17. Analysis-by-Synthesis Speech Coding

J.-H. Chen, J. Thyssen

Since the early 1980s, advances in speech coding technologies have enabled speech coders to achieve bit-rate reductions of a factor of 4 to 8 while maintaining roughly the same high speech quality. One of the most important driving forces behind this feat is the so-called *analysis-by-synthesis* paradigm for coding the excitation signal of predictive speech coders. In this chapter, we give an overview of many variations of the analysis-by-synthesis excitation coding paradigm as exemplified by various speech coding standards around the world. We describe the variations of the same basic theme in the context of different coder structures where these techniques are employed. We also attempt to show the relationship between them in the form of a *family tree*. The goal of this chapter is to give the readers a big-picture understanding of the dominant types of analysis-by-synthesis excitation coding techniques for predictive speech coding.

17.1 Overview

Historically, speech coding technology has been dominated by coders based on linear prediction. To achieve good speech quality, most speech coding standards are *waveform-approximating coders* (or *waveform coders* for short), and the majority use linear prediction to exploit the redundancy in the speech waveform.

Of course, the techniques in speech coding standards are not the only ones available, as there are many other speech coding techniques proposed in the literature that have not been used in standards. However, speech coding standards represent the dominant and most widely deployed speech coding techniques. Often they also include the best-performing techniques for the given requirements of bit rate, coding delay, and codec complexity at the time they were standardized. Therefore, it is useful to examine some of the techniques in speech coding standards to see how the dominant speech coding techniques have evolved over the years. For a comprehensive review of speech coding standards up to about 1995, see [17.1].

The first speech coding standard based on predictive waveform coding is probably 32 kb/s adaptive differential pulse code modulation (ADPCM), which was initially standardized by the Comité Consultatif International Téléphonique et Télégraphique (CCITT, the predecessor of the ITU-T) as recommendation G.721 in 1984 and later modified and restandardized as part of G.726 in 1986. The G.726 standard is a narrow-band (telephone-bandwidth) speech coder with an input sampling rate of 8 kHz and encoding bit rates of 16, 24, 32, and 40 kb/s, with 32 kb/s being the most widely used bit rate of G.726.

The basic idea of the G.726 ADPCM coder is to use an adaptive linear predictor to predict the input speech signal, and then use an adaptive scalar quantizer to quantize the difference signal between the input speech and the predicted version. Since statistically this difference signal tends to be smaller than the input speech signal, one can use a lower bit rate to quantize the difference signal to the same precision as direct quantization of the speech signal itself. This is how predictive coders achieve bit-rate reduction through the use of prediction to exploit the redundancy between nearby speech samples. The difference signal is often called the *prediction residual signal*, because it is the residual of the speech signal after the predictable portion is removed. In analysis-by-synthesis speech coders, this prediction residual signal is often referred to as the *excitation signal*. The reason will become clear during later discussions in this chapter.

In 1982, at about the same time that G.721 ADPCM was being developed, *Atal* and *Remde* proposed the first analysis-by-synthesis speech waveform coding technique called *multipulse linear predictive coding* (MPLPC) [17.2]. This work led to *Atal* and *Schroeder*'s 1984 proposal of another analysis-by-synthesis speech coding technique called *stochastic coding* [17.3], which was renamed *code-excited linear prediction* (CELP) [17.4] in 1985. From then on, CELP became the dominant speech coding technique for the next two decades, with hundreds or perhaps even thousands of technical papers published on variations of CELP techniques.

Due to their high coding efficiency, variations of CELP coding techniques together with other advancements have enabled speech waveform coders to halve the bit rate of 32 kb/s G.726 ADPCM three times while maintaining roughly the same speech quality. This is evidenced by the ITU-T's speech coding standardization effort after standardizing 32 kb/s ADPCM in 1984.

The first halving of bit rate to 16 kb/s happened with the ITU-T G.728 low-delay CELP (LD-CELP) coder [17.5, 6]. At the cost of increasing the buffering delay from one to five samples (0.625 ms at 8 kHz sampling rate), the G.728 coder halved the bit rate of 32 kb/s G.726 ADPCM while maintaining equivalent or better speech quality for all conditions tested.

The second halving of bit rate to 8 kb/s happened with the ITU-T G.729 conjugate-structure algebraic CELP (CS-ACELP) [17.7]. At the cost of increasing the buffering delay further to 80 samples of frame size plus 40 samples of look-ahead (or 15 ms total), the G.729 coder halved the bit rate again to 8 kb/s while maintaining the speech quality for most test conditions except tandeming and speech in background noise.

The third halving of bit rate to 4 kb/s happened with candidate coders [17.8, 9] submitted for the ITU-T's 4 kb/s speech coding standardization. A promising candidate 4 kb/s coder [17.8] was based on CELP. Although the ITU-T eventually did not standardize a 4 kb/s coder, this CELP-based candidate coder did achieve equivalent speech quality as 32 kb/s G.726 ADPCM at least for clean speech conditions at the cost of further increasing the buffering delay to 30–35 ms.

Of course, the G.728, G.729, and ITU-T 4 kb/s candidate coders are merely three example coders that help to make a point that the analysis-by-synthesis

speech coding technique was the main driving force behind the feat of reducing the bit rate by a factor of 4 to 8 within two decades while maintaining equivalent speech quality, at least for clean speech. In the last two decades, researchers have proposed numerous other speech waveform coders in the bit-rate range of 4–16 kb/s that achieve speech quality equivalent to or better than the speech quality of the 32 kb/s G.726 ADPCM coder. The vast majority of these are analysis-by-synthesis speech coders based on either CELP or MPLPC. Therefore, it is fair to say that *analysis-by-synthesis* is undeniably the most important speech coding method in modern low-bit-rate speech waveform coding.

The main emphasis of this chapter is to describe many of the major flavors of analysis-by-synthesis excitation coding for linear predictive speech waveform coders, as exemplified by various speech coding standards around the world. Due to the space limitation, it is not possible to cover all speech coding standards or even all aspects of the standards discussed in this chapter. Readers interested in learning more about other aspects of analysis-by-synthesis speech coders are referred to [17.10].

The rest of this chapter is organized as follows. Section 17.2 describes the basic concepts of the analysis-by-synthesis paradigm in the context of speech coding. Section 17.3 gives an overview of the different flavors of analysis-by-synthesis excitation coding and shows the relationship between them in the form of a *family tree*. Sections 17.4–17.17 then provide somewhat more-detailed descriptions of these different flavors of analysis-by-synthesis speech coders one-by-one, including MPLPC, original CELP, regular-pulse excitation (RPE), federal standard 1016 (FS1016) CELP, vector sum excited linear prediction (VSELP), LD-CELP, pitch synchronous innovation CELP (PSI-CELP), ACELP, CS-ACELP, relaxed CELP (RCELP), extended CELP (eX-CELP), forward backward linear predictive coding (FB-LPC), two-stage noise feedback coding (TSNFC), and embedded CELP, roughly in a chronological order. Section 17.18 summarizes these analysis-by-synthesis speech coders in a table format. Finally, Sect. 17.19 concludes this chapter.

17.2 Basic Concepts of Analysis-by-Synthesis Coding

17.2.1 Definition of Analysis-by-Synthesis

What exactly is *analysis-by-synthesis*? The phrase "*analysis-by-synthesis*" can be traced back to at least 1959 in a paper by *Halle* and *Stevens* [17.11]. In another 1961 paper by *Bell* et al. [17.12], a detailed definition of analysis-by-synthesis is given as follows:

> *The term analysis-by-synthesis is used to refer to an active analysis process that can be applied to signals that are produced by a generator whose properties are known. The heart of an analysis-by-synthesis system is a signal generator capable of synthesizing all and only the signals to be analyzed. The signals synthesized by the generator are compared with the signals to be analyzed, and a measure of error is computed. Different signals are generated until one is found that causes the error to reach some smallest value, at which time the analyzer indicates the properties of the internally generated signal.*

Although this definition was given four and a half decades ago in a paper discussing the subject of speech analysis, it is still equally valid today as a description of the basic concepts behind analysis-by-synthesis speech waveform coding.

17.2.2 From Conventional Predictive Waveform Coding to a Speech Synthesis Model

To understand analysis-by-synthesis speech waveform coding, it is useful to look first at conventional predictive speech waveform coding. In conventional predictive speech coders based on linear prediction, the prediction is typically performed using a weighted sum of previously quantized speech samples rather than unquantized input speech samples. This is because the quantized speech samples are available at the decoder while the input speech samples are not. Performing linear prediction based on the quantized speech in the encoder enables the decoder to produce the same predicted speech and track the encoder states in the absence of transmission errors. Prediction based on previously quantized speech signal is called *closed-loop* prediction, while prediction based on unquantized input speech signal is called *open-loop* prediction. For in-depth discussion of predictive waveform coding principles, see [17.13].

The encoder and decoder of the most basic form of closed-loop predictive waveform coding is shown in Fig. 17.1. In the encoder in Fig. 17.1a, the linear predictor, represented by the transfer function $P(z)$,

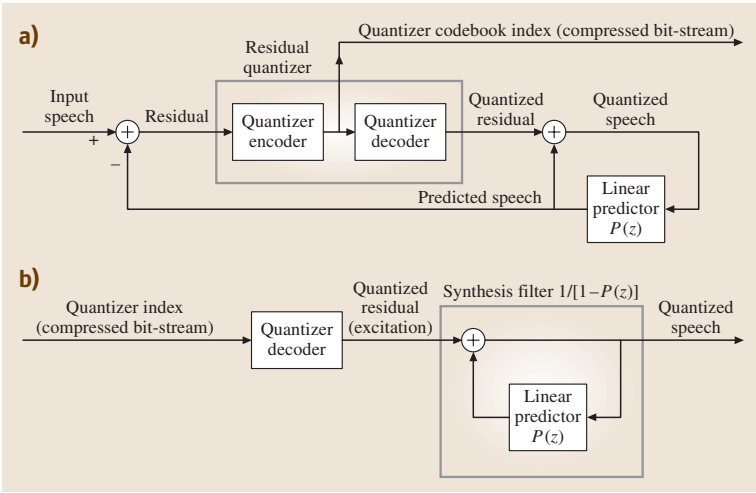

Fig. 17.1a,b Basic structure of conventional linear predictive speech waveform coder. **(a)** Speech encoder, and **(b)** speech decoder

uses previously quantized speech signal as its input to produce the predicted speech signal, which is subtracted from the input speech signal to get the prediction residual signal. After the prediction residual signal is quantized by the quantizer, the same predicted speech signal is added back to the quantized prediction residual signal to get the quantized speech signal. The sequence of quantizer codebook indices corresponding to the sequence of selected quantizer output levels is transmitted to the decoder as the compressed bitstream.

Note that the decoder structure in Fig. 17.1b is basically just a replica of the right half of the encoder structure. Therefore, if the compressed bitstream is received without transmission errors, then the decoder can decode the same quantized speech signal as the quantized speech signal in the encoder. Note that the feedback loop in the decoder that contains the linear predictor can be regarded as a *synthesis filter* with a transfer function of $1/[1-P(z)]$. This synthesis filter corresponds to an autoregressive model. It should also be noted that the quantized prediction residual at the output of the quantizer decoder can be regarded as the *excitation source* for this synthesis filter. From this perspective, the decoder of a conventional linear predictive speech coder can be viewed as a *source-filter model* [17.14] for speech production or speech synthesis. Therefore, the task of the corresponding encoder is to determine the quantized excitation signal and the parameters of the synthesis filter, either on a sample-by-sample or frame-by-frame basis. In other words, the task of the encoder is simply to identify the model parameters of this source-filter model as represented by the decoder structure.

17.2.3 Basic Principle of Analysis by Synthesis

In conventional predictive speech waveform coders, the identification of such model parameters at the encoder is not performed using the analysis-by-synthesis method as defined above by *Bell* et al. in their 1961 paper [17.12]. The encoder in Fig. 17.1a identifies the excitation signal by first calculating the closed-loop prediction residual signal and then directly quantizing this residual signal sample by sample to get the quantized residual (or excitation) signal. The encoder does not perform a synthesis operation when attempting to analyze one of the model parameters, namely, the excitation signal.

In an analysis-by-synthesis speech waveform coder, on the other hand, to achieve better coding efficiency, the prediction residual signal, or excitation signal, is usually quantized block by block rather than sample by sample, where each block is typically 0.5–10 ms long (4–80 samples at 8 kHz sampling). Each block of samples is often called a *vector*. Let the dimension of the excitation vector be K samples and let the excitation encoding bitrate be r bits per sample, then each excitation vector will be represented by Kr bits. Therefore, each vector of the excitation signal can only be quantized into one of 2^{Kr} possible candidate excitation vectors.

An analysis-by-synthesis speech waveform coder does not directly quantize the prediction residual signal as in Fig. 17.1a. Instead, each of the finite 2^{Kr} possible candidates for the excitation signal is passed through the synthesis filter and the resulting synthesized speech signal is compared with the input speech signal. The candidate excitation signal that minimizes a predeter-

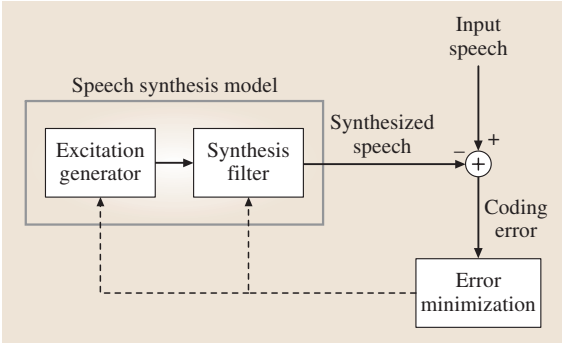

Fig. 17.2 Simplified analysis-by-synthesis speech waveform coder

mined distortion measure between the input speech and the synthesized speech is selected by the encoder as the final excitation signal to be transmitted to the decoder and to be used in the decoder to *excite* the synthesis filter.

The greatly simplified block diagram in Fig. 17.2 illustrates this analysis-by-synthesis coding principle. The gray rectangular box in Fig. 17.2 represents the source-filter speech synthesis model, consisting of an excitation generator followed by a synthesis filter. Note that this speech synthesis model is in direct correspondence to the decoder structure shown in Fig. 17.1b. The difference now is that rather than calculating the prediction residual first and then quantizing it as in Fig. 17.1a, the analysis-by-synthesis coder in Fig. 17.2 passes each of the possible candidate excitation signals through the synthesis filter and identifies the candidate excitation signal that minimizes the error between the corresponding synthesized speech and the input speech. Note that what is shown in Fig. 17.2 is only the encoder structure. The corresponding decoder is simply the excitation generator and synthesis filter enclosed by the gray rectangle.

The encoder in Fig. 17.2 attempts to perform *analysis* of the input speech signal in order to identify the model parameters of the speech synthesis model. The name *analysis-by-synthesis* comes from the fact that the encoder does not try to calculate unquantized model parameters and then quantize them, but instead perform the *analysis* to get the model parameters *by* the *synthesis* of candidate output speech signals using all possible model parameters and identify the set of model parameters that minimizes the error between the synthesized speech and the input speech.

The identification of the best candidate excitation signal that minimizes the coding error is indicated by the dashed arrow going from the *error minimization* block to the *excitation generator* block. This analysis-by-synthesis method of identifying the best model parameters is often called *closed-loop* optimization or quantization, which should not be confused with closed-loop prediction. Closed-loop prediction means the predictor uses previously quantized signal to perform prediction, whereas closed-loop quantization in the current context means the quantization is performed in a way that minimizes the distortion in the speech domain rather than the domain of the parameter to be quantized. In contrast, open-loop quantization means deriving or extracting a parameter and then directly quantize that parameter by minimizing the quantization distortion in the parameter domain.

Strictly speaking, a true analysis-by-synthesis coder will synthesize all possible candidate output speech signals using all combinations of the candidate excitation signals and the candidate synthesis filters and then identify the particular combination that gives the minimum coding error. However, in practice this is rarely done due to the impractically high search complexity that is required. The common approach is to derive the synthesis filter coefficients directly from the input speech signal and perform analysis-by-synthesis coding only for the excitation signal. This approach reduces the performance very little.

17.2.4 Generic Analysis-by-Synthesis Encoder Structure

Figure 17.3 shows a more detailed and more generic encoder structure of an analysis-by-synthesis speech waveform coder. This figure covers most of the analysis-by-synthesis speech coders, at least in the conceptual level. The actual encoder structures used in efficient implementations of such coders may differ, but they are

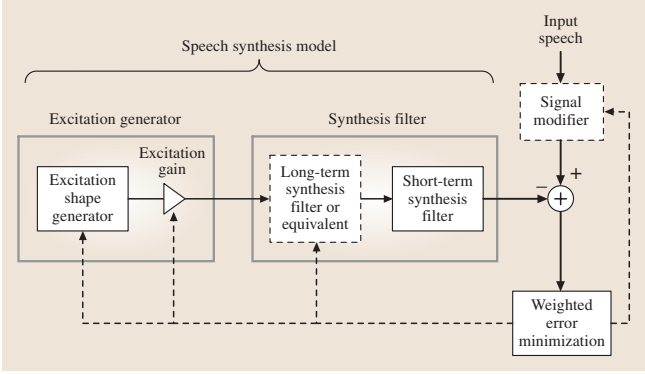

Fig. 17.3 A more detailed generic analysis-by-synthesis speech waveform coder

usually mathematically equivalent to the encoder structure shown in Fig. 17.3. Compared with Fig. 17.2, this figure has

1. added an optional signal modifier for the input speech signal
2. used weighted error minimization
3. expanded the excitation generator and the synthesis filter into more blocks

Each of these differences is briefly explained below.

The addition of the signal modifier is optional; that is why the border of this block is in dashed lines. The purpose of this signal modifier is to modify the input signal in such a way that does not degrade the speech quality appreciably and yet makes the resulting modified speech signal easier for the analysis-by-synthesis coder to encode (i. e., requiring a lower bit rate to encode the signal to the same perceived speech quality) [17.15]. This technique is used to achieve better coding efficiency in some of the CELP coders, such as RCELP [17.15]. However, most analysis-by-synthesis speech coders do not use it because it adds complexity and may occasionally cause slightly audible degradation to speech quality.

The weighted error minimization block in Fig. 17.3 is typically used to shape the coding noise spectrum so that it follows the spectrum of the input signal to some extent – a process usually referred to as *noise spectral shaping* [17.16]. Due to the noise masking effect of the human auditory system, such spectrally shaped coding noise is less audible to human ears.

The excitation generator in Fig. 17.2 is expanded into the excitation shape generator and the excitation gain scaling unit in Fig. 17.3. The excitation shape generator generates excitation vectors with all kinds of possible vector shapes while having gains (as measured by a vector *norm*) either equal to a single value or lie within a narrow range around a single value. The excitation gain scaling unit then scales the excitation shape vectors so that the scaled excitation vector can have a wide dynamic range in terms of vector norm. The separation into generator and scaling blocks is not theoretically necessary, since it is possible for a single excitation generator block to generate excitation signals with all possible shapes and gains. Instead, such separation is motivated by the desire to keep the computational complexity low, since without it a very large excitation codebook is required.

The synthesis filter in Fig. 17.2 is expanded into the cascade of the long-term synthesis filter (or equivalent) and the short-term synthesis filter in Fig. 17.3. A long-term synthesis filter often contains a long-term predictor [17.16] in a feedback loop, and a short-term synthesis filter often contains a short-term predictor in a feedback loop [17.14]. Again, this separation is not theoretically necessary. However, there is an important reason to such decoupling – the long-term synthesis filter (or its equivalent) can model well the quasi-periodicity in voiced speech signals introduced by the periodic vibration of the human vocal cord, while the short-term synthesis filter can model the spectral envelope of speech signals as controlled by the human vocal tract. Thus, such decoupling is motivated by the way the speech signal is produced by a human talker.

Note that the long-term synthesis filter block in Fig. 17.3 is enclosed by dashed lines, signifying that this filter may or may not be used. Most analysis-by-synthesis speech coders do employ this long-term synthesis filter (or its equivalent) as well as the short-term synthesis filter. The G.728 LD-CELP coder [17.6] and the original MPLPC coder [17.2] are two exceptions.

Conceptually, on a block-by-block basis, the encoder in Fig. 17.3 attempts to find the best combination of the excitation shape vector, the excitation gain, the long-term synthesis filter parameters, the short-term synthesis filter parameters, and the signal modifier parameters (if the modifier is present), such that the resulting synthesized speech signal is closest to the modified input speech signal, where *closest* is in the sense of minimizing a weighted mean squared error criterion (distortion measure). Rather than exhaustively searching through all combinations of these parameters, practical analysis-by-synthesis speech coders usually use sequential optimization. In other words, these five sets of parameter values (or four if the signal modifier is not used) are usually optimized one set at a time in a sequential manner so that the computational complexity is manageable. In some speech coders, certain long-term synthesis filter parameters are jointly optimized with at least some of the excitation parameters in an attempt to achieve better coding efficiency. For example, see [17.17].

Essentially all practical analysis-by-synthesis speech coders quantize the short-term synthesis filter parameters in an *open-loop* manner. In other words, the unquantized short-term synthesis filter parameters are first derived from the input speech signal, and then they are directly quantized using a distortion measure in the filter parameter domain rather than using the weighted error in the speech domain as depicted in Fig. 17.3. That is why there is not a dashed arrow from the weight error minimization block to the short-term synthesis filter block in Fig. 17.3.

Exactly how many parameter sets should be closed-loop or jointly quantized depends on the coder design goals and requirements. In some analysis-by-synthesis speech coders that emphasize low computational complexity, only the excitation shape is closed-loop quantized and all other parameter sets are open-loop quantized. As long as the encoding bit rate is not overly low and the speech coder is carefully designed, even such an analysis-by-synthesis coder can achieve fairly high speech quality.

17.2.5 Reasons for the Coding Efficiency of Analysis by Synthesis

As mentioned earlier, the analysis-by-synthesis excitation coding is one of the most efficient speech coding techniques and is the main driving force behind the 4 to 8 times reduction in speech coder bit-rate in the last two decades. A fundamental question to ask is: why is this analysis-by-synthesis technique for predictive speech coders so powerful? One reason is that the speech synthesis model in Fig. 17.3 is a good fit to how speech signal is produced by a human talker.

Perhaps a more important reason is the analysis-by-synthesis method itself. Think of it this way: the task of the encoder is to find the quantized parameters of this speech synthesis model so that the synthesized output speech sounds closest to the input speech. Given this, then which other coding method is better than trying all possible quantized parameters and see which one gives an output speech signal closest to the input speech signal? This *trying all possible values and finding which one gives the best output speech* is the essence of analysis-by-synthesis coding. It is a stark contrast to earlier coding methods where the unquantized optimal parameter value is first derived and then directly quantized using a distortion measure in that parameter domain.

Yet a third reason is that such an analysis-by-synthesis approach naturally lends itself to enable the use of the so-called *vector quantization* (VQ) , which has a higher coding efficiency than conventional sample-by-sample *scalar quantization* as used in ADPCM. Some might argue that ADPCM and adaptive predictive coding (APC) [17.16] are analysis-by-synthesis coders since minimizing the quantization error in the prediction residual domain is mathematically equivalent to minimizing the coding error in the speech domain. While there is indeed such a mathematical equivalence in the degenerate case of scalar quantization, it is questionable, in our opinion, whether ADPCM and APC should be called analysis-by-synthesis coders. Our main objection is that these scalar-quantization-based coders never really perform the *synthesis* operation when trying to do the *analysis* of the model parameters (i. e., when performing residual quantization).

Conventional ADPCM and APC coders are restricted to using sample-by-sample scalar quantization and cannot be extended to vector quantization to reap the benefit of the higher coding efficiency of VQ. (If these coders indeed use VQ, then they will not be called ADPCM and APC anymore and will be called CELP.) These coders need to compute the unquantized prediction residual signal first and then quantize it, but with VQ the unquantized prediction residual signal depends on the quantized prediction residual signal due to the feedback filter structure in Fig. 17.1a. This creates a chicken-and-egg problem. In contrast, the analysis-by-synthesis approach completely avoid this chicken-and-egg problem by directly trying all possible quantized residual (excitation) signals without the need to compute the unquantized residual signal first. Thus, by enabling the use of VQ for the excitation signal, the analysis-by-synthesis approach reaps the benefit of the higher coding efficiency of VQ.

This chapter only explains the most fundamental basic ideas of analysis-by-synthesis coding. If the encoder structure in Fig. 17.3 were implemented as is, the resulting complexity would be quite high. In later sections, computationally much more efficient but mathematically equivalent encoder structures will be introduced. Most analysis-by-synthesis speech coders today are implemented based on the more efficient encoder structures.

17.3 Overview of Prominent Analysis-by-Synthesis Speech Coders

This section gives an overview of some prominent analysis-by-synthesis speech coders and shows the relationship between them in the form of a *family tree*. There are numerous analysis-by-synthesis speech coders proposed in the literature. It is not practical to describe all of them in this chapter. Instead, the intention here is to describe only those analysis-by-synthesis speech coders that are either standard coders or are the first of its kind and thus are definitive and seminal.

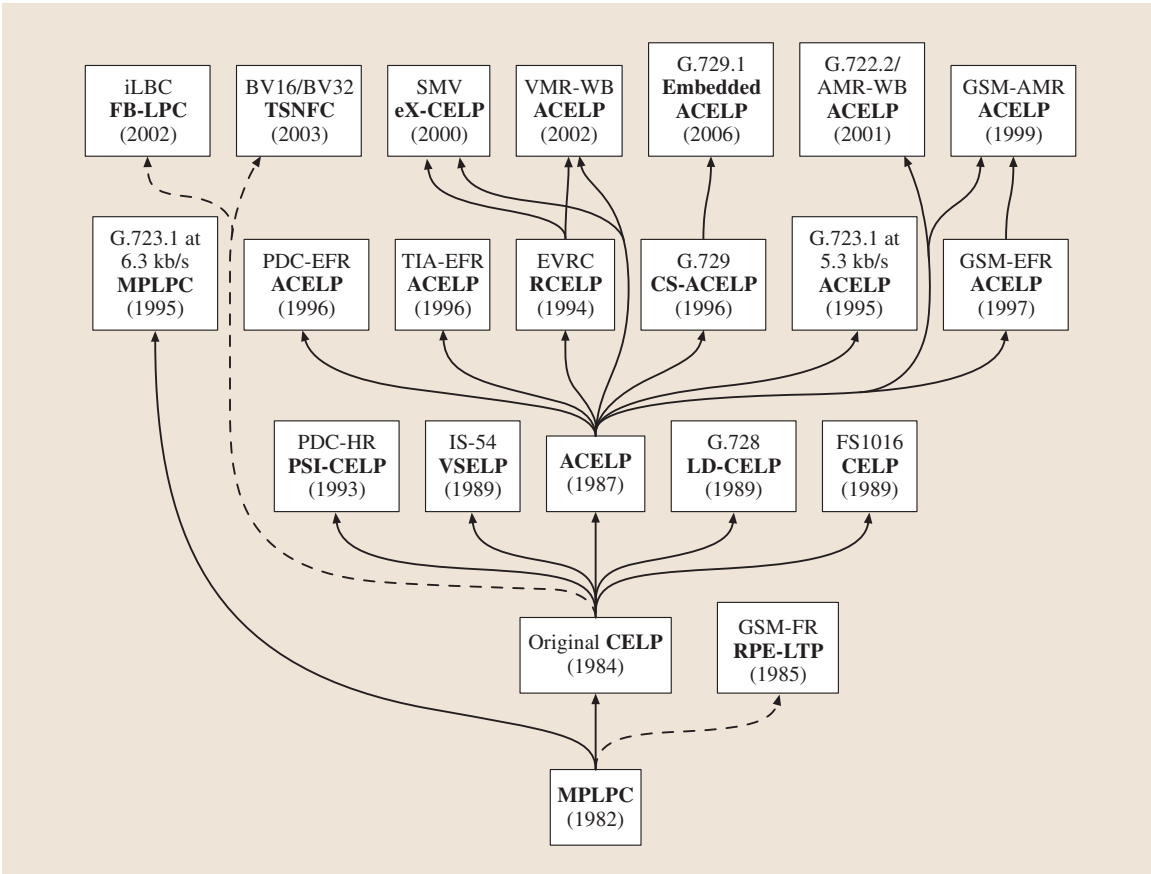

Fig. 17.4 Family tree of prominent analysis-by-synthesis speech waveform coders

Figure 17.4 shows the family tree of these selected analysis-by-synthesis speech coders. Each rectangular block in the tree represents a particular speech coder or a type of speech coders. Within each rectangular block, the first row (sometimes the first two rows) specifies the name of the coding standard, the second row in boldface letters gives the name of the coding technique, and the third row provides the year of the first publication the authors can find for that coder (either as a technical paper or as a standard specification).

The speech coders in Fig. 17.4 are organized in five different rows, with each row roughly corresponding to a different generation of coders. Each generation represents a group of analysis-by-synthesis speech coders developed within a time period of a few years.

The MPLPC coder [17.2] proposed in 1982 is the first modern analysis-by-synthesis speech waveform coder. The original CELP coder [17.3] and the RPE-LTP coder [17.18] in the second row represent the second-generation analysis-by-synthesis coders developed in the mid 1980s. The five coders in the third row represent the third-generation analysis-by-synthesis coders developed in the late 1980s and early 1990s. The seven coders in the fourth row represent the fourth-generation analysis-by-synthesis coders developed in the mid 1990s. Finally, the seven coders in the fifth row represent the fifth-generation analysis-by-synthesis coders developed from the late 1990s to the present day. Strictly speaking, the distinctions between the fourth- and fifth-generation coders are not always clear; however, at least the fifth-generation coders were developed later than the fourth-generation coders.

A brief overview of this family tree of analysis-by-synthesis speech coders is given below. The MPLPC coder proposed in 1982 is at the root of the family tree, since it can be argued that all other coders in Fig. 17.4 can trace their roots back to this coder.

For the second-generation coders, the original CELP coder proposed in 1984 is a direct descendant of MPLPC with long-term prediction [17.19], as *Atal* worked on both coders and the two coders are similar except the multipulse model is replaced by VQ of the excitation in CELP. The excitation VQ codebook of this CELP coder is populated by Gaussian random numbers. The Groupe Spèciale Mobile (GSM) full-rate standard coder regular-pulse excitation with long-term prediction (RPE-LTP) [17.18] is inspired by the MPLPC coder; however, to reduce the computational complexity, it uses an excitation model of regularly spaced pulses which deviates substantially from the original multipulse model. That is why the arrow from MPLPC to RPE-LTP has a dashed line.

The third-generation coders saw the most variety of analysis-by-synthesis excitation coding techniques. The first standard coder based on CELP is the US federal standard FS1016 CELP coder at 4.8 kb/s [17.20]. It uses overlapping center-clipped Gaussian VQ codevectors for excitation coding. The second CELP coder that became a standard coder is the Telecommunications Industry Association (TIA) IS-54 standard vector sum excited linear prediction (VSELP) coder at 8 kb/s [17.17]. It uses sums of several basis vectors as the excitation VQ codevectors. A 6.7 kb/s VSELP coder later also became the Japanese personal digital cellular (PDC) full-rate standard coder, and a 5.6 kb/s VSELP coder became the GSM half-rate standard coder [17.21]. The third CELP coder that became a standard is the ITU-T G.728 standard low-delay CELP (LD-CELP) coder at 16 kb/s [17.6]. It uses a closed-loop trained-gain-shape VQ codebook structure with a small vector dimension and with joint optimization of VQ gain and shape. The fourth CELP coder that became a standard is the Japanese PDC half-rate standard pitch synchronous innovation CELP (PSI-CELP) coder at 3.45 kb/s [17.22]. This coder uses a fairly long vector dimension and repeats the first portion of the VQ codevectors in a pitch-synchronous manner if the pitch period is less than the vector dimension.

An influential class of third-generation coder is the algebraic CELP, or ACELP [17.23]. This coder reduces the VQ codebook search complexity by using an algebraic VQ codebook, with the elements of VQ codevectors having only $+1$, 0, and -1 as the possible sample values. As can be seen from Fig. 17.4, most of the fourth- and fifth-generation coders are derivatives of ACELP. However, it should be noted that ACELP-based speech coding standards use sparse algebraic codes [17.24] and not the original algebraic code proposed in [17.23]. The sparseness contributes to a low computational complexity and the reduction of granular type of coding noise.

All but one fourth-generation standard coders in Fig. 17.4 can trace their roots to ACELP. The single exception is the 6.3 kb/s version of the ITU-T G.723.1 standard [17.25]. It is an improved version of the first-generation MPLPC coder. One of the fourth-generation coders that deserves attention is the so-called relaxed CELP (RCELP) coder used in North American cellular standard enhanced variable-rate coder (EVRC) [17.26]. It is based on a generalization of analysis-by-synthesis [17.15] that adaptively modifies the input speech signal in such a way that the modified signal sounds essentially the same as the original input signal and yet it becomes easier for a CELP coder to encode the signal efficiently. This RCELP technique is also used in the North American cellular standard selectable mode vocoder (SMV) [17.27] and variable-rate multimode wide-band (VMR-WB) coder [17.28].

For fifth-generation standard coders in Fig. 17.4, all but two are based on ACELP. The exceptions are the BV16/BV32 coder [17.29], [17.30] and the internet low bit rate (iLBC) coder [17.31], [17.32]. The iLBC coder is an Internet Engineering Task Force (IETF) experimental standard and also a PacketCable standard. It is based on what is called forward backward linear predictive coding (FB-LPC), where a maximum-energy segment of the short-term prediction residual is independently coded, and then a dynamic codebook is constructed first forward in time and then backward in time. The 16 kb/s BroadVoice16 (BV16) coder [17.33] and its wide-band version 32 kb/s BroadVoice32 (BV32) coder use similar coding algorithms. BV16 is a PacketCable standard, SCTE (Society of Cable Telecommunications Engineers) standard, and American National Standards Institute (ANSI) standard for voice-over-internet protocol (VoIP) applications in cable telephony, and BV32 is a PacketCable 2.0 standard. The BV16/32 coder is based on two-stage noise feedback coding (TSNFC). This coder is not a CELP coder since its encoder structure is totally different from that of CELP; however, it employs many of the complexity reduction and even codebook design techniques of CELP within the TSNFC encoder structure. For this reason, the arrow from the original CELP to TSNFC has a dashed line, signifying that it is not a direct descendant of CELP but leverages nearly two decades of research on CELP.

As discussed above, recent speech coding standards are dominated by ACELP-related coders. For this

reason, ACELP and its derivatives will receive more-thorough treatment in this chapter. Nevertheless, there are still many other interesting kinds of analysis-by-synthesis coders in this family tree. In the sections that follow, the excitation coding methods for the more distinctive types of coders in this family tree will be described in more detail. Due to the similarity between many different ACELP coders in Fig. 17.4, many of them are lumped together and discussed in the same ACELP section.

17.4 Multipulse Linear Predictive Coding (MPLPC)

The MPLPC coder [17.2] was the first predictive waveform coder to abandon the sample-by-sample residual quantization procedure and to encode the entire block of excitation signal as a single entity in an analysis-by-synthesis manner. It achieves this by using an excitation signal with mostly zero samples and finding the optimal locations and amplitudes for a small number of nonzero pulses such that the resulting synthesized speech signal minimizes a weighted distortion measure in the speech domain.

Refer to the generic analysis-by-synthesis speech waveform coder shown in Fig. 17.3. The MPLPC coder does not use the signal modifier in Fig. 17.3. The original MPLPC coder proposed in [17.2] also does not use the long-term synthesis filter. In fact, this original MPLPC coder was developed to improve the linear predictive coding (LPC) vocoder [17.34], which does not use a long-term synthesis filter and have a rigid excitation model of either periodic pulse train or white noise.

The MPLPC coder does use the short-term synthesis filter in Fig. 17.3. This short-term synthesis filter is in the form of an all-pole filter given by the following transfer function

$$H_s(z) = \frac{1}{A(z)} , \tag{17.1}$$

where

$$A(z) = 1 - P(z) = 1 - \sum_{i=1}^{M} a_i z^{-i} \tag{17.2}$$

is the short-term prediction error filter, $P(z)$ is the short-term predictor, $a_i, i = 1, 2, \ldots, M$, are the predictor coefficients, and M is the order of the short-term synthesis filter. The filter order M is typically somewhere between 10 and 16 for 8 kHz sampled signals. The predictor coefficients are adaptive and are usually transmitted once every 10–20 ms.

In this first MPLPC coder the weighted error minimization in Fig. 17.3 is achieved through the use of a so-called perceptual weighting filter in the form of

$$W(z) = \frac{A(z)}{A(z/\gamma)} , \quad 0 < \gamma < 1 , \tag{17.3}$$

where

$$A\left(\frac{z}{\gamma}\right) = 1 - P\left(\frac{z}{\gamma}\right) = 1 - \sum_{i=1}^{M} a_i \gamma^i z^{-i} \tag{17.4}$$

is a modified version of $A(z)$ with the roots of the polynomial moved radially toward the origin (the radii of the roots are scaled by a factor of γ, which is typically 0.8). Since the roots are normally within the unit circle for a stable synthesis filter $H_s(z) = 1/A(z)$, the net effect is that the magnitude frequency response of the filter $1/A(z/\gamma)$ is a smoothed version of that of the filter $1/A(z)$.

For MPLPC, the weighted error minimization in Fig. 17.3 is achieved by passing the difference between the input speech signal and the synthesized speech signal through the perceptual weighting filter $W(z)$ and then identifying which combination of multipulse model parameters gives the minimum mean-squared error of this perceptually weighted difference signal. The perceptual weighting filter emphasizes the spectral valley regions and deemphasize the spectral peak regions of the input speech spectrum. This has an effect of allowing more coding noise under the spectral peaks and suppressing coding noise in the spectral valleys of speech. Thus, the perceptual weighting filter shapes the coding noise spectrum so that it follows the input speech spectrum to some extent. Due to the noise masking effect of the human auditory system, such a spectrally shaped coding noise is perceptually less audible than coding noise with a flat spectrum.

The short-term synthesis filter $H_s(z) = 1/A(z)$ has a magnitude frequency response corresponding to the spectral envelope of the input signal. Thus, the filter $H_s(z/\gamma) = 1/A(z/\gamma)$ has a magnitude response corresponding to a smoothed version of the spectral envelope of input speech. Subtracting one magnitude response

from another in the logarithmic domain is equivalent to dividing in the linear domain. Hence, the filter

$$\frac{1}{W(z)} = \frac{A(z/\gamma)}{A(z)} = \frac{H_s(z)}{H_s(z/\gamma)} \tag{17.5}$$

will have a magnitude response somewhat similar to the spectral envelope of the input speech but with a reduced spectral slope. Since this filter is the inverse of the perceptual weighting filter $W(z)$, given how the weighted error minimization works as described above, the magnitude response of this filter $1/W(z)$ is the spectral envelope of the coding noise that the MPLPC coder will achieve. For an example of what the magnitude responses of $H_s(z)$ and $W(z)$ look like, see [17.2].

The first MPLPC coder in [17.2] encoded the excitation signal block-by-block in an analysis-by-synthesis manner using a block size of 5 ms, or 40 samples at 8 kHz sampling. Most of the 40 samples are zeroes. Only about 10% of the 40 excitation samples (four samples) have nonzero amplitudes. These nonzero samples are called *pulses* because they look like pulses in the graphical representation of such a digital excitation signal. Since there are usually multiple of such nonzero samples in the block of excitation signal to be encoded, the resulting excitation model is called the *multipulse* model, and the resulting linear predictive coding scheme is called *multipulse linear predictive coding* (MPLPC).

Let m be the number of pulses in each block of excitation signal. Each pulse is completely specified by its amplitude and its location within the block. Therefore, the multipulse excitation model has $2m$ model parameters that need to be determined. Jointly optimizing these $2m$ model parameters would cost too high computational complexity. A suboptimal sequential optimization approach is proposed in [17.2].

In the sequential optimization approach, the m pulses are determined one at a time. For a given pulse location, the corresponding optimal amplitude can be obtained through a closed-form solution [17.2]. Before determining the location and amplitude of the first pulse, with the excitation signal set to zero, the output of the short-term synthesis filter during the current block (due to the nonzero filter memory left over at the end of the last block) is subtracted from the input speech. The result is the target signal for the search of the first pulse. Each of the 40 possible pulse locations is tried. With closed-loop optimal amplitude solved for each pulse location, the pulse is passed through the short-term synthesis filter. The filter output signal is subtracted from the target signal and the resulting difference signal is passed through the perceptual weighting filter. The mean-squared error (MSE) of the weighted signal is calculated. This process is repeated until the best pulse location and amplitude that gives the lowest weighted error is identified. Next, the short-term synthesis filter output signal due to this first pulse is subtracted from the target signal to form a new target signal for the search of the second pulse. This process is repeated until the pulse locations of all m pulses are determined. Then, given the m pulse locations, the m pulse amplitudes can be jointly optimized in a single step [17.2].

Using this approach, it is reported in [17.2] that only minimal audio quality improvement are achieved after about eight pulses have been placed in a 10 ms interval. It is also reported [17.2] that the resulting output speech quality sounded natural and perceptually close to the input speech, without the common unnatural and buzzy characteristics of the LPC vocoder output speech.

Strictly speaking, since each pulse in the multipulse model is scaled by a different amplitude, the decomposition of the excitation generator into the excitation shape generator and the excitation gain in Fig. 17.3 is not a good description. This problem can be avoided if the excitation gain in Fig. 17.3 is understood to have the possibility of taking the form of a gain vector (with dimension m), where each element of the gain vector is individually used to scale one of the m pulses.

Further improvements to this initial MPLPC coder was proposed in [17.19]. The most notable is the addition of a pitch predictor, or equivalently, the addition of the long-term synthesis filter in Fig. 17.3. Due to the quasiperiodicity in voiced speech such as vowels, it is found that the multipulse excitation signal shows significant correlation from one pitch period to the next. This correlation can be exploited by using a pitch predictor (also called long-term predictor). Let the pitch period be T samples, and let β be the long-term predictor coefficient. Then, with

$$H_l(z) = \frac{1}{1-\beta z^{-T}} \tag{17.6}$$

chosen as the transfer function of the long-term synthesis filter in Fig. 17.3, each pulse at the output of the long-term synthesis filter will be scaled by β and then be added to the excitation signal T samples later. This has the effect of creating a periodic pulse pattern at a period of T samples. After adding this long-term synthesis filter, fewer pulses are needed to produce the same level of speech quality.

Another improvement proposed in [17.19] is related to pulse amplitude optimization. Rather than waiting until all pulse locations are determined and then jointly

optimize all the pulse amplitudes, the proposed procedure performs reoptimization of amplitudes along the way. In other words, during the sequential search for pulse locations, after the j-th pulse location is determined ($1 \le j \le m$), all pulse amplitudes from the first pulse to the j-th pulse are reoptimized jointly before continuing to search for the $(j+1)$-th pulse location. It is reported [17.19] that this approach improved coder output signal-to-noise ratio (SNR) by 2–10 dB, with an average of slightly over 3 dB.

There are many other improvements and variations proposed in the literature for the MPLPC coder. Due to the space limitation, they have to be omitted in the discussion here.

The multipulse excitation model is used in the ITU-T Recommendation G.723.1 speech coder. G.723.1 has two bit-rates. The lower-bit-rate 5.3 kb/s version is based on ACELP, while the higher-bit-rate 6.3 kb/s version is based on MPLPC with long-term prediction.

This 6.3 kb/s version of G.723.1 uses a frame size of 30 ms, which is equally divided into four subframes of 7.5 ms (60 samples) each. Multipulse excitation search is performed on each of the four subframes. The multipulse excitation model uses six pulses for even subframes and five pulses for odd subframes. The pulse location is restricted to be either all even or all odd, as indicated by a grid bit. Furthermore, all pulses are constrained to have the same magnitude, although different pulses can have different signs. The shared magnitude is first estimated and quantized. Then, four quantized magnitude values around the estimated magnitude are allowed in the analysis-by-synthesis multipulse search. For each of these four possible quantized magnitude values, the pulse locations and signs are sequentially optimized one pulse at a time. This procedure is repeated for both the even and odd pulse position grids. Finally, the best combination of all these parameters (quantized magnitude, pulse locations and signs, and even or odd grid) that gives the minimum weighted error of the synthesized speech is selected as the final multipulse excitation parameters to be transmitted to the G.723.1 decoder.

As can be seen, this G.723.1 multipulse coder has deviated substantially from the original MPLPC proposed in [17.2] and [17.19]. In fact, with all the constraints put on the G.723.1 multipulse model (due to the low-bit-rate limitation), the G.723.1 multipulse excitation signal starts to resemble to some extent the excitation signal of ACELP coders. However, such constraints aside, the spirit of the original multipulse excitation model can still be seen in this 6.3 kb/s G.723.1 multipulse coder.

17.5 Regular-Pulse Excitation with Long-Term Prediction (RPE-LTP)

The regular-pulse excitation (RPE) coder [17.35] can be regarded as a special low-complexity realization of the fundamental analysis-by-synthesis concept proposed in the original multipulse LPC coder [17.2]. A special version of it, the regular-pulse excitation with long-term prediction (RPE-LTP) coder at 13 kb/s [17.18], was selected as the first GSM standard coder for European digital cellular service.

The RPE excitation model consists of J possible sequences of regularly spaced pulses, each with the pulses located at a different phase. The typical value of J is 3 or 4. As an example, for the k-th sequence of pulses, the nonzero excitation samples (the pulses) are located at the positions of $k, k+J, k+2J, k+3J, \ldots$, and the sample values at other locations are zero. The task of excitation coding is to find the amplitude values for each of the pulses in each of the J sequences of regularly spaced pulses, and then find the sequence of pulses that minimizes a weighted error measure.

Similar to the original MPLPC coder proposed in [17.2], the initial RPE coder [17.35] also did not use a long-term predictor. On a conceptual level, the encoder structure of the initial RPE coder is equivalent to the structure in Fig. 17.3 without the long-term synthesis filter and the signal modifier. However, in actual implementation, the LPC inverse filter $A(z)$, which is the numerator portion of the perceptual weighting filter $W(z) = A(z)/A(z/\gamma)$, is moved beyond the adder to the left and above. The one to the left of the adder cancels out the short-term synthesis filter, while the one above the adder stays there and filters the input speech to produce the short-term prediction residual. The remaining denominator portion of the weighting filter, or $1/A(z/\gamma)$, stays below the adder for weighted error minimization.

In [17.35], the procedure for the RPE analysis-by-synthesis excitation search is given. It basically amounts to solving J sets of linear equations to find the optimal amplitude for each pulse of the J possible regular pulse sequences. The weighted distortion measure for each of the J pulse sequences are calculated and the sequence that minimizes the weighted distortion is selected as the final excitation sequence.

In the GSM full-rate (GSM-FR) standard RPE-LTP coder, a single-tap long-term predictor (LTP) is added to the initial RPE coder, so GSM-FR corresponds to the coder structure in Fig. 17.3 with the long-term synthesis filter enabled. The GSM-FR coder uses three possible sequences of regularly space pulses, that is, $J = 3$. It has a frame size of 20 ms and a subframe size of 5 ms. Therefore, for every 5 ms subframe of 40 samples, there are 3 possible sequences of regularly spaced pulses. Each sequence contains only 13 or 14 nonzero pulses, with the remaining samples having zero amplitudes. The selected sequence is identified by 2 bits, and the corresponding pulse amplitudes are encoded with a 3-bit block-adaptive PCM quantizer. The block maximum for each subframe is encoded with 6 bits. The pitch period and the pitch tap are derived once a subframe and quantized to 7 bits and 2 bits, respectively.

This GSM-FR RPE-LTP coder is the speech coder for the first-generation GSM digital cellular telephones. It became the first analysis-by-synthesis coder that was standardized and widely deployed across many countries.

17.6 The Original Code Excited Linear Prediction (CELP) Coder

By using an analysis-by-synthesis procedure, the multipulse excitation model in the MPLPC coder [17.2] opened the door for significantly more efficient excitation coding than was possible with earlier speech waveform coders. However, even with this multipulse model, there is still a limit on how low the excitation encoding bit-rate can go. For example, with the same 40-sample excitation block size and four pulses as in the original MPLPC, suppose it is desirable to encode the excitation signal at a bit rate of 1/4 bits/sample, this gives a mere 10 bits to represent the four pulse locations and four pulse amplitudes. It is virtually impossible to use 10 bits to quantize such eight parameters with sufficient accuracy. This problem led *Atal* and *Schroeder* to use a 10 bit vector quantization (VQ) codebook to quantize the excitation signal [17.3], leading to what is known today as code-excited linear prediction, or CELP [17.4].

To use 10-bit VQ to quantize 40 samples of the excitation signal, Atal and Schroeder constructed a VQ codebook containing $2^{10} = 1024$ codevectors, each of which is a 40-dimensional vector containing 40 white Gaussian random numbers with unit variance. Beside conceding that it is generally difficult to design an optimum deterministic codebook (with a total of 40 960 samples in it), in [17.4] *Schroeder* and *Atal* actually gave a justification for using the Gaussian random numbers. The justification is that the probability density function of the prediction residual signal after both short- and long-term prediction is nearly Gaussian.

The encoder structure of this initial CELP coder fits the structure shown in Fig. 17.3 exactly. Again, the signal modifier in Fig. 17.3 is not used in this CELP coder. The excitation shape generator is basically a table look-up of the 10-bit, 40-dimensional excitation VQ codebook described earlier. The excitation gain scales the unit-variance Gaussian codevectors to the proper gain level that matches the root mean square (RMS) value of the prediction residual signal after both short-term and long-term prediction. It is updated once every 5 ms together with the excitation codevector.

The synthesis filter of CELP contains both the long-term synthesis filter and the short-term synthesis filter. The short-term synthesis filter has the same form as given in (17.1), while the long-term synthesis filter uses a 3-tap pitch predictor rather than the single-tap pitch predictor given in (17.6). The weighted error minimization of CELP is also achieved with a perceptual weighting filter in exactly the same way as in MPLPC, described earlier in Sect. 17.4. The perceptual weighting filter $W(z)$ also has exactly the same form as defined in (17.3).

Atal and *Schroeder* reported [17.3, 4] that when the 40-sample excitation vector is quantized to 0.25 bit/sample in an analysis-by-synthesis manner using a 10-bit Gaussian VQ codebook, with all other speech synthesis model parameters (excitation gain and parameters for the long- and short-term synthesis filters) left at their open-loop-derived optimal values, the resulting synthesized speech sounded very close to the original input speech. Only small differences were noticeable even in close pairwise comparisons over headphones.

Achieving such a high level of output speech quality with such a low bit rate of merely 0.25 bit/sample for the excitation signal was considered a stunning breakthrough in 1984–1985. Almost immediately, a large number of speech coding researchers jumped in and engaged in the research of the CELP coding technique. In the subsequent years that followed, hundreds of CELP-related technical papers have been published and numerous advancements in CELP coding have been

made. As shown in the family tree of Fig. 17.4, this analysis-by-synthesis CELP coding idea eventually led to more than a dozen of modern low-bit-rate speech coding standards, and CELP-related coders dominate almost all speech coding standards established since the late 1980s.

The two initial CELP papers [17.3, 4] were more *proof of concept* papers than actual coder design papers. Not only were all model parameters except the excitation signal were left unquantized, but a major issue was the very high computational complexity required by the analysis-by-synthesis excitation VQ codebook search.

Actually, it is fairly easy to estimate the complexity of such a codebook search. Refer to Fig. 17.3. Given that the short-term filter order they used was $M = 16$, each of the 1024 excitation codevectors has to be scaled by the excitation gain (one multiply per sample), filtered by the long-term synthesis filter (three multiply-adds per sample), filtered by the short-term synthesis filter (16 multiply–adds per sample), subtracted from the input speech (one subtract per sample), filtered by the perceptual weighting filter ($2 \times 16 = 32$ multiply–adds per sample), squared (one multiply per sample), and then added (one add per sample) to get the weighted distortion value of that codevector. Therefore, at a sampling rate of 8000 Hz, the total complexity for searching through all 1024 excitation codevectors to identify the one that minimizes the weighted distortion measure would take at least $8000 \times 1024 \times (1+3+16+1+32+1+1) = 450\,560\,000$ operations per second, or 450.6 MFLOPS (million floating-point operations per second). Even the fastest supercomputer at that time could not perform computations that fast. No wonder *Atal* and *Schroeder* reported [17.3, 4] that their initial CELP simulation took 125 s of central processor unit (CPU) time on the then-supercomputer Cray-1 to process just one second of speech.

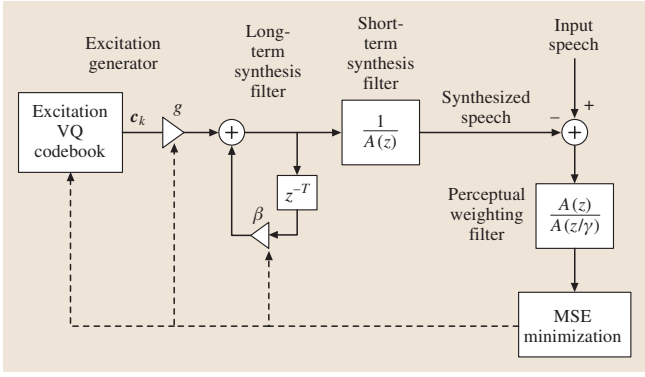

Fig. 17.5 Encoder structure of the original CELP coder

However, even *Atal* and *Schroeder* pointed out [17.4] at that time that

> *a code book with sufficient structure amenable to fast search algorithms could lead to real-time implementation of code-excited coders.*

This turned out to be true. In subsequent years many specially structured excitation VQ codebooks that allowed fast codebook search were proposed – the excitation codebooks of ACELP, VSELP, and FS1016 CELP are all good examples.

However, even without such specially structured excitation codebooks, by just re-arranging the encoder structure in Fig. 17.3 and performing the CELP excitation codebook search in a mathematically equivalent way, the computational complexity can be reduced by almost an order of magnitude. This efficient encoder structure and excitation codebook search procedure is explained below. It forms the basis of essentially all practical CELP-based coders today.

Consider the encoder structure shown in Fig. 17.5. This structure is basically the same as the generic analysis-by-synthesis coder structure shown in Fig. 17.3, except that specific filter structure and transfer functions are given. For convenience of later discussion of the so-called *adaptive codebook*, the long-term synthesis filter is reverted from a three-tap filter back to a single-tap filter as used in the improved MPLPC [17.19]. The short-term synthesis filter is $1/A(z)$. The weighted error minimization is explicitly separated into a perceptual weighting filter as defined in (17.3) followed by MSE minimization. Even with the three-tap pitch filter replaced by a single-tap one, the complexity is still $8000 \times 1024 \times (1+1+16+1+32+1+1) = 434.2$ MFLOPS.

As suggested in [17.19], the perceptual weighting filter can be moved before the adder in Fig. 17.5 so that the input speech and the synthesized speech are each individually weighted before the difference of the two weighted signals are calculated. The cascade of the short-term synthesis filter and the perceptual weighting filter gives a weighted short-term synthesis filter in the form of

$$H(z) = \frac{W(z)}{A(z)} = \frac{1}{A(z)} \frac{A(z)}{A(z/\gamma)} = \frac{1}{A(z/\gamma)} \,. \qquad (17.7)$$

This is shown in Fig. 17.6. Even this step alone can cut the computational complexity by almost a factor of three. There is more that can be saved. However, before

continuing, let us briefly describe another new feature in Fig. 17.6 – the concept of the adaptive codebook as introduced in [17.36].

The adaptive codebook in Fig. 17.6 is a different way to achieve a similar effect as the long-term synthesis filter in Fig. 17.5. If the pitch period T is not smaller than the excitation vector dimension, then once the effect of the filter memory is taken out, the long-term synthesis filter in Fig. 17.5 does not affect the excitation codebook search at all due to the bulk delay z^{-T}. However, if the pitch period T is smaller than the vector dimension, then different excitation VQ codevectors will cause different contributions from the long-term synthesis filter to be added to the excitation to the short-term synthesis filter. This makes it more difficult to perform certain efficient codebook search methods.

To facilitate an efficient codebook search, the effect of the long-term synthesis filter in Fig. 17.5 is modeled by the adaptive codebook in Fig. 17.6. If the pitch period is not smaller than the excitation vector dimension, then the adaptive codebook simply contains different vector sections of the long delay line in the long-term synthesis filter defined by the range of the pitch period. In other words, the codevectors in the adaptive codebook is obtained by using a sliding rectangular window to extract different sections of the long delay line in the long-term synthesis filter, with different codevectors corresponding to different target pitch periods. In this case, the adaptive codebook, the adaptive codebook gain β, and the related adder in Fig. 17.6 is mathematically equivalent to the long-term synthesis filter in Fig. 17.5.

On the other hand, if the pitch period is smaller than the vector dimension, then at least a portion of the codevectors in the adaptive codebook will be beyond the long delay line of the long-term synthesis filter and will correspond to the samples in the current excitation vector that have not been determined yet. Rather than having incomplete codevectors, the adaptive codebook will periodically repeat the samples in the long delay line at the target pitch period.

Thus, the concept of adaptive codebook allows the use of large excitation vector dimension which tends to improve the coding efficiency. In addition, it allows the long-term synthesis filter to be modeled as just another stage of excitation VQ, thus simplifying the codebook search procedure.

The codevectors in an adaptive codebook is changing with time. That is why the codebook is called the *adaptive* codebook. In contrast, the original excitation VQ codebook in Fig. 17.5 does not change with time. Therefore, as shown in Fig. 17.6, it is often called the *fixed* codebook in a CELP coder where an adaptive codebook is used.

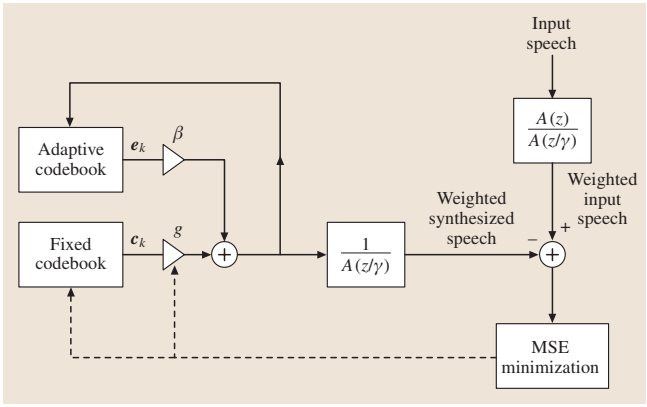

Fig. 17.6 Intermediate encoder structure of the efficient CELP coder

The encoder structure in Fig. 17.6 can be changed to an even more efficient but mathematically equivalent structure as shown in Fig. 17.7. According to the linear system theory, the output of the filter $H(z)$ is the sum of two components

1. the zero-state response (ZSR), which is the output of the filter due to the input signal, with the initial filter memory set to zero,
2. the zero-input response (ZIR), which is the output of the filter due to only the filter memory, with the input signal set to zero.

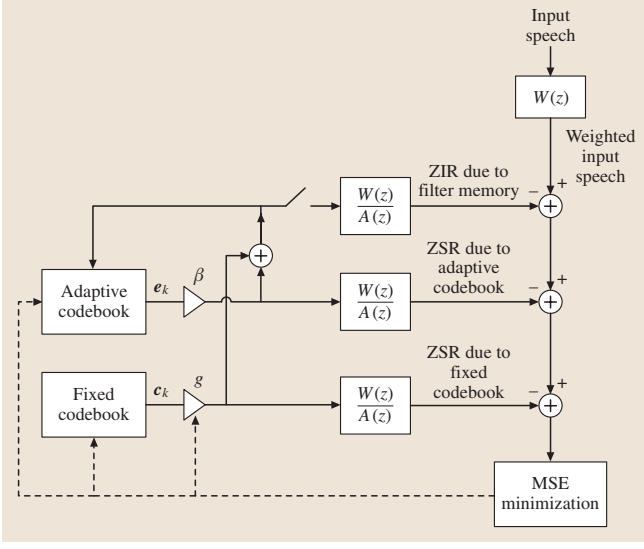

Fig. 17.7 Encoder structure of the efficient CELP coder

Note that the ZIR does not depend on which excitation VQ codevector is used. Thus, by decomposing the output of the weighted short-term synthesis filter $H(z)$ into ZIR and ZSR, the ZIR component can first be subtracted out of the weighted speech signal, and the resulting signal becomes the *target signal* for the codebook search based on the ZSR component. In fact, this idea was already suggested by *Atal* and *Remde* in their original MPLPC paper [17.2], even though they did not use the terms of ZIR and ZSR.

In some later CELP coders, the perceptual weighting filter took a form different from what was specified in (17.3). Therefore, to keep Fig. 17.7 general, the weighted short-term synthesis filter is represented as $H(z) = W(z)/A(z)$.

In Fig. 17.6, the excitation signal to the short-term synthesis filter has two components: the scaled adaptive codebook vector $\beta \boldsymbol{e}_k$ and the scaled fixed codebook vector $g\boldsymbol{c}_k$. Therefore, the weighted synthesized speech signal in Fig. 17.6 can be decomposed into three components: the ZIR signal due to the memory of the filter $W(z)/A(z)$, the ZSR signal due to the adaptive codebook, and the ZSR signal due to the fixed codebook.

In the efficient CELP excitation codebook search method based on Fig. 17.7, the input speech vector is first passed through the perceptual weighting filter $W(z)$ to get the weighted input speech vector. Then, the ZIR vector due to the filter memory is calculated by setting the initial memory of the weighted short-term synthesis filter to the filter memory at the last sample of the last input speech vector and letting the filter *ring* without any input excitation signal (the *switch* in Fig. 17.7 is open). This ZIR vector is subtracted from the weighted input speech vector to get the target vector for the next stage. Next, the adaptive codebook index (the pitch period) and the adaptive codebook gain (the pitch gain) are usually determined in an analysis-by-synthesis manner to minimize the MSE between the target vector obtained above and the ZSR vector due to the adaptive codebook. The ZSR vector corresponding to the best combination of the pitch period and the pitch gain is then subtracted from the target vector to get the next target vector for the ZSR vector due to the fixed codebook. Finally, the fixed codebook index and the fixed codebook gain are determined in an analysis-by-synthesis manner to minimize the MSE between this next target vector and the ZSR vector due to the fixed codebook. After such a codebook search, the selected codebook vectors and gains are used to calculate the sum of the two excitation components $\beta \boldsymbol{e}_k + g\boldsymbol{c}_k$, and the resulting final excitation vector is used to update the adaptive codebook and the memory of the weighted short-term synthesis filter (the *switch* is closed). Then, this procedure is repeated for the next input speech vector.

The encoder structure in Fig. 17.7 looks much more complicated than the structure in Fig. 17.5, so why is it computationally more efficient? The key lies in the fact that by subtracting out the ZIR vector due to filter memory, the search through the adaptive codebook and the fixed codebook can be performed without the actual filtering operation. Let $h(n), n = 0, 1, \ldots, N-1$ be the truncated impulse response of the weighted short-term synthesis filter $H(z) = W(z)/A(z)$ with N being the dimension (length) of the excitation vector. Then, with the initial filter states (memory) set to zero in the lower two branches in Fig. 17.7 containing the filter $W(z)/A(z)$, the filtering operation in these two branches can be replaced by convolution, which can be represented by a matrix–vector multiplication operation [17.37, 38].

Take the fixed codebook search as an example. Let the target vector for the fixed codebook search be $\boldsymbol{t} = [t(0), t(1), \cdots, t(N-1)]^{\mathrm{T}}$, let the k-th fixed codebook vector be $\boldsymbol{c}_k = [c_k(0), c_k(1), \cdots, c_k(N-1)]^{\mathrm{T}}$, and let

$$\boldsymbol{H} = \begin{pmatrix} h(0) & 0 & \cdots & 0 \\ h(1) & h(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h(N-2) & h(N-3) & \cdots & 0 \\ h(N-1) & h(N-2) & \cdots & h(0) \end{pmatrix} \tag{17.8}$$

be the lower triangular Toeplitz matrix populated by the impulse response of the weighted short-term synthesis filter $H(z) = W(z)/A(z)$. Then, the ZSR vector due to the k-th fixed codebook vector can be represented as $g\boldsymbol{H}\boldsymbol{c}_k$, and the codebook search find the codebook index k that minimizes the MSE distortion measure

$$E_k = \|\boldsymbol{t} - g\boldsymbol{H}\boldsymbol{c}_k\|^2 , \tag{17.9}$$

where $\|\cdot\|^2$ indicates the Euclidean norm, or the sum of squares of the vector components. This distortion measure can be expanded as

$$E_k = \|\boldsymbol{t}\|^2 - 2g\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k + g^2\|\boldsymbol{H}\boldsymbol{c}_k\|^2 . \tag{17.10}$$

Since the energy of target vector $\|\boldsymbol{t}\|^2$ is independent of the codebook index k, this term can be ignored when minimizing the distortion measure above. Thus, calculating the distortion measure amounts to calculating the energy of the ZSR vector $g\boldsymbol{H}\boldsymbol{c}_k$ and the correlation between this ZSR vector and the target vec-

tor $\boldsymbol{t}$. The adaptive codebook search can be formulated in the same way using the same kind of distortion measure.

Many efficient codebook search procedures have been proposed based on the distortion measure in (17.10). In addition, many special fixed codebook structures that allow efficient codebook search have been proposed in the literature. Together these techniques enabled the computational complexity to be reduced from the 430+ MFLOPS of the original CELP to 10 MFLOPS and below for many of the practical CELP coders deployed today.

17.7 US Federal Standard FS1016 CELP

The US Federal Standard FS1016 coder [17.20] is the first CELP coder ever standardized. It was developed by the US Department of Defense and is quite similar to the stochastically excited linear prediction (SELP) coder proposed in [17.36]. This coder uses a frame size of 30 ms. Each frame is divided into four subframes of 7.5 ms each (60 samples at 8 kHz sampling). The adaptive codebook approach described in [17.36] is used to determine the pitch period (also called the *pitch lag*) and the pitch gain once every subframe. (In this case, there is one excitation vector in each subframe.) For odd-numbered subframes, the pitch lag takes an integer value between 20 and 147 and thus can be encoded into 7 bits. For even-numbered subframes, the pitch lag is constrained to be within 32 samples relative to the pitch lag of the previous subframe. The pitch gain is coded using a five-bit nonuniform scalar quantizer. For the fixed codebook, the FS1016 standard uses a specially structured codebook where adjacent codevectors are essentially shifted version of each other and differ by only two samples at the end. This kind of shifted fixed codebook was first proposed in [17.39]. However, it was also independently studied in [17.36] using an improved distortion measure, and the optimal tradeoff of the two-sample shift was evaluated and proposed in [17.36].

This shifted fixed codebook introduces a certain constraint and structure into the codebook and allows an efficient codebook search. This is because when adjacent fixed codebook vectors are shifted version of each other, a large part of the ZSR computations in the matrix-vector multiplication $\boldsymbol{Hc}_k$ can be reused when going from one codevector to the next. Therefore, once the ZSR is calculated for the first codevector, the ZSR of the remaining codevectors can be obtained recursively with low complexity. This same principle also applies to the adaptive codebook search for those pitch lags that are not smaller than the excitation vector dimension (subframe size), since the corresponding adjacent adaptive codevectors are simply 1-sample shifted versions of each other.

To illustrate how a shifted codebook can reduce the codebook search complexity, first consider a fixed codebook with a one-sample shift between adjacent codevectors. In [17.39], a 1024-sample circular buffer $v(n), n = 0, 1, \ldots, 1023$ is used to store the 1024 codevectors of a 10-bit fixed codebook with 1-sample shift. The j-th codevector is defined to be $c_k(n) = v(k+n), 0 \le n \le N-1, 0 \le k \le 1023$, where $v(k+1024) = v(k)$ since $v(n)$ is in a circular buffer. Let $\boldsymbol{Hc}_k = \boldsymbol{r}_k = [r_k(0), r_k(1), \ldots, r_k(N-1)]^{\mathrm{T}}$. Then, due to the lower triangular Toeplitz structure of the $\boldsymbol{H}$ matrix, it is shown in [17.39] that $r_{k+1}(n)$ can be computed recursively from $r_k(n)$ as follows.

$$r_{k+1}(n-1) = r_k(n) - v(k)h(n)\,, \quad 1 \le n \le N-1\,,$$

$$r_{k+1}(N-1) = \sum_{n=0}^{N-1} v(k+1+n)h(N-1-n)\,.$$

This recursion is best understood with a graphical interpretation of the matrix-vector multiplication $\boldsymbol{Hc}_k$. In such a multiplication, the column vector $\boldsymbol{c}_k$ is turned sideways by 90° counterclockwise, then each of its elements is sample-by-sample multiplied with each element of the corresponding column of the matrix $\boldsymbol{H}$. Next, all the product terms along each row of the matrix is summed. The resulting column vector is $\boldsymbol{r}_k$. In the recursion above proposed in [17.39], when going from $\boldsymbol{c}_k$ to $\boldsymbol{c}_{k+1}$, the top element of $\boldsymbol{c}_k$ is removed, the remaining $N-1$ element is shifted up by one sample, and a new element is added at the bottom. Due to the lower triangular Toeplitz structure of the matrix $\boldsymbol{H}$, most of the product terms and the partial sums needed for $\boldsymbol{r}_{k+1}$ are already calculated for $\boldsymbol{r}_k$ and thus can be reused. However, the contribution to $r_k(n)$ due to the removed top element of $\boldsymbol{c}_k$ is already added to the element of $\boldsymbol{r}_k$. That is why in the first equation of the recursion above, the product term contribution due to the top element of $\boldsymbol{r}_k$ needs to be subtracted out again. None of the product terms in the second equation of the recursion above involving $r_{k+1}(N-1)$ has been calculated when calculating $\boldsymbol{r}_k$, so the entire summation in this second

equation needs to be calculated. This recursion above costs $(N-1)+N=2N-1$ multiply–add operations for each new $\boldsymbol{r}_{k+1}$ vector.

In fact, there is an even more-efficient recursion that requires only $N-1$ multiply–add operations for each new $\boldsymbol{r}_k$ vector. The trick is to reverse the order of the $\boldsymbol{r}_k$ vectors in the recursion. Using the graphical interpretation of the matrix-vector multiplication $\boldsymbol{Hc}_k$ explained above, one can see that if the recursion now starts at $\boldsymbol{r}_{1023}$ and going backward toward $\boldsymbol{r}_0$, then each time when going from $\boldsymbol{c}_{k+1}$ to $\boldsymbol{c}_k$, the bottom (last) element of $\boldsymbol{c}_{k+1}$ is removed, the remaining elements are shifted down by one sample, and a new element is added at the top. However, for the first $N-1$ element of $\boldsymbol{c}_{k+1}$ that are being shifted down, their partial sum contributions to $\boldsymbol{r}_k$ are exactly the first $N-1$ elements of $\boldsymbol{r}_{k+1}$. After these first $N-1$ elements of $\boldsymbol{c}_{k+1}$ have been shifted down and a new element added at the top to get $\boldsymbol{c}_k$, their partial sum contributions to $\boldsymbol{r}_k$ correspond to the product terms in the second through the N-th columns of the matrix $\boldsymbol{H}$. Thus, only the product terms in the first column need to be added to the partial sums already calculated and stored in the first $N-1$ elements of $\boldsymbol{r}_{k+1}$. Thus, the recursion can be described as

$$r_k(n) = r_{k+1}(n-1) + v(k)h(n)\,, \quad 1 \le n \le N-1\,,$$
$$r_k(0) = v(k)h(0) = v(k)\,.$$

The last equality above holds because $h(0)=1$. Therefore, this recursion performed in the reverse order requires only $N-1$ multiply-add operations for each new $\boldsymbol{r}_k$ vector. As mentioned above, this recursion can be used for the adaptive codebook search for those pitch lags not smaller than the subframe size.

The effect of shifting more than one sample between adjacent codevectors in the fixed codebook has been studied and reported in [17.36]. As the amount of shift between adjacent fixed codebook vectors increases, the coder performance increases, but the codebook search complexity also increases. It was found in [17.36] that for a large codebook (with 128 or more codevectors), a two-sample shift gives the same performance as a fully independent codebook. For this reason, the FS1016 coder uses such a fixed codebook with a two-sample shift between adjacent codevectors.

The same basic idea in the more efficient recursion above can easily be extended to the fixed codebook with a two-sample shift. In this case, each time when going from $\boldsymbol{c}_{k+1}$ to $\boldsymbol{c}_k$, the bottom two elements of $\boldsymbol{c}_{k+1}$ are removed, the remaining elements are shifted down by two samples, and two new elements are added at the top. For the first $N-2$ element of $\boldsymbol{c}_{k+1}$ that are being shifted down, their partial sum contributions to $\boldsymbol{r}_k$ are exactly the first $N-2$ elements of $\boldsymbol{r}_{k+1}$. These partial sum contributions to $\boldsymbol{r}_k$ correspond to the product terms in the third through the N-th columns of the matrix $\boldsymbol{H}$. Thus, only the product terms in the first two columns need to be added to the partial sums already calculated and stored in the first $N-2$ elements of $\boldsymbol{r}_{k+1}$. Thus, this recursion takes only $(N-1)+(N-2)=2N-3$ multiply–add operations to calculate each new $\boldsymbol{r}_k$.

In the case of the FS1016 coder, the elements of the fixed codebook are obtained by using a sequence of zero-mean, unit-variance, white Gaussian random numbers that are center-clipped at 1.2, which results in roughly 75% of the samples being zero. Thus, the fixed codebook is not only shifted, but also 75% *sparse*.

A sparse fixed codebook is said to produce slightly improved perceptual quality of the CELP output speech [17.36,38]. Furthermore, a sparse fixed codebook also reduces the codebook search complexity [17.38]. The reason is quite simple. Consider the graphical interpretation of the matrix–vector multiplication $\boldsymbol{Hc}_k$ again. When 75% to 90% of the elements in $\boldsymbol{c}_k$ are zero, the product terms in the corresponding columns of the matrix $\boldsymbol{H}$ do not need to be calculated, and thus the corresponding calculations can be saved. According to [17.20], the shifted sparse fixed codebook structure reduces the computation by a factor of 20.

In the FS1016 coder, the fixed codebook gain is quantized to 5 bits using a non-uniform scalar quantizer. An interoperable coder may use only a subset of the fixed codebook to reduce the computational complexity. The FS1016 coder also uses unequal forward error correction to protect perceptually most sensitive bits, and it uses extensive parameter smoothers to reduce the quality degrading effects of bit errors. At the time it was standardized, the output speech quality of the FS1016 coder was considered one of the best at around 4.8 kb/s.

17.8 Vector Sum Excited Linear Prediction (VSELP)

Vector sum excited linear prediction (VSELP) [17.17, 40] is another class of CELP coder that uses a specially structured fixed codebook to reduce the codebook search complexity. VSELP is the second type of CELP coder that was standardized.

Each codevector in the fixed codebook of VSELP is constructed as a weighted sum of M independent basis vectors, with the weights taking only two possible values: $+1$ or -1. Thus, an M-bit fixed codebook uses exactly M basis vectors. Let $\boldsymbol{v}_m = [v_m(0), v_m(1), \ldots, v_m(N-1)]^{\mathrm{T}}$, $m = 1, 2, \ldots, M$ be the M basis vectors. Then, the 2^M codevectors in an M-bit VSELP fixed codebook is constructed as

$$\boldsymbol{c}_k = \sum_{m=1}^{M} \theta_{km} \boldsymbol{v}_m \tag{17.11}$$

for $k = 0, 1, \ldots, 2^M$, where $\theta_{km} = +1$ if bit m of the k-th codeword is 1, and $\theta_{km} = -1$ if bit m of the k-th codeword is 0.

An 8 kb/s VSELP coder [17.17] was selected as the TIA interim standard IS-54 for North American digital cellular telephone applications, although this IS-54 standard was later rescinded. A 6.7 kb/s VSELP coder was selected as the Japanese digital cellular PDC full-rate (PDC-FR) standard coder, and a 5.6 kb/s VSELP coder [17.21] also became the GSM half-rate (GSM-HR) standard coder.

In the IS-54 VSELP coder, there are two fixed codebooks sequentially searched, similar to a multistage VQ configuration. In the PDC-FR VSELP coder, only one fixed codebook is used. In the GSM-HR VSELP coder, there are two fixed codebooks, but how many fixed codebooks are used depends on the voicing mode. For example, for a voiced mode the adaptive codebook and one of the fixed codebooks is used, but for an unvoiced mode the adaptive codebook is not used and two fixed codebooks are used.

In the IS-54 VSELP coder, the adaptive codebook and the two fixed codebooks are treated like three successive stages of quantization. The codebook gains are left *floating* when searching for the codebook vectors of these three stages. The mathematics of the efficient codebook search is somewhat involved. Due to the space limitation, only the basic concepts will be described below. Interested readers are referred to [17.17] for more mathematical details.

The adaptive codebook is first searched using the method described in Sect. 17.7. Next, when searching each of the two fixed codebooks, each of the M basis vectors are first filtered through the weighted short-term synthesis filter with zero initial memory to get the ZSR vector of that basis vector. Then, each of such ZSR vectors are orthogonalized with respect to each other and with respect to the ZSR vector due to the selected adaptive codebook vector. The ZSR vector due to each of the 2^M fixed codebook vectors can then be expressed as a linear combination of the M orthogonalized ZSR vectors due to the M basis vectors.

The search procedure needs to find the fixed codebook vector that minimizes a distortion measure that is the ratio of two quantities:

1. the square of the correlation between the target vector and the ZSR vector due to the fixed codebook vector
2. the energy of the ZSR vector due to the fixed codebook vector

These two quantities can each be evaluated in a recursive manner with low computational complexity if the codebook search procedure sequences through the codevectors using a binary Gray code so that the codewords of the adjacent codevectors differ by only one bit.

Another factor of two saving can be obtained by observing that the two fixed codevectors corresponding to the two codewords that are 1s complement of each other have the same shape but only differ in sign. Thus, the two quantities above for the distortion measure will be the same for these two complementary codevectors. The winner of the two complementary codevectors can be determined easily by just examining the sign of the correlation term in the first quantity. This means that only half as many distortion values need to be calculated.

Another novel feature of the IS-54 VSELP coder is the way it quantizes the gains of the adaptive codebook and the fixed codebooks. First the adaptive codebook vector and fixed codebook vectors are identified by the codebook search procedure outlined above. After that, rather than separate scalar quantization of each of the codebook gains as was the normal procedure in previous coders, this VSELP coder first encodes the speech energy of the entire 20 ms frame. Then, the approximate excitation signal energy in each of the 5 ms subframes is calculated based on the frame energy of speech and the reflection coefficients in each subframe. Next, the adaptive codebook gain and the two fixed codebook gains in each subframe are converted to the energy domain expressed as fractions of the total excitation energy in that subframe. The resulting three energy fractions are then jointly vector quantized using a trained codebook. At the VSELP decoder, the three decoded energy fractions are converted back to the codebook gain domain.

It is said [17.17] that such a gain VQ scheme in the energy-fraction domain not only makes the resulting energy fractions more correlated which allows more efficient VQ coding, but also the fact that all codebook gains are represented as energy fractions makes the en-

ergy of the decoded speech less susceptible to bit errors in the codebook gain codeword.

The GSM-HR VSELP coder also has another feature not yet discussed in this chapter – a long-term predictor with a fractional (non-integer) pitch period [17.41, 42]. Rather than using an integer pitch period as in previous predictive coders, an interpolation filter is used to achieve the effect of a non-integer pitch period. This allows the long-term synthesis filter to model the pitch periodicity of voiced speech with a higher temporal resolution, thus resulting in higher output speech quality.

In VSELP, the basis vectors for the fixed codebook can be optimized using a training database. It is reported [17.17] that such optimization of the basis vectors resulted in 0.64 dB improvement in weighted segmental SNR and significant improvement in subjective quality.

17.9 Low-Delay CELP (LD-CELP)

The low-delay CELP (LD-CELP) coder [17.5, 6, 43] is another different kind of CELP coder. It is the third kind of CELP coder that was selected as a standard coder. The ITU-T recommendation G.728 standard coder [17.6] is based on it. There are several important differences between G.728 and the previous CELP coders. These differences will be described in this section.

First of all, the excitation vector dimension used in G.728 LD-CELP is much shorter than that of earlier CELP coders. Most of the earlier CELP coders used an excitation vector dimension of 40 samples (5 ms) or more in order to get a better excitation VQ efficiency (since VQ performance generally increases with increasing vector dimension). However, using such a large vector dimension will not meet the ITU-T requirement of a very low coding delay. In order to achieve a one-way coding delay of less than 2 ms (16 samples at 8 kHz sampling), the G.728 coder is forced to use a frame size and vector dimension of merely five samples (0.625 ms).

With a frame size of five samples, basically the G.728 encoder is required to produce 10 bits as output for every five samples of input speech. This strict constraint, which is due to the low-delay requirement, greatly limits the flexibility of the coder design. While previous CELP coders could spend 20–40 bits per frame to encode the short-term predictor parameters and another 10 to 12 bits per subframe to encode the long-term predictor parameters, the G.728 coder cannot do any of these since it only has 10 bits total to encode everything in a frame.

This leads to the second major difference between G.728 and previous CELP coders – the predictor parameters are made *backward-adaptive*. Previous CELP coders all used *forward-adaptive* predictors, where the optimal predictor parameters are derived from the input speech and then quantized and transmitted to the decoder. With backward adaptation, no bits are spent in sending the parameters to the decoder; instead, the parameters are locally derived at both the encoder and the decoder from previously decoded speech signal or parameters. See [17.13] for a more-detailed discussion of backward adaptation versus forward adaptation. For G.728, both the short-term synthesis filter coefficients and the excitation gain are backward-adaptive and thus does not require any bit to transmit them to the decoder.

The third major difference between G.728 and previous CELP coders is the elimination of the long-term predictor, and in its place, the use of a high-order backward-adaptive short-term predictor. The reason is that a backward-adaptive long-term predictor has a strong tendency to diverge at the decoder due to bit errors or other reasons that cause a mismatch of the encoder states and the decoder states. On the other hand, when the backward-adaptive short-term predictor coefficients are obtained by performing LPC analysis on previously decoded speech signal, it is relatively easy to maintain the convergence at the decoder even with bit errors and states mismatch. It was found [17.5] that, with the short-term predictor order set at the conventional value of 10, the lack of a long-term predictor degrades female speech quality significantly. On the other hand, if the short-term predictor order is allowed to increase to 50, then a 50-th-order short-term predictor can exploit the pitch periodicity in female speech, since most female voices have a pitch period of less than 50 samples.

Using a 50-th-order short-term predictor in a conventional forward-adaptive CELP coder would be impractical due to the large number of predictor coefficients that need to be transmitted. However, since G.728 uses backward adaptation to update the short-term predictor, it doesn't cost any bit to increase the predictor order to 50. The only price paid is the increased computational complexity to derive the 50 predictor coefficients. In order to achieve toll quality

at 16 kb/s and with a frame size of only five samples, it was necessary to exploit the pitch periodicity in speech by using a 50-th-order short-term predictor. Thus, under the severe low-delay constraint, the G.728 had to make the complexity-quality trade-off – achieving toll quality through the use of a higher-complexity 50-th-order backward-adaptive short-term predictor.

The encoder structure of the G.728 LD-CELP coder is equivalent to the structure shown in Fig. 17.3 but without the long-term synthesis filter and the signal modifier. Both the excitation gain and the short-term synthesis filter are backward-adaptive. The excitation shape codebook (the fixed codebook) is a 10-bit codebook for five-dimensional VQ. However, to control the computational complexity of the codebook search, a gain-shape structured codebook is used, with the codebook constructed as the product of a 3-bit gain codebook and a 7-bit shape codebook. The three gain bits consists of one sign bit and two magnitude bits. The 7-bit shape codebook contains 128 independent codevectors.

Both the 7-bit shape codebook and the 2-bit magnitude codebook are closed-loop-trained based on the weighted distortion measure of LD-CELP using a large training speech file. Thus, the effects of backward adaptation of the predictor and gain are automatically taken into account in the training. Such closed-loop codebook training gives significant audio quality improvement and is crucial for achieving good speech quality in LD-CELP.

17.10 Pitch Synchronous Innovation CELP (PSI-CELP)

Pitch synchronous innovation CELP (PSI-CELP) [17.44] is the fourth kind of CELP coder that was selected as a standard coder. For the personal digital cellular (PDC) system of Japan, the half-rate standard coder (PDC-HR) is a 3.45 kb/s PSI-CELP coder [17.22].

The main distinguishing feature of PSI-CELP is that the fixed codebook is made adaptive by repeating the first portion of each codevector in a pitch-synchronous manner if the pitch period is smaller than the excitation vector dimension. Specifically, if the pitch period T is less than the vector dimension N, then the first T samples of the fixed codebook vectors are periodically repeated for later samples starting at the $(T+1)$-th sample. This is important for the PDC-HR coder because the coder uses a fairly large subframe size (vector dimension) of 80 samples (10 ms), which is larger than many typical pitch period values.

In place of the typical adaptive codebook in other CELP coders, the PDC-HR standard PSI-CELP coder actually uses either an adaptive codebook or a fixed codebook, with the fixed codebook mainly used in non-periodic regions of the speech signal. The adaptive codebook uses fractional pitch period [17.41, 42]. To save the codebook storage requirement, the fixed codebook has four basis vectors each rotated eight times to get 32 codevectors.

In place of the typical fixed codebook in other CELP coders, the PDC-HR standard PSI-CELP coder uses two stochastic codebooks in a conjugate structure [17.45] (which is similar to two-stage VQ). Each of the two stochastic codebooks contains 16 vectors. The selected codevector from each codebook is multiplied by a sign and the resulting two codevectors are added together. The pitch synchronous innovation (PSI) procedure is applied to the stochastic codebooks. To reduce the codebook search complexity, six out of the 16 codevectors from each stochastic codebook are preselected using a simplified method. Only the preselected codevectors go through full-complexity search.

The adaptive/fixed codebook gain and the stochastic codebook gain are jointly vector quantized to seven bits per subframe in a way somewhat similar to the codebook gain quantization of VSELP [17.17]. A delayed-decision coding technique is used to improve the PSI-CELP performance. Two best candidate codevectors are selected from the adaptive/fixed codebook search, and the remaining quantization procedures are performed for each candidate. The candidate that gives the lower distortion in the fully quantized version is selected as the winner.

17.11 Algebraic CELP (ACELP)

Algebraic code-excited linear prediction (ACELP) signifies the use of algebraic codes to make up the excitation in CELP. Due to the popularity of ITU-T recommendation G.729 [17.46] many people think of ACELP [17.24]

and G.729 as synonymous. However, in reality it is only the excitation of G.729 that is algebraic. Furthermore, G.729 and other speech coding standards utilizes sparse algebraic codes, (SACs) [17.24], and not an algebraic code as originally proposed in [17.23] in 1987 for CELP. The sparseness contributes significantly to a low computational complexity. Of further interest, a number of the techniques and structural changes proposed in [17.23] to the original CELP structure [17.3, 4] form the cornerstones of G.729 and many other speech coding standards. It should be noted that partly overlapping techniques were proposed in [17.38] in 1986. These techniques and structures are reviewed in other relevant sections, e.g., Sect. 17.6.

As indicated above, although the abbreviation ACELP was introduced in 1990 in [17.24], earlier work on algebraic codes for CELP appeared in 1987 in [17.23] and [17.47]. A significant advancement is the notion of sparse algebraic codes [17.24]. The idea of using sparse excitation in CELP coding was introduced in [17.38] in 1986, but in the context of Gaussian excitation vectors and not algebraic codes. Hence, the pulse amplitudes would be Gaussian distributed and not binary $(+1, -1)$ as in sparse algebraic codes. In [17.38], the use of sparse excitation for CELP is compared to MPLPC (multipulse linear predictive coding) [17.2] in terms of the number of nonzero pulses for voiced speech. Other early publications on sparse pulse excitation for CELP include [17.48] and [17.49].

ACELP has been a dominating form of excitation as the fixed codebook excitation in CELP speech coding standards from the mid 1990s until today (2006).

17.11.1 ACELP Background

ACELP appears to originate from the idea of using binary error correcting codes to represent N points on an M dimensional hyper sphere, i. e., a fixed codebook of size N and vector dimension M [17.47]. Even earlier, [17.50] proposed a spherical vector quantizer for encoding of the 1000 Hz base band of the short-term prediction residual after eighth-order LPC. The justification for the view of the fixed codebook representing points on a hyper sphere originates from the separate gain of the fixed codebook. Basically, the length of the residual vector, that is being approximated by the fixed codebook, can be considered normalized. Hence, the residual vectors will be points on a hypersphere. The task of the fixed codebook is to populate the hyper sphere in an optimal way. If the common assumption, that the samples of the residual vector after short- and long-term prediction are independent and identically distributed (i.i.d.) Gaussian, is reasonable, then the normalized residual vectors in terms of points on the hypersphere will be a uniform distribution. All $(+1, -1)M$-tuples will represent a uniform discrete sampling of the hypersphere with 2^M points, and it seems reasonable to pick the fixed codebook as a subset of the $2^M(+1, -1)M$-tuples. Error-correcting codes are attractive for picking the subset as they basically maximize the minimum distance between any two codewords, codevectors in the context of a fixed codebook, and spread out codewords for maximum coverage. Naturally, a mapping from the $(0, 1)M$ bits of a error correcting codeword to the $(+1, -1)M$ elements of the codevector of the fixed codebook is required. The benefit of such a fixed codebook is that no storage is required and it lends itself well to efficient methods.

An important additional feature is the realization that only relatively few nonzero pulses are required. For analysis-by-synthesis multipulse with only short-term prediction, the need for few nonzero pulses was reported in [17.2], where little improvement was observed after 8 pulses per 80 samples (10 ms). Similarly in the context of CELP, i. e., with long-term prediction as well, [17.38] reported the need for few nonzero elements. Although [17.38] includes long-term prediction an equivalent of four pulses per 40 samples (5 ms) was reported. However, neither [17.2] nor [17.38] were considering pulses of binary amplitude, but instead pulses of arbitrary amplitude. In [17.48,51,52], and [17.49] sparse binary pulse codevectors were discussed, and in [17.24] it was proposed along with the definition of the abbreviation ACELP and SAC. Technically, sparse $(+1, -1)$ binary excitation is really $(+1, 0, -1)$ ternary excitation.

The concept of pulse tracks and interleaved permutation codes [17.53, 54] contribute to further reducing complexity of searching and bit-rate of coding the sparse binary pulses. Partitioning the sample of an excitation vector into multiple tracks and applying a permutation code (resulting in sparse binary pulses) to each track has proven effective. Furthermore, interleaving the tracks is typical as it allows the flexibility of high local density of pulses in the final fixed codebook excitation.

Table 17.1 Example ACELP ISPP excitation structure

Track/code	Sample positions	Bits
1	0, 5, 10, 15, 20, 25, 30, 35	3 + 1 = 4
2	1, 6, 11, 16, 21, 26, 31, 36	3 + 1 = 4
3	2, 7, 12, 17, 22, 27, 32, 37	3 + 1 = 4
4	3, 8, 13, 18, 23, 28, 33, 38	3 + 1 = 4
5	4, 9, 14, 19, 24, 29, 34, 39	3 + 1 = 4

Table 17.1 shows an example of a 40 sample ACELP fixed codebook made up of five single pulse interleaved permutation codes (5 tracks). Hence, it is a so-called interleaved single pulse permutation (ISPP) design. It results in a $5 \cdot (3+1) = 20$-bit fixed codebook excitation with five pulses. Other ACELP interleaved permutation codes employ multiple pulses per code/track [17.55].

Key advantages of ACELP are that it

- lends itself well to efficient methods,
- eliminates the need to store codebook,
- offers good speech quality,
- is flexible, and
- enables the use of large codebooks.

17.11.2 ACELP Efficient Search Methods

Modern ACELP is typically used in conjunction with the adaptive codebook implementation of the long-term (pitch) synthesis filter. Hence, the present section will present the search methods of ACELP in that context. Furthermore, the perceptual weighting filter of modern ACELP coders is frequently given by

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} , \tag{17.12}$$

where $A(z)$ is the unquantized prediction error filter and $0 < \gamma_2 < \gamma_1 < 1$. This form is used here without limitation, and all equations are easily extended to accommodate any perceptual weighting filter.

The efficient search methods for ACELP are based on a target signal, $t(n)$, which is basically the weighted input speech from which contributions from

1. ringing of filter memory
2. adaptive codebook

have been subtracted. Hence, the target signal, $t(n)$, is in the weighted speech domain. Searching the ACELP fixed codebook involves finding the entry that minimizes the MSE between the target signal and the ACELP codevector passed through the perceptual weighting filter and short-term synthesis filter, both with zero memory. This is illustrated in Fig. 17.8, and it is expressed as

$$I = \arg\min_k (E_k) = \arg\min_k \left\{ \sum_{n=0}^{N-1} [t(n) - y_k(n)]^2 \right\} , \tag{17.13}$$

where $y_k(n)$ is the output from passing the k-th ACELP codevector through the perceptual weighting filter and short-term synthesis filter. In terms of the ACELP codevector $c_k(n)$ (17.13) is expressed as

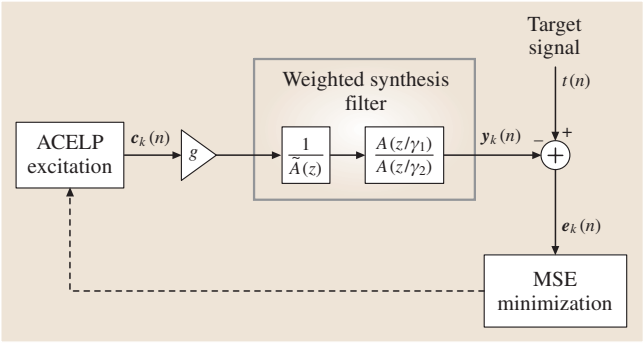

Fig. 17.8 Basic ACELP fixed codebook excitation search

$$I = \arg\min_k \left[\sum_{n=0}^{N-1} (t(n) - \{h(n) * [g \cdot c_k(n)]\})^2 \right] , \tag{17.14}$$

where $h(n)$ is the impulse response of the weighted synthesis filter, i. e., the inverse z-transform of

$$H(z) = \frac{1}{\tilde{A}(z)} W(z) = \frac{1}{\tilde{A}(z)} \frac{A(z/\gamma_1)}{A(z/\gamma_2)} , \tag{17.15}$$

where $1/\tilde{A}(z)$ is the quantized short-term synthesis filter. Note that enhancements to the ACELP codebook can be included into the impulse response, $h(n)$, without affecting the following search techniques. The enhancements are in the form of a prefilter, $F(z)$. One such example is the in-frame pitch enhancement for pitch lags shorter than the length of the ACELP codevector [17.25, 56]:

$$F(z) = \frac{1}{1 - \beta Z^{-T}} , \tag{17.16}$$

where T is the pitch lag, and β is a suitable, typically adaptive, filter coefficient related to the periodicity. This particular ACELP pitch prefilter is incorporated into $h(n)$ according to

$$\begin{aligned} h(n) &\leftarrow h(n) + \beta h(n-T) , \\ n &= T, T+1, \ldots , N-1 . \end{aligned} \tag{17.17}$$

In AMR-WB (adaptive multi-rate) [17.57], the ACELP prefilter additionally includes a tilt part:

$$F(z) = \frac{1}{1 - \beta Z^{-T}} (1 - \alpha z^{-1}) , \tag{17.18}$$

where also α is adaptive.

Equation (17.14) is written in vector form as

$$
\begin{aligned}
I &= \arg\min_k \left[(\boldsymbol{t} - g\boldsymbol{H}\boldsymbol{c}_k)^{\mathrm{T}} (\boldsymbol{t} - g\boldsymbol{H}\boldsymbol{c}_k) \right] \\
&= \arg\min_k \left[\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t} + g^2 (\boldsymbol{H}\boldsymbol{c}_k)^T (\boldsymbol{H}\boldsymbol{c}_k) \right. \\
&\quad \left. - g\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k - g (\boldsymbol{H}\boldsymbol{c}_k)^{\mathrm{T}} \boldsymbol{t} \right] \\
&= \arg\min_k \left(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t} + g^2 \boldsymbol{c}_k^{\mathrm{T}} \boldsymbol{H}^{\mathrm{T}} \boldsymbol{H}\boldsymbol{c}_k - 2g\boldsymbol{t}^{\mathrm{T}} \boldsymbol{H}\boldsymbol{c}_k \right) ,
\end{aligned}
\tag{17.19}
$$

where

$$
\begin{aligned}
\boldsymbol{t} &= [t(0)t(1)\cdots t(N-1)]^{\mathrm{T}} , \\
\boldsymbol{c}_k &= [c_k(0)c_k(1)\cdots c_k(N-1)]^{\mathrm{T}} , \\
\boldsymbol{H} &= \begin{pmatrix}
h(0) & 0 & \cdots & 0 \\
h(1) & h(0) & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
h(N-2) & h(N-3) & \cdots & 0 \\
h(N-1) & h(N-2) & \cdots & h(0)
\end{pmatrix} .
\end{aligned}
$$

According to common procedure in ACELP, the excitation, $\boldsymbol{c}_k$, is found under the assumption of optimal gain g.

Table 17.2 Key properties of ACELP excitation in standards

Standard	Rate (kbps)	ACELP frame size (samples)	Pulses	Tracks	Bits	
G.723.1	5.3	60	4	4	17	
G.729		40	4	4	17	
G.722.2/AMR-WB	23.85 / 23.05	64	24	4	88	
	19.85		18	4	72	
	18.25		16	4	64	
	15.85		12	4	52	
	14.25		10	4	44	
	12.65		8	4	36	
	8.85		4	4	20	
	6.6		2	2	12	
GSM AMR	12.2 (GSM EFR)	40	10	5	35	
	10.2		8	4	31	
	7.95 / 7.4 (TIA EFR)		4	4	17	
	6.7 (PDC EFR)		3	3	14	
	5.9		2	2	11	
	5.15 / 4.75		2	5	9	
EVRC	Full-rate	53 / 54	8	5	35	
	Half-rate		3	3	10	
VMR-WB	Full-rate	64	8	4	36	
	Voiced and generic half-rate		2	2	12	
MPEG-4	8 kHz core		3-12			
	8 kHz enhancement	40	2			
SMV	Rate 1/1 type 1	40	8	8	30	
	Rate 1/1 type 0	40	5	5	21	
			5	5	20	⇒ 22
			5	4	20	
	Rate 1/2 type 1	53/54	2	2	12	
			3	3	11	⇒ 13
			5	5	11	
	Rate 1/2 type 0	80	2	2	14	
			3	3	13	⇒ 15
			Gaussian	NA	13	

The gain is determined by minimizing the error energy:

$$\frac{\partial E_k}{\partial g} = \frac{\partial}{\partial g}\left(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t} + g^2\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k - 2g\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k\right) = 0$$

$$\Downarrow$$

$$g = \frac{\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k}{\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k} \,. \tag{17.20}$$

Inserting this result in (17.19) yields

$$I = \arg\min_k \left(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t} - \frac{\left(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k\right)^2}{\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k}\right) . \tag{17.21}$$

Since $\boldsymbol{t}$ is independent of the codevector, $\boldsymbol{c}_k$, this is equivalent to

$$I = \arg\max_k \left(\frac{\left(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k\right)^2}{\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k}\right) . \tag{17.22}$$

The numerator is often rewritten in terms of the backward filtered target vector [17.23,53], $\boldsymbol{t}_{\mathrm{b}} = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{t}$, and the search is expressed as

$$I = \arg\max_k \left(\frac{\left(\boldsymbol{t}_{\mathrm{b}}^{\mathrm{T}}\boldsymbol{c}_k\right)^2}{\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k}\right) = \arg\max_k \left(\frac{\left(\boldsymbol{t}_{\mathrm{b}}^{\mathrm{T}}\boldsymbol{c}_k\right)^2}{\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{c}_k}\right) , \tag{17.23}$$

where $\boldsymbol{\Phi} = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}$ is symmetric and contains the autocorrelation of the impulse response of $H(z)$:

$$\Phi(i,\,j) = \begin{cases} \sum_{l=0}^{N-1-i} h(l)h(l+i-j) \,, & j \le i \\ \Phi(j,\,i) \,, & j > i \end{cases} . \tag{17.24}$$

The backward filtered target $\boldsymbol{t}_{\mathrm{b}}$ and the autocorrelation matrix $\boldsymbol{\Phi}$ are calculated prior to searching the ACELP codebook.

The efficient search methods all explore ways to most efficiently evaluate (17.23), either in a mathematically equivalent way, or in a slightly suboptimal way with only minor sacrifice in performance. The following subsections will present some of the methods typically used in ACELP.

Sparse ACELP

In sparse ACELP only relatively few pulses are nonzero, and if they are binary, the nonzero pulses take on an amplitude of ± 1. The P pulses, $P < N$, are described uniquely by the location and amplitude, m_i and a_i, respectively, $i = 0, 1, \ldots, P-1$. Note that $a_i = \pm 1$. With this notation, the search of (17.23) can be written as [17.53]

$$I = \arg\max_{\boldsymbol{m}\times\boldsymbol{a}} \left(\frac{\left(\sum_{i=0}^{P-1} t_{\mathrm{b}}(m_i)a_i\right)^2}{\sum_{i=0}^{P-1}\Phi(m_i,m_i) + 2\sum_{i=0}^{P-2}\sum_{j=i+1}^{P-1} a_i a_j \Phi(m_i,m_j)}\right) , \tag{17.25}$$

where the search is identifying the optimal P pulse positions $\boldsymbol{m} = (m_0, m_1, \ldots, m_{P-1})$ and P amplitudes $\boldsymbol{a} = (a_0, a_1, \ldots, a_{P-1})$. This search can be performed in a nested loop so that the addition of one more pulse is considered in each loop. At the inner most loop, the final cost function for a given candidate is calculated by one additional addition and the square to get the numerator, and P additional additions and one multiplication to get the numerator. This provides for a very efficient exhaustive search, but as the size of the codebook increases this quickly becomes impractical [17.53]. Sparse ACELP is used in all ACELP standards.

A Priori Sign

Studying the numerator of (17.25) it is evident that the cross correlation will be maximized by setting the pulse amplitudes, $\boldsymbol{a}$, basically the pulse signs, to be the same as the sign of the backward filtered target, $\boldsymbol{t}_{\mathrm{b}}$ [17.58]. Accordingly, by setting the pulse signs a priori only the pulse locations $\boldsymbol{m}$ need to searched. This is achieved by incorporating the a priori pulse signs into the backward filtering target and correlation matrix prior to the search [17.58] according to:

$$\tilde{t}_{\mathrm{b}}(n) = |t_{\mathrm{b}}(n)| \,, \tag{17.26}$$

$$\tilde{\Phi}(i,\,j) = \mathrm{sign}[t_{\mathrm{b}}(i)]\mathrm{sign}[t_{\mathrm{b}}(j)]\Phi(i,\,j) \,. \tag{17.27}$$

and performing the search for pulse positions according to

$$I = \arg\max_{\boldsymbol{m}} \left(\frac{\left(\sum_{i=0}^{P-1} \tilde{t}_{\mathrm{b}}(m_i)\right)^2}{\sum_{i=0}^{P-1}\tilde{\Phi}(m_i,m_i) + 2\sum_{i=0}^{P-2}\sum_{j=i+1}^{P-1}\tilde{\Phi}(m_i,m_j)}\right) , \tag{17.28}$$

where, again, the search identifies the optimal P pulse positions $\boldsymbol{m} = (m_0, m_1, \ldots, m_{P-1})$. To eliminate the multiplication by 2 during the search, all off-diagonal elements of $\tilde{\Phi}(i,\,j)$ can be multiplied by 2 prior to the search. The a priori determination of the pulse signs is

Table 17.3 Effectiveness of ACELP focused search [17.53]

Percentage of exhaustive search	SNR (dB)
100	22.20
4	22.14
1.6	22.00
0.2	22.05
0.15	21.83
0.05	21.80
0.03	21.50

not in general equivalent to the exhaustive search given by (17.25). It is used in G.729 [17.46], G.723.1 [17.25], EVRC [17.26], and GSM-EFR [17.56] among others. Note that the presetting of the sign can be based on the sign of a signal that is a linear combination of the backward filtered target and the long-term prediction residual [17.56] (or a similarly suitable signal), and hence not only based on the sign of the backward-filtered target.

Focused Search

Various focused search methods exist [17.53, 54, 58]. Generally, the P pulses are searched in nested loops and constraints on the correlation,

$$C = \sum_{i=0}^{p-2} \tilde{t}_b(m_i) , \quad p < P , \tag{17.29}$$

are set up to disregard subsections of the ACELP codebook (represented by entry into the p-th loop) if the correlation C does not meet a predefined threshold. A table from [17.53] is reproduced in Table 17.3, illustrating this in the form of the relation between the percentage of exhaustive search and the SNR for a sentence spoken by a female talker. The threshold can be calculated in various ways. It can be based on the maximum possible correlation of the existing $p-1$ pulse candidates, or the average correlation over their possible positions, or a combination thereof as in [17.58]. Generalizing [17.58] leads to a threshold based on maximum pulse correlation

$$C_{\max} = \sum_{l=0}^{p-2} \max_{i \in \mathcal{T}_l} \left[\tilde{t}_b(i)\right] \tag{17.30}$$

and average pulse correlation

$$C_{\text{avg}} = \sum_{l=0}^{p-2} \text{avg}_{i \in \mathcal{T}_l} \left[\tilde{t}_b(i)\right] . \tag{17.31}$$

In (17.30) and (17.31), $\mathcal{T}_l$ is the set of possible pulse positions of pulse l. The threshold can then be constructed as a combination of the maximum and average correlation:

$$C_{\text{thr}} = C_{\text{avg}} + \alpha_{\text{thr}}(C_{\max} - C_{\text{avg}}) . \tag{17.32}$$

This will reduce the average complexity. To have a firm limit on the worst-case complexity a threshold on the number of times a certain loop can be entered may be enforced [17.58]. A very large part of the ACELP codebook can be ruled out this way.

Depth-First Tree Search Procedure

As the number of pulses P increases, additional methods can be used to search the effectively increasing codebook efficiently. One such method is used in G.729 Annex A [17.59], GSM-EFR [17.56], and AMR-WB [17.57]. The depth-first tree search procedure would typically be used instead of the focused search to further reduce complexity. The set of P pulses is divided into N_m subsets, typically with an equal number of M pulses in each. While the subsets are searched sequentially, the pulses in each subset are searched jointly. First subset 1, then subset 2, etc. Subsequent sets are searched given the pulses of previous sets. This search is performed iteratively, circulating the assignment of pulses to tracks. For instance, at the first iteration the first two pulses can be assigned to tracks 0 and 1, while for the second iteration the first two pulses can be assigned to tracks 2 and 3, etc. In the second iteration the last two pulses can be assigned to tracks 0 and 1.

A reference signal may be constructed to assist in selecting a subset of pulse positions to consider in a given track. As an example, with pulses 0 and 1 being assigned to tracks 2 and 3 (each of eight positions), based on the reference signal, only four positions in track 2 are preselected for consideration for pulse 0. Such preselection reduces complexity. Techniques like this can be used to control the complexity and balance complexity of multiple rates in a speech coder. The AMR-WB speech coder [17.57] provides a good example of this technique. The reference signal is typically identical to the signal used to set the signs a priori (Sect. 17.11.2). It can be the backward filtered target signal as in [17.60], a linear combination of the backward filtered target and the long-term prediction residual signals as in [17.57], or some other suitable signal.

In G.729A, the depth-first tree search is used instead of the focused search of G.729 and provides a direct comparison. It results in a very significant saving of 5 MIPS (million instructions per second) at the cost of

a slight degradation in performance, about 0.2 dB SNR according to [17.60].

17.11.3 ACELP in Standards

A number of speech coding standards utilizes the ACELP excitation, that is, the sparse binary pulse excitation. In grouping by standards body: ITU-T G.723.1 [17.25] (ACELP for the 5.3 kbit/s rate, multipulse for the 6.3 kbit/s rate), ITU-T G.729 [17.46], ITU-T G.722.2 [17.61], ETSI GSM-EFR (European Telecommunications Standards Institute) [17.56], ETSI/3GPP GSM AMR (3-rd Generation Partnership Project) [17.62], 3GPP GSM AMR-WB [17.57], TIA-127 EVRC (enhanced variable rate codec) [17.26], TIA-136 EFR [17.63], TIA/3GPP2 SMV [17.64], 3GPP2 VMR-WB [17.28], ARIB PDC-EFR (Association of Radio Industries and Businesses), and MPEG-4 CELP (Moving Pictures Expert Group) narrow-band audio [17.65].

It should be noted that ITU-T G.722.2 and 3GPP GSM AMR WB is the same wide-band multirate speech coder. It was initially standardized for cellular by 3GPP and subsequently submitted to the ITU for consideration in an, at the time, ongoing standardization effort of wide-band coding of speech at around 16 kbit/s. It should also be noted that the AMR-WB rate of 12.65 kbit/s is interoperable with one of the rates of 3GPP2 VMR-WB. Furthermore, the GSM EFR is identical to the 12.2 kbit/s rate of ETSI/3GPP AMR, TIA-136 EFR is equivalent to the 7.4 kbit/s rate of AMR, and ARIB PDC-EFR is equivalent to the 6.7 kbit/s rate of AMR. Note that it can be somewhat confusing that multiple standards bodies use the same acronym, EFR. The MPEG-4 CELP is really somewhat of a hybrid between ACELP and multipulse as it doesn't use binary or ternary pulses, but instead apply a VQ of the pulse amplitudes. Similarly, TIA/3GPP2 SMV is a hybrid as it is based on sub-codebooks where most sub-codebooks are ACELP-like, but one is different, and the codebooks are weighted differently in selecting the overall best excitation.

Table 17.2 summarizes some of the key properties of the ACELP fixed codebook excitation structures used in the standards listed above.

17.12 Conjugate Structure CELP (CS-CELP) and CS-ACELP

Conjugate structure CELP (CS-CELP) introduces conjugate structure VQ for CELP. It was originally proposed in [17.66] for CELP. However, earlier the conjugate structure was used in other coders, e.g., in transform coding [17.67]. Generally, conjugate structure VQ constructs the output codevector $\boldsymbol{c}_{i,j}$ as a linear combination of the output codevectors from two codebooks $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ respectively:

$$\boldsymbol{c}_{i,j} = \alpha_1 \cdot \boldsymbol{c}_i + \alpha_2 \cdot \boldsymbol{c}_j \,. \tag{17.33}$$

The advantages of the conjugate structure compared to a single VQ is threefold [17.66]:

- improves resilience to bit-errors,
- reduces memory requirement,
- facilitates reduced complexity methods.

These advantages are demonstrated in [17.66] and [17.68], which also present methods to train the conjugate structure codebooks. Furthermore, [17.66] and [17.68] present an 8 kb/s CS-CELP speech coder where the conjugate structure is used for both the fixed codebook excitation and the joint quantization of the subframe based adaptive codebook gain and fixed codebook gain. This coder was submitted by NTT for the ITU-T (CCITT at the time) G.729 standardization. Although the coder was not standardized as G.729, techniques from multiple candidates were merged and eventually formed G.729. The resulting G.729 standard [17.46] uses the conjugate structure for the 2-dimensional joint VQ of the subframe based adaptive codebook gain and fixed codebook gain. In G.729 the linear combination of the codevectors from the two codebooks is a simple summation, and the conjugate structure codevector is

$$\boldsymbol{c}_{i,j} = \boldsymbol{c}_i + \boldsymbol{c}_j \,. \tag{17.34}$$

One codebook has a bias towards the element corresponding to the fixed codebook gain, while the other codebook has a bias towards the element corresponding to the adaptive codebook gain. This allows open-loop preselection of both codebooks based on the respective dominant parameter. This leaves only a subset (a quarter) for the more-complex joint closed-loop search without any noticeable degradation compared to an exhaustive closed-loop search [17.58].

17.13 Relaxed CELP (RCELP) – Generalized Analysis by Synthesis

Relaxed CELP (RCELP) [17.69] has become synonymous with a specific and practical usage of the generalized analysis-by-synthesis principle proposed in [17.15,70]. The basic idea of generalized analysis-by-synthesis is to relax the waveform matching constraint of CELP without affecting the speech quality. In principle, a relaxation of the waveform matching can be incorporated into the error criterion. However, generalized analysis-by-synthesis proposes a general signal modification function, that is applied to the original signal, constrained to provide a perceptually similar signal. The idea is to modify the signal into a signal that is simpler to represent (in the sense of minimizing distortion relative to the modified signal), yet perceptually indistinguishable from the original. The fundamental and the generalized analysis-by-synthesis paradigms as presented in [17.15] are depicted in Figs. 17.9 and 17.10, respectively.

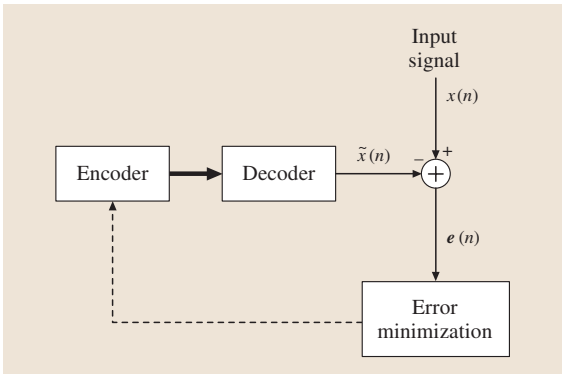

Fig. 17.9 Basic analysis by synthesis

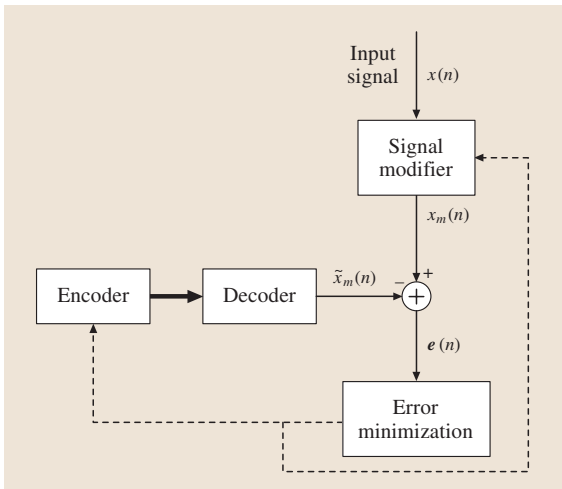

Fig. 17.10 Generalized analysis by synthesis

17.13.1 Generalized Analysis by Synthesis Applied to the Pitch Parameters

Application of generalized analysis-by-synthesis to the pitch parameters implies that one or both of the pitch period contour and the pitch gain contour of the speech signal is modified for easier encoding. A detailed discussion of the application to the pitch period and gain is available in [17.70]. Typically, the pitch period and pitch gain are updated approximately every 5 ms. The idea of RCELP is to update and encode them far less frequently, e.g., every 20 ms, and then use an interpolated pitch period contour and pitch gain contour throughout the 20 ms. This is justified by the observation that the pitch period and periodicity of voiced speech evolve slowly. In order to maintain the coding efficiency when using the interpolated pitch period and gain (compared to the more-frequent update), the speech signal must be modified to follow these contours. Otherwise, the waveform matching of the analysis-by-synthesis principle will break down as the interpolated pitch period and gain will result in a pitch contribution from the adaptive codebook (or equivalently, the pitch predictor) that is misaligned with the reference signal for the analysis-by-synthesis. Since the pitch evolution in voiced speech is slow, typically only minor adjustments to the speech signal are necessary and without impact to the speech quality. This was demonstrated in [17.69] where subjective results showed that the modified speech received mean opinion scores (MOS) very close to those of the original speech, at a level similar to those of 64 kbit/s μ-law (Table 17.4).

Besides exploiting the typical slow evolution of the pitch period to reduce the bits required to encode the pitch period, RCELP can also save the bits otherwise often used to specify fractional pitch lags. Basically, the speech can be modified to fit an interpolated pitch period contour specified by integer lags. Note that the interpolated pitch period contour will have fractional

Table 17.4 MOS of RCELP signal modification [17.69]

	MOS–IRS input	MOS–flat input
64 kb/s μ-law	3.94	4.03
modified speech	3.90	3.99

lags, but at the points of transmission the pitch lag has integer values.

Application to the Pitch Period

Although [17.70] discusses the application of generalized analysis-by-synthesis to both the pitch period and the pitch gain, in speech coding standards, the application to the pitch period has found the widest usage. Accordingly, the application of generalized analysis-by-synthesis to the pitch period is commonly referred as RCELP. In narrow-band speech coding this can bring the bit allocation for the pitch period from $8+5+8+5$ bits to seven bits for a 20 ms frame. This is a significant saving that frees up bits for improving quantization of other parameters of the coder. In practice the pitch period may be estimated open loop and used to create an interpolated continuous pitch track. The speech signal is subjected to minor adjustments in order to fit this interpolated pitch track. Often the adjustments are carried out through time warping and conveniently carried out in the short-term prediction residual domain [17.15]. The modified residual signal can be passed through the short-term synthesis filter or the weighted synthesis filter in order to obtain the modified speech signal or the modified weighted speech signal, respectively. For lower complexity, time shifting of sequential blocks can be applied instead of time warping [17.69]. Later, [17.71] proposed to carry out the signal modification in the weighted speech domain and use a combination of time warping and time shifting of blocks for modifying the speech signal. The method only shifts the high-energy blocks (supposedly containing the pitch pulse and immediate vicinity), while the low-energy blocks are time warped to adjust the pitch period. Effectively, this method does not alter the local waveform properties of the pitch pulses.

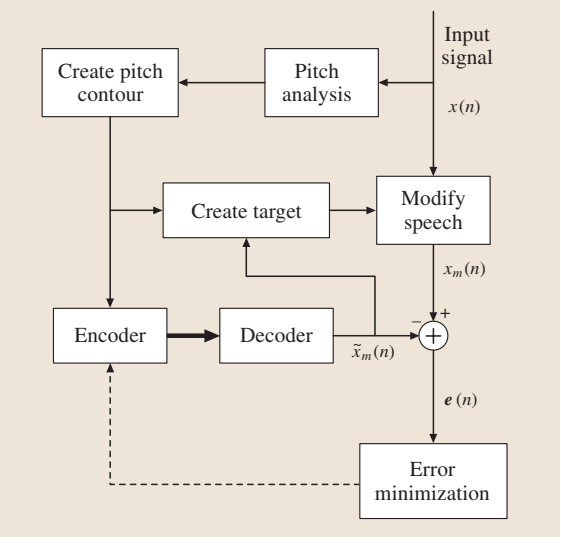

Fig. 17.11 Principle of RCELP

The general sequence of operations for RCELP are:

1. estimate open loop pitch,
2. create pitch contour,
3. create target signal for modifying the speech,
4. modify the speech, and
5. generate the adaptive codebook contribution.

This is depicted in principle at a high level in Fig. 17.11, where the generation of the adaptive codebook contribution takes place inside the encoder like usual, except now according to a pitch contour as opposed to a fixed pitch. Note that the additional blocks in Fig. 17.11 as compared to Fig. 17.9 belong to the encoder just like the *signal modifier* in Fig. 17.10. Furthermore, the decoder would comprise identical functions to

1. create the pitch contour, and
2. generate the adaptive codebook contribution according to the pitch contour.

Note that Fig. 17.11 should be viewed at a conceptual level as the actual signal modification could take place in any domain, but with a corresponding modified input signal.

Estimate Open-Loop Pitch. It is critical for optimal performance of RCELP to get a good estimate of the pitch period. Typically, it is estimated in an open loop manner at the boundary of each frame. The pitch period towards the end of frame m is denoted by $\mathrm{pp}(m)$.

Create Pitch Contour. In the general case where the pitch period of the previous frame, $\mathrm{pp}(m-1)$, and the current frame $\mathrm{pp}(m)$ are relatively close, the pitch period contour is created as a continuous interpolation between the two. A simple continuous interpolation is the linear interpolation between the two pitch periods:

$$\mathrm{ppc}(m,n) = \frac{n+1}{N+1}\mathrm{pp}(m) + \frac{N-n}{N+1}\mathrm{pp}(m-1)\,,$$
$$n = 0, 1, \ldots, N-1\,. \qquad (17.35)$$

The continuous interpolated pitch period contour is referred as the interpolated pitch contour in the following.

Create Target Signal for Modifying the Speech. In order to modify the speech signal to follow the in-

Table 17.5 Properties of RCELP in standards

Standard	RCELP mode	Time-scale search		Time-scale modification		Max. delay drift
		Domain	Resolution	Domain	Method	
EVRC	Continuous	Short-term residual	(1/8)-th sample	Short-term residual	Interpolation	
SMV	Switched	Weighted speech	(1/10)-th sample	Weighted speech	Interpolation & block shifting	±2.5 ms
VMR-WB	Switched	Weighted speech	(1/8)-th sample	Short-term residual	Block shifting	0 ms

terpolated pitch period contour a target signal can be constructed. One possibility is to extrapolate the previously modified speech signal according to the pitch contour and use this as a target for modifying the time-scale of the speech signal of the current frame. Note that in practice, the time-scale modification of the speech signal can take place in any domain. As an example, in EVRC [17.26] it takes place in the short-term residual signal domain, in SMV [17.64] it takes place in the weighted speech signal domain, and in VMR-WB [17.28] the actual time-scale modification is done in the short-term residual signal domain, but the time-scale modification is calculated in the weighted speech signal domain.

Modify the Speech. The speech signal can be modified in multiple ways, and in multiple domains as mentioned above. The goal is to modify the time-scale of the speech signal so that it follows the interpolated pitch contour. With the target signal described above, this can be achieved by modifying the time-scale of the speech signal so as to maximize the correlation between the modified speech signal and the target signal. If the time-scale modification is carried out on a pitch period by pitch period basis, then it becomes a matter of identifying the pitch pulses and perform time-scale modification so that the pitch pulses of the modified speech signal lines up with the pitch pulses of the target signal. The time-scale modification can be carried out either by time warping in the form of resampling, time shifting of blocks, or a combination thereof.

Generate the Adaptive Codebook Contribution. The final step is to generate the adaptive codebook (or equivalently, the pitch predictor) contribution. Based on the past short-term synthesis filter excitation the adaptive codebook contribution is generated by signal interpolation according to the interpolated pitch contour. Fractional resolution in the order of 1/8-th is generally used, and truncated sinc windows are used for the signal interpolation.

A number of practical issues need to be considered when using RCELP:

1. pitch dependence
2. delay drift
3. pitch doubling and halving
4. complexity

Pitch Dependence. Not surprisingly, there is a strong dependency on the pitch period estimation. If the pitch period estimation is not of sufficient accuracy, naturally, degradation in performance should be expected. Furthermore, RCELP works the best for voice speech with slow evolution of the pitch period. Accordingly, some coders have separate mode(s) with more frequent update of the pitch period for frames where a single interpolated pitch contour is not expected to provide satisfactory performance.

Delay Drift. Generally, a variable delay is introduced during the signal modification. This contributes to the overall system delay, and it may drift. Typically, a constraint on the maximum delay drift of about 3 ms is enforced during RCELP. One exception is VMR-WB where the pitch contour is constrained to provide perfect time synchrony between the original speech and the modified speech at frame boundaries. This means that there is no additional delay from RCELP in this implementation.

Pitch Doubling and Halving. Pitch doubling and halving can present a significant challenge if RCELP is used continuously. For multimode coders an alternative mode with more frequent pitch period updates without pitch period interpolation could be used in such cases. However, solutions to continuously use RCELP also for pitch doubling and halving are discussed in [17.70].

Complexity. As reported in [17.70] the complexity of initial versions was very high. Some initial experimental versions were reported to be 75 million operations per second (MOPS), but projections indicated solutions with

a complexity of about 15 MOPS would be achievable. Such solutions have since materialized in the form of standards such as EVRC, SMV, and VMR-WB which are all around 40 weighted MOPS (WMOPS) or less for full duplex encoding and decoding.

17.13.2 RCELP in Standards

RCELP is utilized in a number of standards to efficiently represent the pitch period: TIA-127 EVRC [17.26], TIA/3GPP2 SMV [17.64], 3GPP2 VMR-WB [17.28], and it was part of a proposal for the ITU-T 4 kb/s standardization effort [17.8] before that effort was abandoned by the ITU. Although, those standards implement RCELP in different ways and variations, they all achieve the same goal of reducing the bit rate required to represent the pitch period. Some apply RCELP continuously while others apply RCELP only for specific modes. Some of the RCELP features of these standards are summarized in Table 17.5.

17.14 eX-CELP

The eXtended CELP (eX-CELP) technique [17.71] is more a collection of techniques or a general approach than a single specific technique. The general idea is to emphasize the perceptual important features during encoding within the context of analysis-by-synthesis. Basically, the closed-loop waveform matching of analysis-by-synthesis is relaxed by combining open-loop and closed-loop control. The necessity to relax the strict waveform matching originates from the observation that at low bit rate it is not possible to achieve toll quality through strict waveform matching, and instead emphasis on the perceptually important features is necessary. This is achieved by incorporating signal classification into weighting, using RCELP-like techniques, combining open-loop and closed-loop, designing the fixed codebook with multiple sub-codebooks addressing different signal characteristics, and utilizing multimode encoding.

Signal Classification. eX-CELP uses elaborate signal classification. As many as six classes are used [17.27]:

- silence/background noise
- stationary unvoiced
- nonstationary unvoiced
- onset
- nonstationary voiced
- stationary voiced

The classification takes place in multiple stages and is refined as information becomes available in the algorithm.

Signal Modification. The signal modification algorithm not only modifies the pitch contour to allow a lower bit rate for the encoding of the pitch period like RCELP, it also modifies the speech signal to increase the pitch contribution of the adaptive codebook. According to [17.71], it uses waveform interpolation or harmonic smoothing to pre-smooth voiced transition areas in an open-loop manner. This results in faster buildup of the adaptive codebook. The underlying objective is to increase/enhance the pitch contribution as much as possible without introducing noticeable distortion to the signal.

Combination of Open-Loop and Closed-Loop. The algorithm combines open-loop and closed-loop extensively. One example is the gain quantization. It is well known that for background noise, accurate waveform matching is not necessary for a perceptual accurate reproduction. However, features such as a natural (smooth) energy contour and a dense excitation is more important. Hence, in the presence of background noise the gain quantization favors open-loop over close-loop, and it aims at maintaining a smooth energy contour. Similarly, it would favor selection of excitation from a relatively denser codebook.

Fixed Codebook. The fixed codebook is made up of multiple sub-codebooks of varying pulse density and even a sub-codebook of dense random-like noise. Each sub-codebook is designed and tuned with a certain type of signal or speech in mind. Each codebook is generally searched according to the analysis-by-synthesis principle, but the selection between the best outputs of the various sub-codebooks is influenced by the classification information, and other signal parameters such as estimate of background noise level and *peakiness* of the speech.

Multimode. eX-CELP distinguishes between two fundamentally different encoding modes, types 0 and 1.

Table 17.6 Key general features of type 0 and type 1 coding modes in eX-CELP

Entity	Type 0	Type 1
Signal character	All other	Staitonary voiced
Subframes	Fewer	More
Pitch period (adaptive codebook)	Subframe based	Frame based
Fixed codebook	Pulse & Gaussian sub-codebooks	Pulse sub-codebooks
Adaptive codebook gains	Joint 2-D VQ (subframe based)	A priori VQ (frame based)
Fixed codebook gains	Joint 2-D VQ (subframe based)	A posteriori VQ (frame based)

Table 17.7 MOS of SMV at various operating points [17.71]

Coder	Average bit rate	MOS (clean speech)	MOS (noisy speech)
EVRC	x	3.58	3.35
SMV Mode 0	$1.0 \cdot x$	3.90	3.57
SMV Mode 1	$0.71 \cdot x$	3.64	3.53
SMV Mode 2	$0.56/0.60 \cdot x$	3.46	3.53

Table 17.8 MOS of 4 kbit/s eX-CELP [17.8]

Input	Coder	MOS
MIRS (modified intermediate reference system)	G.726	3.14
	G.729	3.43
	4 kb/s eX-CELP	3.36
Flat	G.726	3.12
	G.729	3.27
	4 kb/s eX-CELP	3.32

The bit allocation, the quantization methods, and even the subframe structure for the two modes can be different. In type 1, targeting stationary voiced speech, the subframe adaptive codebook gains are prequantized open-loop using VQ prior to subframe processing, while fixed codebook gains are left unquantized during subframe processing and quantized jointly using VQ after subframe processing is complete. For type 0, a more-conventional joint two-dimensional VQ of the subframe based adaptive and fixed codebook gains is used. However, the gain estimation and quantization is still not strictly analysis-by-synthesis for type 0, as focus remains on producing a perceptually faithful output, mixing open-loop and closed-loop. Key general features of the two types are listed in Table 17.6.

17.14.1 eX-CELP in Standards

The 3GPP2 SMV standard [17.64] is based on eX-CELP. Table 17.7 is reproduced from [17.71] and compares the MOS scores of SMV at various operating points of quality versus average bit rate with EVRC. The third column contains the MOS scores for clean speech, and the fourth column contains the MOS scores of speech in background noise. Also one of the promising ITU-T 4 kbit/s candidates [17.8] was based on eX-CELP. Some MOS scores comparing the performance to 32 kbit/s G.726 and 8 kbit/s G.729 are reproduced in Table 17.8.

17.15 iLBC

The iLBC coder [17.31, 32] is quite different from other analysis-by-synthesis speech coders. Most other analysis-by-synthesis speech coders use a long-term predictor or an adaptive codebook continuously and across the frame boundaries. Due to the relatively large bulk delay in repeating the previous waveform stored in the long-term predictor memory or the adaptive codebook, if a frame is lost, the degrading effect tends to propagate a while. The iLBC coder attempts to address this issue and improve the robustness against frame loss. It achieves this by employing block-independent coding of the adaptive codebook. In other words, the adaptive codebook of each speech frame is encoded independent of the previous frames. Therefore, if a frame is lost, the quality degrading effect due to a mismatched adaptive codebook is limited to the lost frame and will not propagate to the future frames.

Of course, the improved robustness does not come for free. By not allowing the adaptive codebook to be used across the frame boundaries, the iLBC coder gives up the opportunity to exploit the corresponding long-term redundancy in the speech signal across the frame boundaries. Therefore, for clear channel it inevitably sacrifices the output speech quality to some extent when compared with more-conventional CELP coders. In effect, the iLBC coder makes a trade-off between the

speech quality in clear-channel and the speech quality under frame loss conditions. Given that the iLBC coder was developed for voice over internet protocol (VoIP) over the general Internet that can have a fairly high packet loss rate, it is reasonable for iLBC to make such a trade-off.

The iLBC coder was standardized as an IETF experimental standard [17.32]. It has two versions: a 13.3 kb/s version with a 30 ms frame size and 10 ms look-ahead, and a 15.2 kb/s version with a 20 ms frame size and 5 ms look-ahead. The two versions divide each speech frame into 5 ms subframes. The short-term prediction residual signal is first calculated. The two consecutive subframe of residual having the largest weighted energy are identified. Within these two subframes, the *start state (segment)* is selected as either the first S samples or the last S samples of the two consecutive subframes, depending on which segment has a higher energy. The integer S is 57 or 58 samples, depending on whether it is the 20 ms version of iLBC or the 30 ms version. The adaptively selected start state is encoded with scalar quantization without using information in the previous frames.

A dynamic codebook encoding procedure first encodes the remaining samples in the two subframes containing the start state. Next, it encodes the remaining subframes forward in time, and then it encodes the remaining subframes backward in time. The maximum-energy selection criterion allows iLBC to capture the pitch epoch of a pitch cycle waveform or the onset of voiced segments as the start state. The forward–backward approach mentioned above then allows an adaptive codebook to be constructed and used in the entire current frame without using the adaptive codebook in the previous frames.

There are further details in the codebook search method and how iLBC encodes the codebook gains and re-scale the gain for power matching after encoding. Interested readers are referred to [17.32].

It should be noted that even though iLBC performs frame-independent coding of the adaptive codebook, the iLBC coder is not completely frame independent. At least the short-term synthesis filter memory from the previous frame is still used when starting the encoding and decoding operation in the current frame. This means that iLBC still has a slight error propagation from one frame to the next, although the degree of error propagation is significantly smaller than that of the other conventional CELP coders.

17.16 TSNFC

Two-stage noise feedback coding (TSNFC) [17.72] is a relatively new class of analysis-by-synthesis coders although it builds on the decade-old technique of noise feedback coding (NFC) [17.16, 73]. The ANSI Amer-

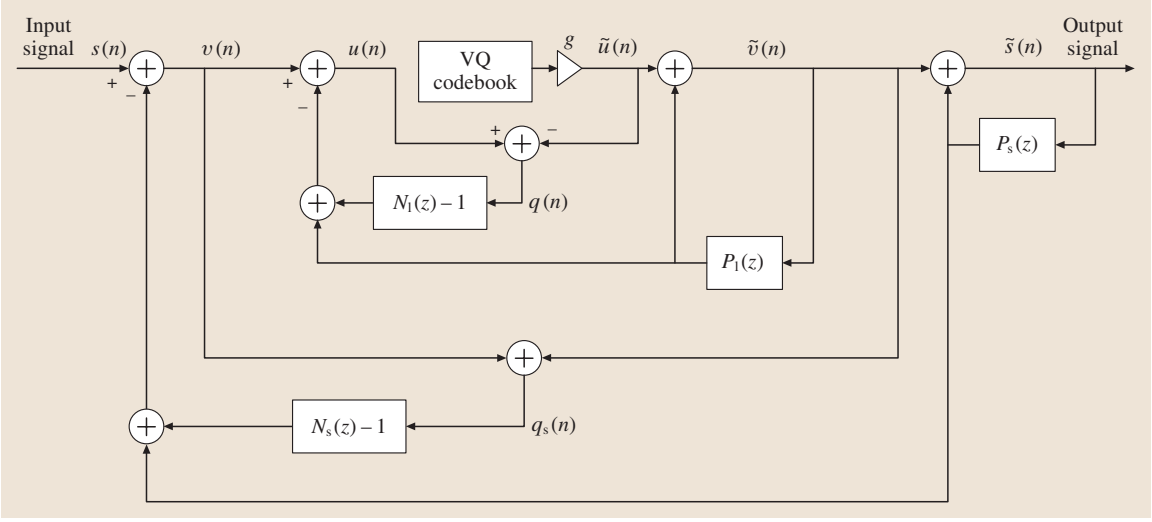

Fig. 17.12 TSNFC encoder structure

ican national standard BV16 coder [17.33] and the similar wide-band version BV32 are based on TSNFC. While the original NFC uses sample based quantization of the excitation signal and has less in common with analysis-by-synthesis, the recent extension of NFC to TSNFC and the development of CELP-like VQ techniques [17.72, 74] for vector quantization of the excitation signal have created a commonality between TSNFC and CELP, making a brief overview in the context of analysis-by-synthesis relevant.

One structure for TSNFC is shown in Fig. 17.12. It clearly looks nothing like a CELP coder, compare to Fig. 17.5. The short-term predictor, $P_s(z)$, is equivalent to the short-term predictor of CELP coders and results in a short-term prediction error filter given by

$$A(z) = 1 - P_s(z)\,. \tag{17.36}$$

Similarly, the long-term predictor $P_l(z)$ corresponds to the pitch predictor of CELP (or the adaptive codebook). Before TSNFC is described in more detail another interesting observation in relation to analysis-by-synthesis can be made. Studying the TSNFC encoder structure in Fig. 17.12 one can observe the decoder output speech, $\tilde{s}(n)$, being generated as part of the quantization process of the excitation. Conceptually, every codevector of the VQ codebook is fed through the structure in Fig. 17.12 and the candidate minimizing the MSE of $q(n)$ is selected as the output of the VQ. With a short-term noise feedback filter given by $N_s(z) - 1$, minimizing the MSE of $q(n)$ will result in a spectral envelope of the quantization noise, $e(n) = \tilde{s}(n) - s(n)$ given by $N_s(z)$. Hence, the spectral envelope of the coding noise can be controlled directly by $N_s(z)$. This has a parallel to the CELP coder where the spectral envelope of the coding noise is controlled by the inverse of the perceptual weighting filter. Hence, a CELP coder with a perceptual weighting filter of

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)}\,, \quad 0 < \gamma_2 < \gamma_1 < 1\,, \tag{17.37}$$

and a TSNFC coder with a short-term noise feedback filter of

$$N_s(z) - 1 = \frac{A(z/\gamma_2)}{A(z/\gamma_1)} - 1\,, \quad 0 < \gamma_2 < \gamma_1 < 1 \tag{17.38}$$

will both result in a coding noise with a spectral envelope shaped according to

$$N_s(z) = \frac{A(z/\gamma_2)}{A(z/\gamma_1)}\,. \tag{17.39}$$

Similarly, the TSNFC structure in Fig. 17.12 will result in a coding noise with spectral fine structure shaped according to $N_l(z)$. A suitable choice according to [17.33] is

$$N_l(z) = 1 + \beta z^{-T}\,, \tag{17.40}$$

where β is the long-term noise feedback filter coefficient and T is the pitch period. Typically, β is adaptively controlled by the pitch analysis, and $0 \le \beta \le 1$.

Derivation of short-term predictor coefficients and quantization thereof can be carried out with appropriate techniques from the literature, similarly with the estimation and quantization of the long-term predictor parameters. The methods and structures for VQ of the excitation and the similarities to analysis-by-synthesis in CELP coders will be discussed in more detail in the following.

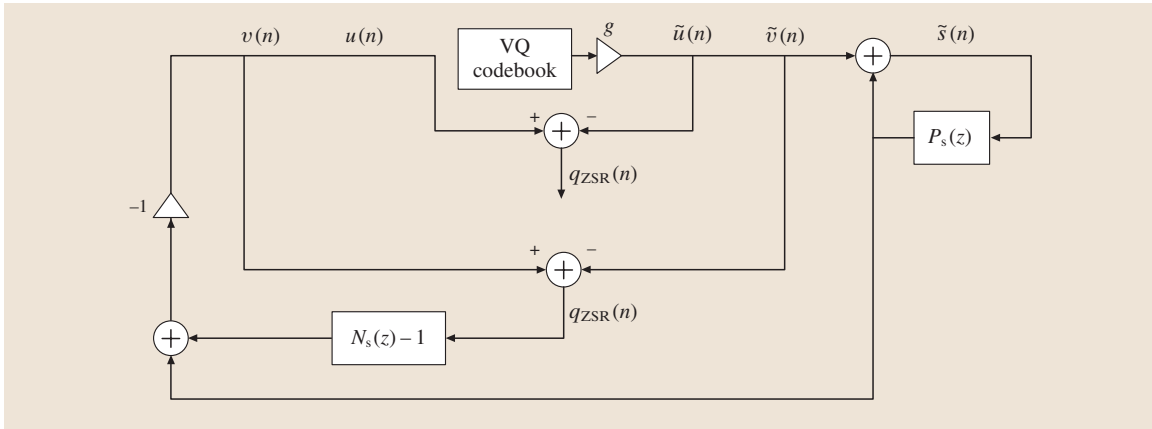

Fig. 17.13 TSNFC ZSR structure

17.16.1 Excitation VQ in TSNFC

Similar to CELP it is advantageous to apply the superposition principle when calculating the error signal, $q(n)$, for the excitation VQ. It is calculated as the superposition of the zero state response (ZSR), $q_{\mathrm{ZSR}}(n)$, and the zero-input response (ZIR), $q_{\mathrm{ZIR}}(n)$ [17.72]. With a minimum pitch lag being greater than the dimension of the excitation VQ, the structure in Fig. 17.12 reduces to the ZSR structure in Fig. 17.13. This structure is simplified as shown in Fig. 17.14 according to [17.74], where $H(z)$ with the short-term noise feedback filter of (17.38) is given by

$$H(z) = \frac{Q_{\mathrm{ZSR}}(z)}{\tilde{U}(z)} = -\frac{1}{N_{\mathrm{s}}(z)A(z)} = -\frac{A(z/\gamma_1)}{A(z/\gamma_2)A(z)} \,. \tag{17.41}$$

The ZIR structure is derived from Fig. 17.12 as shown in Fig. 17.15. Typically in TSNFC, a relatively short dimension of the excitation VQ is used. In BV16 [17.33] a 4-dimensional excitation VQ is used. The frame size is 40 samples (5 ms) during which $A(z)$ remains constant. Hence, the codebook needs only to be filtered with $H(z)$ according to Fig. 17.14 once per frame, or equivalently, once per 10 codebook searches. In other words, the filtered codebook is generated only once per 10 codebook searches. The filtered codebook vectors are denoted $q_{\mathrm{ZSR}}(k, n)$, where k indicates the codebook index, $k = 0, 1, \ldots, K-1$. On the other hand, the ZIR, $q_{\mathrm{ZIR}}(n)$ needs to be generated for every four-sample excitation vector. However, it is independent of the codebook vectors. Hence, on a frame basis:

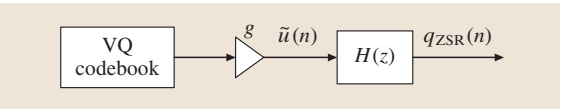

Fig. 17.14 Simplified TSNFC ZSR structure

- the filtered codebook vectors need to be generated once,
- the 10 ZIR vectors needs to be generated, once for each four sample excitation vector,
- and after each of the 10 codebook searches and before the next ZIR calculation the filter memories need to be updated.

Efficient methods for updating the filter memory is presented in [17.74]. The codebook search is carried out efficiently according to

$$k_{\mathrm{opt}} = \arg\min_k \left\{ \sum_{n=0}^{N-1} [q_{\mathrm{ZIR}}(n) + q_{\mathrm{ZSR}}(k, n)]^2 \right\} \,, \tag{17.42}$$

where $N = 4$ in the example above. With a signed codebook as in BV16, and using vector form notation, this reduces to

$$k_{\mathrm{opt}} = \arg\min_k \left\{ E\left[\boldsymbol{q}_{\mathrm{ZSR}}(k)\right] \overset{s}{\pm} R\left[\boldsymbol{q}_{\mathrm{ZSR}}(k), \boldsymbol{q}_{\mathrm{ZIR}}\right] \right\} \,, \tag{17.43}$$

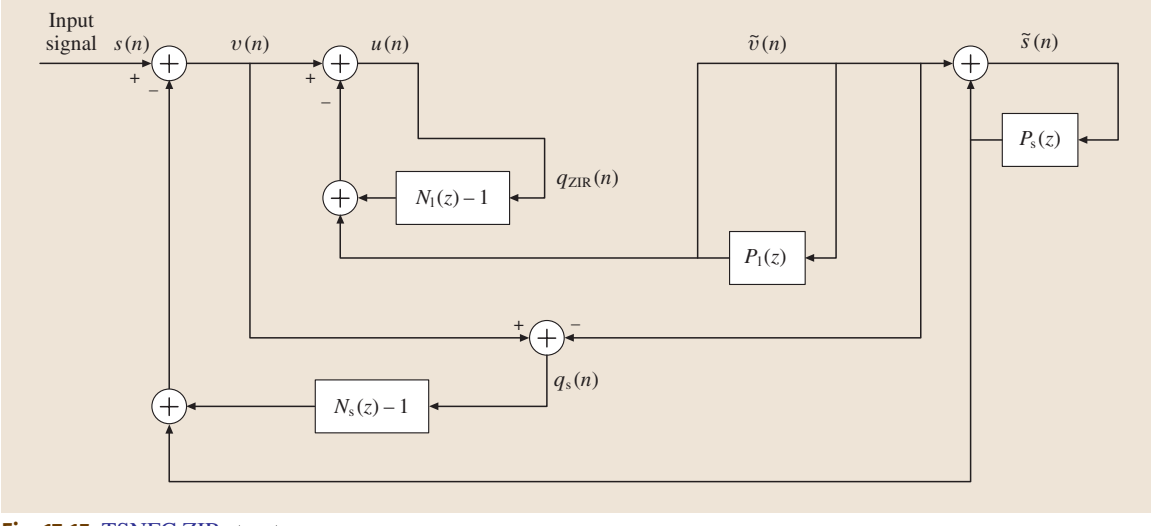

Fig. 17.15 TSNFC ZIR structure

Table 17.9 MOS of BV16

Coder	MOS
G.711	3.91
G.726	3.56
G.728	3.54
G.729	3.56
BV16	3.76

Table 17.10 MOS of BV32

Coder	MOS
G.722 @ 48 kb/s	3.60
G.722 @ 56 kb/s	3.88
G.722 @ 64 kb/s	3.96
BV32	4.11

where s indicates the sign, and

$$E\left[\boldsymbol{q}_{\mathrm{ZSR}}(k)\right] = \sum_{n=0}^{N-1} q_{\mathrm{ZSR}}^2(k,n)$$

$$R\left[\boldsymbol{q}_{\mathrm{ZSR}}(k), \boldsymbol{q}_{\mathrm{ZIR}}\right] = \sum_{n=0}^{N-1} q_{\mathrm{ZSR}}(k,n) \cdot q_{\mathrm{ZIR}}(n)\,.$$

Note that the optimal sign is given as the opposite sign of the correlation, and hence, effectively, only half the codevectors need to be searched. Similarly, with a signed codebook only half the codevectors need to be filtered with $H(z)$ as the other half of filtered codevectors are given by simple negation. In (17.43), $E\left[\boldsymbol{q}_{\mathrm{ZSR}}(k)\right]$ can be pre-computed before the first of the 10 excitation vectors is quantized. Hence, the only search-loop computation ends up being the calculation of $R\left[\boldsymbol{q}_{\mathrm{ZSR}}(k), \boldsymbol{q}_{\mathrm{ZIR}}\right]$. These techniques are very similar to techniques used in CELP coding, but now applied to the TSNFC structure.

17.16.2 TSNFC in Standards

The PacketCable, SCTE, and ANSI BV16 standard for voice over cable is based on TSNFC. BV16 is a narrow-band coder and has an algorithmic delay of 5 ms, a bit rate of 16 kbit/s, a complexity comparable to G.729A (significantly lower than G.729 and G.728). Results from a formal subjective test of BV16 are summarized in Table 17.9. CableLabs has also included BV32 [17.30], a wide-band version of BV16, in PacketCable 2.0 codec and media specification [17.75]. BV32 also has an algorithmic delay of 5 ms, a bit-rate of 32 kbit/s, a complexity comparable to G.729 (significantly lower than G.722.2/AMR-WB), and a subjective quality better than 64 kbit/s G.722. Formal subjective test results of BV32 are summarized in Table 17.10.

17.17 Embedded CELP

The recently standardized ITU-T G.729.1 coder [17.76] is a bit-rate- and bandwidth-scalable embedded coder based on G.729. The 16 kHz sampled wide-band input speech is split into equal-bandwidth high and low bands with a quadrature mirror filterbank (QMF). The low-band (narrow-band) signal is encoded with the narrow-band core (also referred as Layer 1) layer. Although the frame size of G.729.1 is 20 ms as opposed to the 10 ms frame size of G.729, the core layer is bit-stream compatible with G.729. However, G.729.1 has a significantly longer algorithmic delay. The algorithmic delay of G.729 is 15 ms while the algorithmic delay of G.729.1 is 48.9375 ms. Layer 2 is a 4 kb/s narrow-band embedded enhancement layer to G.729. Layers 3–12 provide wide-band (50–7000 Hz) capability at a total bit rate increasing from 14 kbit/s to a maximum of 32 kb/s at increments of 2 kb/s per layer. An overview of the encoder is presented in Fig. 17.16. Layer 3 encodes the high-band signal using time-domain bandwidth extension (TDBWE) [17.76]. Layers 4–12 encode the weighted error signal from the narrow-band layer 1 and 2 embedded CELP coder, 50–4000 Hz, jointly with the high-band signal, 4000–7000 Hz. A transform predictive coder (TPC) with time-domain aliasing cancelation (TDAC) is utilized for layers 4–12. The coder sends redundant information to the decoder in order to mitigate

Table 17.11 Overview of G.729.1 embedded CELP

Layer	Technology	Analysis by synthesis	Bandwidth	Cummulative bit rate
1	ACELP	Yes	50–4000 Hz	8 kb/s
2	ACELP	Yes	50–4000 Hz	12 kb/s
3	TDBWE	No	4000–7000 Hz	14 kb/s
4–12	TPC	No	50–7000 Hz	16–32 kb/s

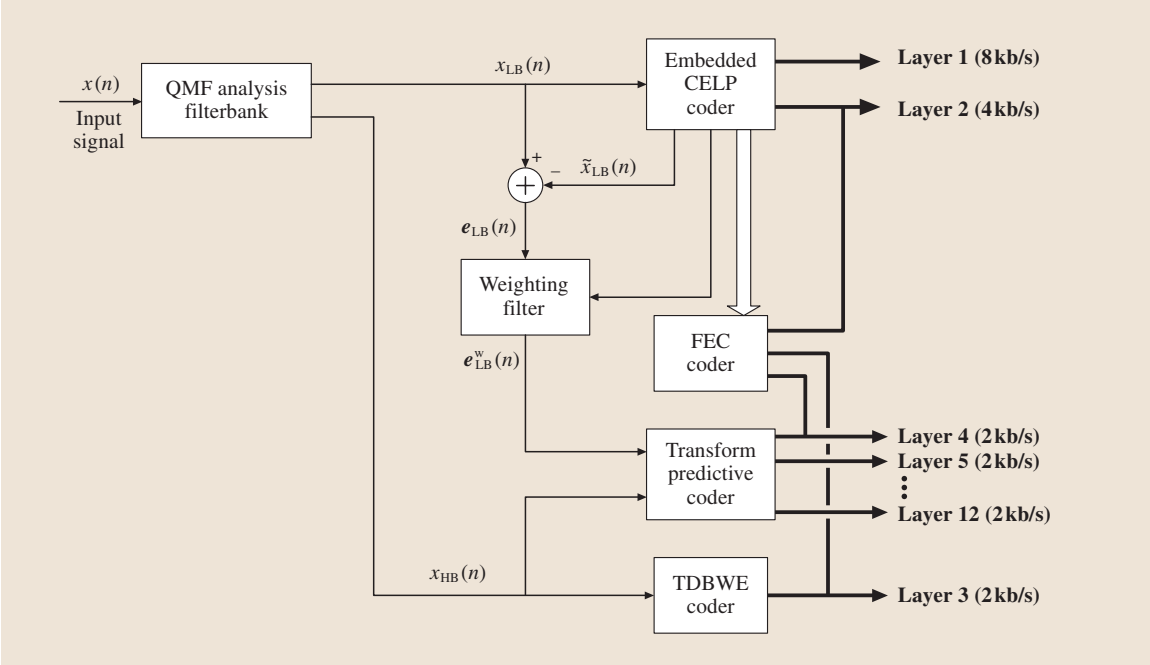

Fig. 17.16 High-level block diagram of G.729.1

the effect of frame loss. This information is calculated with the frame erasure concealment (FEC) encoder and comprises information pertaining to the narrow-band embedded CELP coder in order to speed up recovery of the CELP decoder after frame loss. The information includes

- signal classification information
- estimated position of the last glottal pulse
- energy calculated as either maximum or average sample energy (dependent on signal classification)

Table 17.12 elaborates the FEC encoded information with bit-allocation and information regarding which layer includes the additional information. It should be noted that the FEC information is only available if the layers listed in Table 17.12 are actually received by the decoder.

Table 17.12 Redundant information included in G.729.1 bitstream to help embedded CELP coder recover from frame loss

Information	Location	Bit allocation
Classification	Layer 2	2 bits
Glottal pulse location	Layer 3	7 bits
Energy	Layer 4	5 bits

Since the TDBWE and transform predictive coders are not analysis-by-synthesis coders, they will not be treated in any greater details here. Instead, the reader is referred to [17.76] for further information.

The embedded CELP coder is, however, of interest as not only the core layer, but also the enhancement layer, is derived according to analysis-by-synthesis and presents a particular problem. In the following, the terms encoder and decoder are used to refer to the embedded CELP only encoder and decoder portions of G.729.1. The particular problem pertains to the issue of ensuring that the encoder and decoder remain synchronized regardless of the number of decoded layers. Consider the fundamental meaning of the term analysis-by-synthesis. Regardless of the number of decoded layers the coder must remain an analysis-by-synthesis coder:

At the encoder, without knowing how many layers the decoder will decode, how is analysis-by-synthesis maintained?

This appears to be an oxymoron, but the key is to realize that the encoder has multiple layers of encoding, and then structure the algorithm such that the various levels of encoding remain synchronized with the various levels of decoding. This may sound simple. However, looking at an example, it quickly becomes clear that it is not quite that simple. Let us assume that

a CELP enhancement layer includes a fixed codebook. The straightforward way would be to update the adaptive codebook at the encoder with a short-term excitation signal including this enhancement layer contribution. This would be optimal in the sense of the output speech quality including the enhancement layer. However, at the decoder this would present a problem if the enhancement layer is not received as the adaptive codebook would then no longer be synchronized with the encoder. The analysis-by-synthesis property would be violated: in retrospect the encoder did not do the same synthesis part in the analysis-by-synthesis process as the decoder. Once the analysis-by-synthesis property is violated the speech quality is somewhat up in the air, and the waveform matching during analysis-by-synthesis is somewhat senseless as the decoder will not repeat the same synthesis.

Not only does the example illustrate the particular issue of embedded analysis-by-synthesis, it also demonstrates one reason why an embedded analysis-by-synthesis coder would be of inferior speech quality to a nonembedded analysis-by-synthesis coder: the obvious optimal approach, in terms of optimizing the speech quality of the output including the enhancement layer, had to be disregarded in order not to violate the analysis-by-synthesis property.

To maintain the analysis-by-synthesis property for embedded analysis-by-synthesis, the memory of one layer cannot be updated with any information of a higher layer. In the example above, with an enhancement layer containing an additional fixed codebook, it means that the adaptive codebook of the lower layer (the core layer) can only be updated with the short-term synthesis filter excitation without the fixed codebook contribution of the enhancement layer. Furthermore, at the encoder, separate short-term synthesis filter memory may need to be maintained. Another example, albeit perhaps not truly analysis-by-synthesis, is the ADPCM encoder of G.726 [17.77] and the low-band ADPCM encoder of G.722 [17.78]. Both are embedded backward adaptive predictive coders. In G.722 the low-band prediction error signal is quantized sample by sample with six bits. However, the low-band ADPCM decoder can decode based on four, five, or six bits per sample. In order for the backward adaptive entities of the low-band ADPCM to stay synchronized between encoder and decoder at all times, the encoder and decoder only use the four-bit information for updating the backward adaptive entities of the low-band ADPCM.

17.18 Summary of Analysis-by-Synthesis Speech Coders

Table 17.13 lists and compares the properties of the various analysis-by-synthesis techniques and coders. Each of the Sects. 17.4 through 17.17, with detailed discussions of the various analysis-by-synthesis techniques, summarize which standards utilize the techniques. This section will view it from a different angle, and for individual speech coding standards list the applicable techniques and features. Note that multiple techniques apply to some of the standards. Furthermore, key characteristics of the coders are provided as well in order to put the techniques into context and provide the reader with a general overview of properties of speech coding standards. It should be noted that the scope was limited to analysis-by-synthesis coders, and that space limitations prevent inclusion of every standard speech coder. Note that ACB for 'pitch type' indicates the use of the adaptive codebook approach to implement the pitch prediction, while 'filter' indicates implementation as a regular filter. The term 'frac. lag' indicates usage of fractional pitch lag. Furthermore, 'excitation' refers to the fixed codebook (or innovation) excitation, and hence, excludes the pitch related part of the excitation. The bit rate for the 'excitation' in Table 17.13 also excludes the gain for the fixed codebook excitation. The column 'year standardized' should only be considered approximate, as there is some variation in the years listed in the literature due to a common delay between the initial selection of a speech coding algorithm, and the date it is officially standardized. Furthermore, the 'complexity' should only be considered a rough guideline as great variations for a given algorithm exist. Such variation can be due to many factors, e.g., capability of target processor, level of optimization, etc. Also, it should be noted that the unit for the complexity is either WMOPS or MIPS. The WMOPS number is typically obtained from a simulation software in C code with operators with weights emulating the complexity of each operator on a typical digital signal processor (DSP). On the other hand, the MIPS number is typically obtained from an actual implementation on a specific DSP. Although the purpose of the WMOPS number is to estimate the complexity on a DSP (the MIPS number), it does not always provide an exact number.

Table 17.13 Overview of analysis-by-synthesis speech coding standards

Coder	Year standardized	Standards organization	Bit rate	Audio bandwidth	Algorithmic delay	Complexity	Excitation type	Excitation dimension	Excitation bit rate	Pitch type	Other AbS techniques
GSM-FR	1986	ETSI	13 kb/s	0–4 kHz	20 ms	4.5 MIPS	RPE	40	8.2 kb/s	Filter	
FS-1016	1989	DoD	4.8 kb/s	0–4 kHz	30 ms	25 MIPS	Overlap sparse	60	1.33 kb/s	ACB	
IS-54	1989	TIA	8 kb/s	0–4 kHz	28.125 ms	25 MIPS	VSELP	40	2.8 kb/s	ACB	
G.728	1992	ITU-T	16 kb/s	0–4 kHz	0.625 ms	36 MIPS	Trained	5	11.2 kb/s	None	LD-CELP
GSM-HR	1993	ETSI	5.6 kb/s	0–4 kHz	25 ms	25 MIPS	VSELP	40	1.8 kb/s 2.8 kb/s	ACB fraction lag	
PDC-HR	1993	ARIB	3.45 kb/s	0–4 kHz	45 ms	18.7 MOPS	PSI	80		ACB fraction lag	
EVRC	1994	TIA	8.55 kb/s 4.0 kb/s	0–4 kHz	33 ms	25 MIPS	ACELP	53/43	5.25 kb/s 1.5 kb/s	ACB	RCELP
G.723.1	1995	ITU-T	6.3 kb/s 5.3 kb/s	0–4 kHz	37.5 ms	19 MIPS	MP ACELP	60	3.3 kb/s 2.27 kb/s	ACB 5-tap	
G.729	1996	ITU-T	8 kb/s	0–4 kHz	15 ms	22 MIPS	ACELP	40	3.4 kb/s	ACB fraction lag	CS-ACELP
TIA-EFR	1996	TIA	7.4 kb/s	0–4 kHz	25 ms	14 WMOPS	ACELP	40	3.4 kb/s	ACB fraction lag	
PDC-EFR	1996	ARIB	6.7 kb/s	0–4 kHz			ACELP	40			
GSM-EFR	1997	ETSI	12.2 kb/s	0–4 kHz	20 ms	18 MIPS	ACELP	40	7 kb/s	ACB fraction lag	
GSM-AMR	1999	ETSI	4.75 kb/s to 12.2 kb/s	0–4 kHz	25 ms	20 MIPS	ACELP	40	1.8 kb/s to 7 kb/s	ACB fraction lag	
SMV	2001	3GPP2	8.55 kb/s 4.0 kb/s	0–4 kHz	30.5–35.5 ms 27.5–32.5 ms	40 WMOPS	Sub ACELP	40 80, 53/54	4.4, 6 kb/s 1.5, 1.95 kb/s	ACB	eX-CELP RCELP
AMR-WB G.722.2	2001	3GPP ITU-T	6.6 kb/s to 23.85 kb/s	0–8 kHz	26 ms	38 WMOPS	ACELP	64	2.4–17.6 kb/s	ACB fraction lag	
VMR-WB	2002	3GPP2	6.6 kb/s 8.85 kb/s 12.65 kb/s	0–8 kHz	33.75 ms	38 WMOPS	ACELP	64	2.4–7.2 kb/s	ACB fraction lag	RCELP
iLBC	2002	IETF PacketCable	13.3 kb/s 15.2 kb/s	0–4 kHz	40 ms 25 ms	18 MIPS 15 MIPS	FB-LPC	40	12.0 kb/s 14.2 kb/s	FB-ACB	
BV16	2003	SCTE ANSI	16 kb/s	0–4 kHz	5 ms	12 MIPS	Trained	4	10 kb/s	Filter 3-tap	TSNFC
G.729.1	2006	ITU-T	8 kb/s to 32 kb/s	0–4 kHz 0–8 kHz	48.9375 ms	36 WMOPS	ACELP	40	3.4 kb/s+ 3.4 kb/s	ACB fraction lag	Embedded CS-CELP

17.19 Conclusion

The analysis-by-synthesis method for coding the excitation signal of linear predictive coders has been the most important driving force behind the relentless reduction of the bit-rate for high-quality speech coding in the last two decades. Almost all speech coding standards established after 1989 are based on it. This chapter gives a tutorial on analysis-by-synthesis speech coding techniques in general and describes many variations of the analysis-by-synthesis excitation coding paradigm in particular. Due to space limitation, the description concentrates on dominant types of analysis-by-synthesis excitation coding techniques as exemplified by various speech coding standards, with the relationship between them discussed in the context of a family tree. It is hoped that, after reading this chapter, the reader will have a good understanding of the dominant types of analysis-by-synthesis speech coding techniques.

References

17.1 R.V. Cox: Speech coding standards. In: *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier Science, Amsterdam 1995)

17.2 B.S. Atal, J.R. Remde: A new model of LPC excitation for producing natural-sounding speech at low bit rates, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1982) pp. 614–617

17.3 B.S. Atal, M.R. Schroeder: Stochastic coding of speech signals at very low bit rates, Proc. IEEE Int. Conf. Commun. (1984) p. 48.1

17.4 M.R. Schroeder, B.S. Atal: Code-excited linear prediction (CELP): high quality speech at very low bit rates, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1985) pp. 937–940

17.5 J.-H. Chen: A robust low-delay CELP speech coder at 16 kbit/s, Proc. IEEE Global Commun. Conf. (1989) pp. 1237–1241

17.6 J.-H. Chen, R.V. Cox, Y.-C. Lin, N.S. Jayant, M.J. Melchner: A low-delay CELP coder for the CCITT 16 kb/s speech coding standard, IEEE J. Sel. Areas Commun. **10**(5), 830–849 (1992)

17.7 R. Salami, C. Laflamme, J.-P. Adoul, A. Kataoka, S. Hayashi, T. Moriya, C. Lamblin, D. Massaloux, S. Proust, P. Kroon, Y. Shoham: Design and description of CS-ACELP: a toll quality 8 kb/s speech coder, IEEE Trans. Speech Audio Process. **6**(2), 116–130 (1998)

17.8 J. Thyssen, Y. Gao, A. Benyassine, E. Shlomot, C. Murgia, H.-Y. Su, K. Mano, Y. Hiwasaki, H. Ehara, K. Yasunaga, C. Lamblin, B. Kovesi, J. Stegmann, H.-G. Kang: A candidate for the ITU-T 4 kbit/s speech coding standard, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2001) pp. 681–684

17.9 A. McCree, J. Stachurski, T. Unno, E. Ertan, E. Paksoy, R. Viswanathan, A. Heikkinen, A. Ramo, S. Himanen, P. Blocher, O. Dressler: A 4 kb/s hybrid MELP/CELP speech coding candidate for ITU standardization, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2002) pp. 629–632

17.10 P. Kroon, W.B. Kleijn: Linear-prediction based analysis-by-synthesis coding. In: *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier Science, Amsterdam 1995)

17.11 M. Halle, K.N. Stevens: Analysis by synthesis, Proc. Sem. Speech Compression and Process., Vol. II, ed. by W. Wathen-Dunn, L.E. Woods (1959), AFCRC-TR-59-198, Paper D7

17.12 C.G. Bell, H. Fujisaki, J.M. Heinz, K.N. Stevens, A.S. House: Reduction of speech spectra by analysis-by-synthesis techniques, J. Acoust. Soc. Am. **33**(12), 1725–1736 (1961)

17.13 N.S. Jayant, P. Noll: *Digital Coding of Waveforms* (Prentice Hall, Englewood Cliffs 1984)

17.14 L.R. Rabiner, R.W. Schafer: *Digital Processing of Speech Signals* (Prentice Hall, Englewood Cliffs 1978)

17.15 W.B. Kleijn, R.P. Ramachandran, P. Kroon: Generalized analysis-by-synthesis coding and its application to pitch prediction, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1992) pp. I–337–I–340

17.16 B.S. Atal, M.R. Schroeder: Predictive coding of speech signals and subjective error criteria, IEEE Trans. Acoust. Speech Signal Process. **3**, 247–254 (1979)

17.17 I.A. Gerson, M.A. Jasiuk: Vector sum excited linear prediction (VSELP) speech coding at 8 kbps, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1990) pp. 461–464

17.18 P. Vary, K. Hellwig, R. Hofmann, R.J. Sluyter, C. Galand, M. Rosso: Speech codec for the European mobile radio system, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1988) pp. 227–230

17.19 S. Singhal, B. Atal: Improving performance of multi-pulse LPC coders at low bit rates, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1984) pp. 9–12

17.20 J.P. Campbell Jr., V.C. Welch, T.E. Tremain: An expandable error-protected 4800 bps CELP coder

(U.S. Federal standard 4800 bps voice coder), Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1989) pp. 735–738
17.21 I.A. Gerson, M.A. Jasiuk: A 5600 bps VSELP speech coder candidate for half-rate GSM, Proc. 1993 IEEE Workshop Speech Coding for Telecommunications (1993) pp. 43–44
17.22 T. Ohya, H. Suda, T. Miki: 5.6 kbits/s PSI-CELP of the half-rate PDC speech coding standard, Proc. 1994 IEEE Vehicular Technol. Conf. (1994) pp. 1680–1684
17.23 J.-P. Adoul, P. Mabilleau, M. Delprat, S. Morisette: Fast CELP coding based on algebraic codes, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1987) pp. 1957–1960
17.24 C. Laflamme, J.-P. Adoul, H.Y. Su, S. Morisette: On reducing computational complexity of codebook search in CELP coder through the use of algebraic codes, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1990) pp. 177–180
17.25 ITU-T: G.723.1: Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s (1996)
17.26 ANSI/TIA-127-A-2004: Enhanced variable rate codec speech service option 3 for wideband spread spectrum digital systems (2004) (ANSI/TIA-127-A-2004)
17.27 Y. Gao, E. Shlomot, A. Benyassine, J. Thyssen, H. Su: The SMV algorithm selected by TIA and 3GPP2 for CDMA applications, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2001) pp. 709–712
17.28 3GPP2 C.S0052-0 V1.0: Source-controlled variable-rate multimode wideband speech codec (VMR-WB) service option 62 for spread spectrum systems, (June 11 2004)
17.29 J.-H. Chen, J. Thyssen: BroadVoice16: A PacketCable speech coding standard for cable telephony, Proc. 40th Asilomar Conf. Signals Systems and Computers (2006)
17.30 J.-H. Chen, J. Thyssen: The BroadVoice speech coding algorithm, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 4 (2007) p. IV-549-IV-552
17.31 S.V. Andersen, W.B. Kleijn, R. Hagen, J. Linden, M.N. Murthi, J. Skoglund: iLBC – a linear predictive coder with robustness to packet losses, Proc. IEEE Workshop Speech Coding (2002) pp. 23–25
17.32 S.V. Andersen, A. Duric, H. Astrom, R. Hagen, W.B. Kleijn, J. Linden: Internet low bit rate codec (iLBC), IETF RFC **3951** (2004)
17.33 American National Standard: BV16 speech codec specification for voice over ip applications in cable telephony, ANSI/SCTE 24-21 2006 (2006)
17.34 B.S. Atal, S.L. Hanauer: Speech analysis and synthesis by linear prediction, J. Acoust. Soc. Am. **50**, 637–655 (1971)
17.35 E.F. Deprettere, P. Kroon: Regular excitation reduction for effective and efficient LP-coding of speech, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1985) pp. 965–968
17.36 W.B. Kleijn, D.J. Krasinski, R.H. Ketchum: Improved speech quality and efficient vector quantization in SELP, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1988) pp. 155–158
17.37 I.M. Trancoso, B.S. Atal: Efficient procedures for finding the optimum innovation in stochastic coders, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1986) pp. 2375–2378
17.38 G. Davidson, A. Gersho: Complexity reduction methods for vector excitaiton coding, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1986) pp. 3055–3058
17.39 D. Lin: Speech coding using pseudo-stochastic block codes, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1987) pp. 1354–1357
17.40 I.A. Gerson, M.A. Jasiuk: Vector sum excited linear prediction (VSELP), Proc. 1989 IEEE Workshop Speech Coding for Telecommunications (1989) pp. 66–68
17.41 P. Kroon, B.S. Atal: Pitch predictors with high temporal resolution, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1990) pp. 661–664
17.42 J.S. Marques, I.M. Trancoso, J.M. Tribolet, L.B. Almeida: Improved pitch prediction with fractional delays in CELP coding, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1990) pp. 665–668
17.43 J.H. Chen, M.S. Rauchwerk: An 8 kb/s low-delay CELP speech coder, Proc. IEEE Global Commun. Conf. (1991) pp. 1894–1898
17.44 S. Miki, K. Mano, H. Ohmuro, T. Moriya: Pitch synchronous innovation CELP (PSI-CELP), Proc. 1993 Eurospeech Conf. (1993) pp. 261–264
17.45 T. Moriya: Two-channel conjugate vector quantization for noisy channel speech coding, IEEE J. Sel. Areas Commun. **10**(5), 866–874 (1992)
17.46 ITU-T: G.729: coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP) (1996)
17.47 J.-P. Adoul, C. Lamblin: A comparison of some algebraic structures for CELP coding of speech, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1987) pp. 1953–1956
17.48 R.A. Salami: Binary code excited linear prediction (BCELP): a new approach to CELP coding of speech without the codebooks, IEEE Electron. Lett. **25**(6), 401–403 (1989)
17.49 A. Le Guyader, D. Massaloux, J.P. Petit: Robust and fast code-excited linear predictive coding of speech signals, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1989) pp. 120–122
17.50 J.-P. Adoul, C. Lamblin, A. Leguyader: Baseband speech coding at 2400 BPS using spherical vector quantization, IEEE Int. Conf. Commun. (1984) pp. 1.12.1–1.12.4
17.51 C. Laflamme, J.-P. Adoul, S. Morisette: A real time 4.8 Kbits/sec CELP on a single DSP chip (TMS320C25), IEEE Workshop Speech Coding for Telecommun. (1989) pp. 35–36

17.52 R.A. Salami, D.G. Appleby: A new approach to low bit rate speech coding with low complexity using binary pulse excitation (BPE), IEEE Workshop Speech Coding for Telecommun. (1989) pp. 63–65
17.53 C. Laflamme, J.-P. Adoul, R. Salami, S. Morisette, P. Mabilleau: 16 kbps wideband speech coding technique based on algebraic CELP, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1991) pp. 13–16
17.54 R. Salami, C. Laflamme, J.-P. Adoul, D. Massaloux: A toll quality 8 kb/s speech codec for the personal communications system (PCS), IEEE Trans. Vehicular Technol. **43**(3), 808–816 (1994)
17.55 K. Järvinen, J. Vaino, P. Kapanen, T. Honkanen, P. Haavisto, R. Salami, C. Laflamme, J.-P. Adoul: GSM enhanced full rate speech codec, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1997) pp. 771–774
17.56 ETSI EN 300 726 V8.0.1: Digital cellular telecommunications systems (Phase 2+); enhanced full rate (EFR) speech transcoding; (GSM 06.60 version 8.0.1 Release 1999) (2000)
17.57 3GPP TS 26.190 V6.1.1: 3rd generation partnership project; technical specification group services and system aspects; speech codec speech processing functions; adaptive multi-rate – wideband (AMR-WB) speech codec; transcoding functions (release 6) (2005)
17.58 R. Salami, C. Laflamme, J.P. Adoul, A. Kataoka, S. Hayashi, T. Moriya, C. Lamblin, D. Massaloux, S. Proust, P. Kroon, Y. Shoham: Design and description of CS-ACELP: a toll quality 8 kb/s speech coder, IEEE Trans. Speech Audio Process. **6**(2), 116–130 (1998)
17.59 ITU-T: G.729 Annex A: reduced complexity 8 kbit/s CS-ACELP speech codec (1996)
17.60 R. Salami, C. Laflamme, B. Bessette, J.P. Adoul: ITU-T G.729 Annex A: reduced complexity 8 kb/s CS-ACELP Codec for simultaneous vocie and data, IEEE Commun. Mag. **35**, 56–63 (1997)
17.61 ITU-T: G.722.2: wideband coding of speech at around 16 kbit/s using adaptive multi-rate wideband (AMR-WB) (2002)
17.62 3GPP TS 26.090 V6.0.0: 3rd generation partnership project; technical specification group services and system aspects; mandatory speech codec speech processing functions; adaptive multi-rate (AMR) speech codec; transcoding functions (release 6) (2004)
17.63 ANSI/TIA/EIA-136-410-99: TDMA cellular PCS – radio interface enhanced full-rate voice codec (ANSI/TIA/EIA-136-410-99) (R2003) (1999)
17.64 3GPP2 C.S0030-0 V1.0: Selectable mode vocoder service option for wideband spread spectrum communication systems, (June 15 2001)
17.65 ISO/IEC 14496-3 FCD, ISO/JTC 1/SC 29 N2203CELP: Information technology – coding of audiovisual objects, Part 3: audio, subpart 3: CELP, (May 13 1998)
17.66 A. Kataoka, T. Moriya, S. Hayashi: An 8-kbit/s speech coder based on conjugate structure CELP, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1993) pp. II–592–II–595
17.67 T. Moriya, H. Suda: An 8 kbit/s transform coder for noisy channels, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1989) pp. 196–199
17.68 A. Kataoka, T. Moriya, S. Hayashi: An 8-kb/s conjugate structure CELP (CS-CELP) speech coder, IEEE Trans. Speech Audio Process. **4**(6), 401–411 (1996)
17.69 W.B. Kleijn, P. Kroon, L. Cellario, D. Sereno: A 5.85 kb/s CELP algorithm for cellular applications, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1993) pp. II–596–II–599
17.70 W.B. Kleijn, R.P. Ramachandran, P. Kroon: Interpolation of the pitch-predictor parameters in analysis-by-synthesis speech coders, IEEE Trans. Speech Audio Process. **2**(1), 42–53 (1994)
17.71 Y. Gao, A. Benyassine, J. Thyssen, H. Su, E. Shlomot: A speech coding paradigm, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2001) pp. 689–692
17.72 J.-H. Chen: Novel codec structures for noise feedback coding of speech, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2006) pp. I.681–I.684
17.73 J.D. Makhoul, M. Berouti: Adaptive noise spectral shaping and entropy coding in predictive coding of speech, IEEE Trans. Acoust. Speech Signal Process. **27**, 63–73 (1979)
17.74 J. Thyssen, J.-H. Chen: Efficient VQ techniques and general noise shaping for noise feedback coding, Proc. Interspeech 2006 ICSLP (2006) pp. 221–224
17.75 PacketCable 2.0 codec and media specification, PKT-SP-CODEC-MEDIA-I02-061013 (2006)
17.76 ITU-T: G.729.1: G.729 based embedded variable bit-rate coder: An 8–32 kbit/s scalable wideband coder bitstrean interoperable with G.729 (2006)
17.77 ITU-T: G.726: 40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM) (1990)
17.78 ITU-T: G.722: 7 kHz audio-coding within 64 kbit/s (1988)