severely contaminated by acoustic background noise, the coder output may contain significant distortions. On the other hand, when the input does not have an apparent periodicity such as music or multi-speaker utterances, the WI coder also behaves erratically. Therefore, further research work is warranted to make WI less pitch-dependent.

- One promising way to bring WI closer to toll-quality at 4 kbps is to adopt the multimode Fixed Bit-Rate (FBR) or Variable Bit-Rate (VBR) schemes. In both cases, the coder operates on various modes. The key difference between the two

schemes is that the FBR operates at the same rate for all modes whereas the VBR has different rates for different modes. In fact, speech is very suitable for the multimode FBR or VBR due to its non-stationary character. Furthermore, the voice activity during telephone conversation can be expected to be less than 50%. Voiced and unvoiced speech have different perceptual relevance as well.

Recently, the introduction of Code-Division Multiple Access (CDMA) for digital cellular telephony motivated a new surge of interest in multimode VBR. Such variable bit-rate transmission rate is crucial to the performance of a CDMA system because it minimizes the mutual interference among users and hence raises the system capacity. Therefore, as the CDMA technology continues to grow, there will be a significant demand for various types of VBR speech coders and a WI-based VBR coder is potentially one of the candidates.

- Our quantization layer is not yet capable of handling pitch (sub)multiple occurrences. The main obstacle to implementing such a feature is the insertion of the default phase into the SEWs. Future work is required to revise this default phase as well as other components of the quantization layer so that the entire WI coder is robust to pitch (sub)multiple situations.

- The impact of channel errors on the coder performance has not been formally assessed in this thesis. Robustness against channel noise has always been an important concern to cellular users as the mobile environments suffer from high levels of channel errors. In speech coding, it is customary to apply various quantization strategies to reduce the channel-noise distortion on reconstructed speech. For example in our present scheme, a leaky differential quantizer can be used to code the power to prevent indefinite channel-error propagation.

- The computational load in our current WI implementation is far greater than desired and may cause problems in real-time applications. A bulk of the computational load is due to the CW interpolation procedure and the waveform alignment (realignment) process. The present optimized WI algorithm (including the analysis, coding, decoding and the synthesis) requires approximately 1.6 second of processing time per second of input speech, on a 200 MHz Sun SPARC general purpose computer system. Such a complexity may not be within grasp

in the present DSP chip technology. However, in a recent paper [20], several novel methods have been developed to drastically reduce the complexity in WI.

- Due to its inherent time asynchrony, WI is not able to reconstruct speech exactly in the absence of quantization. Recently, [22] claims that perfect reconstruction in WI is possible with the use of Gabor Transform. It would be interesting to find out how much improvement this new technique can bring to our coder.

- The algorithmic delay of the WI coder is at the high end of the scale as compared to most conventional toll-quality speech coders. The delay of the current implementation is 55 ms which is 40 ms longer than that of G.729. Such delay may not be compatible with the ITU-T 4 kbps standard. One straightforward way to reduce the delay is to use shorter frames. This, however, may affect the performance of the SEW-REW decomposition in which the order (and the accuracy) of the FIR decomposition filter is proportional to the size of the frame. Reference [52] employs filters other than FIR to accomplish the decomposition. Reportedly, it achieves high filtering accuracy with less coder delay.

- It is still uncertain as to whether the SEW-REW decomposition is optimal. New experimental evidence in [52] has shown that the REW contains significant components of the SEW. In other words, the two components may not be entirely independent of each other. Therefore, it is feasible that some other decomposition schemes may give similar or even higher performance.

- Contemporary speech coders use many extra components like pre- and post-processing for further quality enhancement. For instance, G.729 employs an adaptive postfilter with spectral tilt compensation. It is expected that such pre- and post-processing can further enhance the quality of our coded speech.

- As discussed in Section 3.6.3, the non-linear effects of the LP filter occasionally causes variations in the time envelope of the output speech. For high-power input speech signals, such variations may result in clipping in the coded speech. Special procedures are yet to be devised to control the time envelope variations.

- The circular shifting in the alignment procedure often causes a CW to have its "tail" before its "front". It is still unclear whether such procedure is justified

despite the high performance of the analysis-synthesis layer. Reference [33] avoids such circular shifting by elegantly incorporating the alignment procedure into the extraction process. That is, the CWs are extracted in such a way that they are aligned for maximum correlation.

- Reference [23] demonstrated that "transparent coding" quality can be achieved in the LSF quantization by using split-VQ at 24 bits/frame. Recently, [53, 54] reported excellent results at even lower rates — around 20 bits/frame. Since our LSFs are coded at 30 bits/frame, this means that we can free up at least 6 bits/frame for improving the quantizations of other coder parameters.

# Appendix A

## The Constants in the WI Coder

**Table A.1** The constants used in the WI simulation

| Symbol | Value | Description |
|---|---|---|
| $L_f$ | 160 | Frame length |
| $L_sf$ | 20 | Subframe length |
| $N$ | 10 | Order of the LP filter |
| $\gamma$ | 0.98829 | Bandwidth expansion of the LP filter |
| $L_w$ | 240 | Length of the LP analysis window |
| $L_{sf}$ | 20 | Subframe length, $L_{sf} = L_f \div R_{extr}$ |
| $P_{\min}$ | 20 | Minimum pitch period allowed |
| $P_{\max}$ | 120 | Maximum pitch period allowed |
| $R_{extr}$ | 8 | Number of extractions per frame |
| $\delta$ | 10 | Boundary energy window length (alignment process) |
| $-$ | 1/4 | Alignment resolution |
| $\alpha$ | 0.1 | Pre-emphasis factor |
| $\gamma_z$ | 0.85 | Bandwidth expansion in the numerator of processor **448** |
| $\gamma_p$ | 0.9 | Bandwidth expansion in the denominator of processor **448** |
| $\gamma_w$ | 0.8 | Perceptual weighting factor in SEW quantizers **344** |
| $\epsilon_{\max}$ | 16 | Maximum extraction point offset |

# Bibliography

[1] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Prentice-Hall, 1978.

[2] V. C. Welch and T. E. Tremain, "A new government standard 2400 bps speech coder," *Proc. IEEE Workshop on Speech Coding for Telecom.* (Sainte-Adèle, Québec), pp. 41–42, Oct. 1993.

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

[4] R. M. Gray, "Vector quantization," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-34, Apr. 1984.

[5] A. Das, A. V. Rao, and A. Gersho, "Variable-dimension vector quantization," *IEEE Signal Processing Letters*, vol. 3, pp. 200–202, July 1996.

[6] D. O'Shaughnessy, *Speech Communication: Human and Machine*. Addison-Wesley Publishing Company, 1987.

[7] W. B. Kleijn, "Continuous representations in linear predictive coding," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Toronto), pp. 201–204, May 1991.

[8] W. B. Kleijn and W. Granzow, "Methods for waveform interpolation in speech coding," *Digital Signal Processing*, vol. 1, pp. 215–230, Jan. 1991.

[9] W. B. Kleijn, "Encoding speech using prototype waveforms," *IEEE Trans. Speech and Audio Processing*, vol. 1, pp. 386–399, Oct. 1993.

[10] A. S. Spanias, "Speech coding: A tutorial review," *Proc. IEEE*, vol. 82, pp. 1541–1582, Oct. 1994.

[11] G. Kubin, B. S. Atal, and W. B. Kleijn, "Performance of noise excitation for unvoiced speech," *Proc. IEEE Workshop on Speech Coding for Telecom.* (Sainte-Adèle, Québec), pp. 35–36, Oct. 1993.

[12] I. A. Atkinson, A. M. Kondoz, and B. G. Evans, "Time envelope vocoder, a new LP based coding strategy for use at bit rates of 2.4 kb/s and below," *IEEE J. Selected Areas Commun.*, vol. 13, pp. 449–457, Feb. 1995.

[13] I. A. Atkinson, A. M. Kondoz, and B. G. Evans, "Time envelope LP vocoder: A new coding technique at very low bit rates," *Proc. European Conf. on Speech Commun. and Technology* (Madrid), pp. 241–244, Sept. 1995.

[14] J.-H. Chen, R. V. Cox, Y.-C. Lin, N. Jayant, and M. J. Melchner, "A low delay CELP coder for the CCITT 16 kb/s speech coding standard," *IEEE J. Selected Areas Commun.*, vol. 10, pp. 830–849, June 1992.

[15] R. Salami, C. Laflamme, J.-P. Adoul, A. Kataoka, S. Hayashi, C. Lamblin, D. Massaloux, S. Proust, P. Kroon, and Y. Shoham, "Description of the proposed ITU-T 8-kb/s speech coding standard," *Proc. IEEE Workshop on Speech Coding for Telecom.* (Annapolis), pp. 3–4, Sept. 1995.

[16] A. McCree, K. Truong, E. B. George, T. P. Barnwell III, and V. Viswanathan, "A 2.4 kbit/s MELP coder candidate for the new U.S. Federal Standard," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Atlanta), pp. 200–203, May 1996.

[17] W. B. Kleijn, *Analysis-by-Synthesis Speech Coding Based on Relaxed Waveform Matching Constraints*. PhD thesis, Delf University of Technology, Delf, The Netherlands, Dec. 1991.

[18] W. B. Kleijn and J. Haagen, "A general Waveform-Interpolation structure," *Proc. European Signal Processing Conf.* (Edinburg), pp. 1665–1668, Sept. 1994.

[19] W. B. Kleijn and J. Haagen, "Speech coder based on decomposition of characteristic waveforms," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Detroit), pp. 508–511, May 1995.

[20] W. B. Kleijn, Y. Shoham, D. Sen, and R. Hagen, "A low-comlexity Waveform Interpolation coder," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Atlanta), pp. 212–215, May 1996.

[21] J. Thyssen, W. B. Kleijn, and R. Hagen, "Using a perception-based frequency scale in Waveform Interpolation," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Munich, Germany), pp. 1595–1598, Apr. 1997.

[22] H. Yang and W. B. Kleijn, "Pitch-synchronous subband representation of the linear prediction residual of speech," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Seattle), pp. 529–532, May 1998.

[23] K. K. Paliwal and B. S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame," *IEEE Trans. Speech and Audio Processing*, vol. 1, pp. 3–14, Jan. 1993.

[24] G. H. Golub and C. F. V. Loan, *Matrix Computations*. The Johns Hopkins University Press, second ed., 1989.

[25] W. B. Kleijn and K. K. Paliwal, eds., *Speech Coding and Synthesis*. Elsevier, 1995.

[26] F. Itakura, "Line spectrum representation of linear prediction coefficients of speech signals," *Journal Acoustical Society America*, vol. 57, p. 535, 1975. (abstract).

[27] P. Kabal and R. P. Ramachandran, "The computation of line spectral frequencies using Chebyshev polynomials," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-34, pp. 1419–1426, Dec. 1986.

[28] J. R. Deller Jr., J. G. Proakis, and J. H. L. Hansen, *Discrete-Time Processing of Speech Signal*. Macmillan, 1993.

[29] W. B. Kleijn and J. Haagen, "Transformation and decomposition of the speech signal for coding," *IEEE Signal Processing Letters*, vol. 1, pp. 136–138, Sept. 1994.

[30] W. B. Kleijn and J. Haagen, "Waveform interpolation for coding and synthesis," in *Speech Coding and Synthesis* (W. B. Kleijn and K. K. Paliwal, eds.), pp. 175–208, Elsevier, 1995.

[31] B. S. Atal, V. Cuperman, and A. Gersho, eds., *Advances in Speech Coding*. Kluwer Academic Publishers, 1991.

[32] Telecommunications Industry Association, TIA/EIA/PN-3292, *EIA/TIA Interim Standard, Enhanced Variable Rate Codec (EVRC)*, Mar. 1996.

[33] J. Stachurski, *A Pitch Pulse Evolution Model for Linear Predictive Coding of Speech*. PhD thesis, McGill University, Montreal, Canada, May 1997.

[34] M. Leong, "Representing voiced speech using protoype waveform interpolation for low-rate speech coding," Master's thesis, McGill University, Montreal, Canada, Nov. 1992.

[35] M. Leong and P. Kabal, "Smooth speech reconstruction using Prototype Waveform Interpolation," *Proc. IEEE Workshop on Speech Coding for Telecom.* (Sainte-Adèle, Québec), pp. 39–41, Oct. 1993.

[36] K. Yaghmaie and A. M. Kondoz, "Multiband prototype waveform analysis synthesis for very low bit rate speech coding," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Munich, Germany), pp. 1571–1574, Apr. 1997.

[37] D. Marston and F. Plante, "PWI speech coder in the speech domain," *Proc. IEEE Workshop on Speech Coding for Telecom.* (Pennsylvania), pp. 31–32, Sept. 1997.

[38] J. Stachurski and P. Kabal, "A pitch pulse evolution model for a dual excitation linear predictive speech coder," *Proc. Seventeenth Biennial Symposium on Communications* (Kingston), pp. 107–110, May 1994.

[39] D. W. Griffin and J. S. Lim, "Multiband excitation vocoder," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. 36, pp. 1223–1235, Aug. 1988.

[40] J. C. Hardwick and J. S. Lim, "A 4800 bps improved multi-band excitation speech coder," *Proc. IEEE Workshop on Speech Coding for Telecom.* (Vancouver), Sept. 1989.

[41] Y. Shoham, "Low-rate speech coding based on time-frequency interpolation," *Proc. Int. Conf. on Spoken Language Processing*, pp. 37–40, Oct. 1992.

[42] A. McCree and W. B. Kleijn, "Mixed Excitation Prototype Waveform Interpolation for low bit rate speech coding," *Proc. IEEE Workshop on Speech Coding for Telecom.* (Sainte-Adèle, Québec), pp. 51–52, Oct. 1993.

[43] Y. Tanaka and H. Kimura, "Low-bit-rate speech coding using a two-dimensional transform of residual signals and Waveform Interpolation," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Adelaide), pp. 173–176, Apr. 1994.

[44] D. Sen and W. B. Kleijn, "Synthesis methods in sinusoidal and Waveform-Interpolation coders," *Proc. IEEE Workshop on Speech Coding for Telecom.* (Annapolis), Sept. 1995.

[45] M. R. Zad-Issa and P. Kabal, "A new LPC error criterion for improved pitch tracking," *IEEE Workshop on Speech Coding* (Pocono Manor, PA), pp. 1–2, 1997.

[46] B. Sylvestre, "Time-scale modification of speech: A time-frequency approach," Master's thesis, McGill University, Montreal, Canada, Apr. 1991.

[47] Y. Jiang and V. Cuperman, "Encoding prototype waveforms using a phase codebook," *Proc. IEEE Workshop on Speech Coding for Telecom.* (Annapolis), pp. 21–22, Sept. 1995.

[48] M. Festa and D. Sereno, "A speech coding algorithm based on prototype interpolation with critical bands and phase coding," *Proc. European Conf. on Speech Commun. and Technology* (Madrid), pp. 229–232, Sept. 1995.

[49] I. S. Burnett and G. J. Bradley, "Low complexity decomposition and coding of prototype waveforms," *Proc. IEEE Workshop on Speech Coding for Telecom.* (Annapolis), pp. 23–24, Sept. 1995.

[50] I. S. Burnett and G. J. Bradley, "New techniques for multi-prototype waveform coding at 2.84 kb/s," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Detroit), pp. 261–264, May 1995.

[51] I. S. Burnett and J. Ni, "Waveform Interpolation and analysis-by-synthesis — a good match," *IEEE Workshop on Speech Coding* (Pocono Manor, PA), pp. 29–30, Sept. 1997.

[52] N. R. Chong, I. S. Burnett, J. F. Chicharo, and M. M. Thomson, "Use of the pitch synchronous wavelet transform as a new decomposition method for WI," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Seattle), pp. 513–516, May 1998.

[53] J. H. Y. Loo and W.-Y. Chan, "Nonlinear predictive vector quantization of speech spectral parameters," in *Proc. IEEE Workshop on Speech Coding for Telecom.*, (Annapolis, MD), pp. 51–52, September 1995.

[54] J. Skoglund and J. Lindén, "Predictive VQ for noisy channel spectrum coding: AR or MA," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing* (Munich, Germany), pp. 1351–1354, Apr. 1997.