Basic Audio Compression Techniques

13

Compression of audio information is somewhat special in multimedia systems. Some of the techniques used are familiar, while others are new. In this chapter, we take a look at basic audio compression techniques applied to speech compression, setting out a general introduction to a large topic with a long history. More extensive information can be found in the References section at the end of the chapter.

In Chap. 14, we consider the set of tools developed for general audio compression under the aegis of the Motion Picture Experts Group (MPEG). Since this is generally of high interest to readers focusing on multimedia, we treat that subject in greater detail.

To begin with, let us recall some of the issues covered in Chap. 6 on digital audio in multimedia, such as the μ-law for companding audio signals. This is usually combined with a simple technique that exploits the temporal redundancy present in audio signals. We saw in Chap. 10, on video compression, that differences in signals between the present and a past time could very effectively reduce the size of signal values and, importantly, concentrate the histogram of pixel values (differences, now) into a much smaller range. The result of reducing the variance of values is that the entropy is greatly reduced, and subsequent Huffman coding can produce a greatly compressed bitstream.

The same applies here. Recall from Chap. 6 that quantized sampled output is called Pulse Code Modulation, or PCM. The differences version is called DPCM, and the adaptive version is called ADPCM. Variants that take into account speech properties follow from these.

In this chapter, we look at ADPCM, Vocoders, and more general Speech Compression: LPC, CELP, MBE, and MELP.

Z.-N. Li et al., *Fundamentals of Multimedia*,
Texts in Computer Science, DOI: 10.1007/978-3-319-05290-8_13,

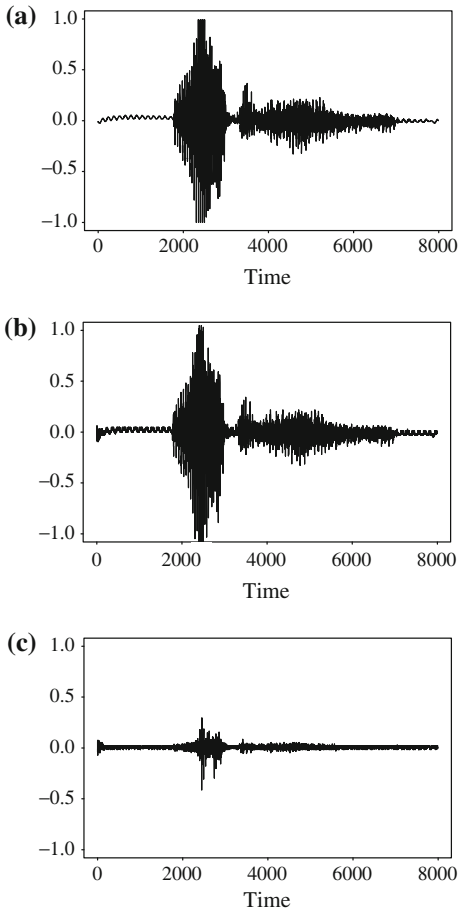

Fig. 13.1 Waveform of the word "audio:" **a** speech sample, linear PCM at 8 kHz and 16 bits per sample; **b** speech sample, restored from G.721-compressed audio at 4 bits per sample; **c** difference signal between (**a**) and (**b**)

13.1 ADPCM in Speech Coding

13.1.1 ADPCM

ADPCM forms the heart of the ITU's speech compression standards G.721, G.723, G.726, G.727, G.728, and G.729. The differences among these standards involve the bitrate and some details of the algorithm. The default input is μ-law-coded PCM 16-bit samples. Speech performance for ADPCM is such that the perceived quality of speech at 32 kbps (kilobits per second) is only slightly poorer than with the standard 64 kbps PCM transmission and is better than DPCM.

Figure 13.1 shows a 1 s speech sample of a voice speaking the word "audio." In Fig. 13.1a, the audio signal is stored as linear PCM (as opposed to the default μ-law PCM) recorded at 8,000 samples per second, with 16 bits per sample. After compression with ADPCM using ITU standard G.721, the signal appears as in Fig. 13.1b.

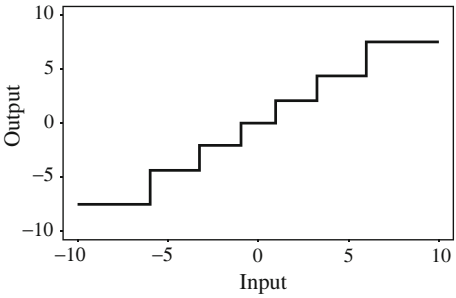

Fig. 13.2 G.726 quantizer

Figure 13.1c shows the difference between the actual and reconstructed, compressed signals. Although differences are apparent electronically between the two, the compressed and original signals are *perceptually* very similar.

13.2 G.726 ADPCM, G.727-9

ITU G.726 provides another version of G.711, including companding, at a lower bitrate. G.726 can encode 13- or 14-bit PCM samples or 8-bit μ-law or *A*-law encoded data into 2, 3, 4, or 5-bit codewords. It can be used in speech transmission over digital networks.

The G.726 standard works by adapting a *fixed* quantizer in a simple way. The different sizes of codewords used amount to bitrates of 16, 24, 32, or 40 kbps. The standard defines a multiplier constant α that will change for every difference value e_n, depending on the current scale of signals. Define a scaled difference signal f_n as follows:

$$\begin{aligned} e_n &= s_n - \hat{s}_n \\ f_n &= e_n/\alpha \end{aligned} \tag{13.1}$$

where $\hat{s}_n$ is the predicted signal value. f_n is then fed into the quantizer for quantization. The quantizer is displayed in Fig. 13.2. Here, the input value is defined as a ratio of a difference with the factor α.

By changing the value of α, the quantizer can adapt to change in the range of the difference signal. The quantizer is a nonuniform midtread quantizer, so it includes the value zero. The quantizer is *backward adaptive*.

A backward-adaptive quantizer works in principle by noticing if too many values are quantized to values far from zero (which would happen if the quantizer step size in f were too small) or if too many values fell close to zero too much of the time (which would happen if the quantizer step size were too large).

In fact, an algorithm due to Jayant [1] allows us to adapt a backward quantizer step size after receiving just one output! The Jayant quantizer simply expands the step size if the quantized input is in the outer levels of the quantizer and reduces the step size if the input is near zero.

Suppose we have a uniform quantizer, so that every range to which we compare input values is of size Δ. For example, for a 3-bit quantizer, there are $k = 0 \ldots 7$ levels. For 3-bit G.726, only seven levels are used, grouped around zero.

The Jayant quantizer assigns *multiplier values* M_k to each level, with values smaller than one for levels near zero and values larger than one for outer levels. The multiplier multiplies the step size for the next signal value. That way, outer values enlarge the step size and are likely to bring the next quantized value back to the middle of the available levels. Quantized values near the middle reduce the step size and are likely to bring the next quantized value closer to the outer levels.

So, for signal f_n, the quantizer step size Δ is changed according to the quantized value k, for the previous signal value f_{n-1}, by the simple formula

$$\Delta \leftarrow M_k \Delta. \tag{13.2}$$

Since the *quantized* version of the signal is driving the change, this is indeed a backward-adaptive quantizer.

In G.726, how α is allowed to change depends on whether the audio signal is actually speech or is likely data that is simply using a voice band. In the former case, sample-to-sample differences can fluctuate a great deal, whereas in the latter case of data transmission, this is less true. To adjust to either situation, the factor α is adjusted using a formula with two pieces.

G.726 works as a backward-adaptive Jayant quantizer by using fixed quantizer steps based on the logarithm of the input difference signal, e_n divided by α. The divisor α is written in terms of its logarithm:

$$\beta \equiv \log_2 \alpha. \tag{13.3}$$

Since we wish to distinguish between situations when difference values are usually small, and when they are large, α is divided into a so-called *locked* part, α_L, and an *unlocked* part, α_U. The idea is that the locked part is a scale factor for small difference values and changes slowly, whereas the unlocked part adapts quickly to larger differences. These correspond to log quantities β_L and β_U.

The logarithm value is written as a sum of two pieces,

$$\beta = A\beta_U + (1 - A)\beta_L \tag{13.4}$$

where A changes so that it is about one for speech, and about zero for voice-band data. It is calculated based on the variance of the signal, keeping track of several past signal values.

The "unlocked" part adapts via the equation

$$\begin{aligned} \alpha_U &\leftarrow M_k \alpha_U \\ \beta_U &\leftarrow \log_2 M_k + \beta_U \end{aligned} \tag{13.5}$$

where M_k is a Jayant multiplier for the kth level. The locked part is slightly modified from the unlocked part, via

$$\beta_L \leftarrow (1 - B)\beta_L + B\beta_U \tag{13.6}$$

where B is a small number, say 2^{-6}.

The G.726 predictor is complicated: it uses a linear combination of six quantized differences and two reconstructed signal values from the previous six signal values f_n.

ITU standards G.728 and G.729 use Code Excited Linear Prediction (CELP), discussed in Sect. 13.3.5.

13.3 Vocoders

The coders (encoding/decoding algorithms) we have studied so far could have been applied to any signals, not just speech. *Vocoders* are specifically voice coders.

Vocoders are concerned with modeling speech, so that the salient features are captured in as few bits as possible. They use either a model of the speech waveform in time (*Linear Predictive Coding* (LPC) vocoding), or else break down the signal into frequency components and model these (channel vocoders and formant vocoders).

Incidentally, we likely all know that vocoder simulation of the voice is not wonderful yet—when the library calls you with your overdue notification, the automated voice is strangely lacking in zest.

13.3.1 Phase Insensitivity

Recall from Sect. 8.5 that we can break down a signal into its constituent frequencies by analyzing it using some variant of Fourier analysis. In principle, we can also reconstitute the signal from the frequency coefficients developed that way. But it turns out that a complete reconstituting of speech waveform is unnecessary, perceptually: all that is needed is for the amount of energy at any time to be about right, and the signal will sound about right.

"Phase" is a shift in the time argument, inside a function of time. Suppose we strike a piano key and generate a roughly sinusoidal sound $\cos(\omega t)$, with $\omega = 2\pi f$ where f is the frequency. If we wait sufficient time to generate a phase shift $\pi/2$ and then strike another key, with sound $\cos(2\omega t + \pi/2)$, we generate a waveform like the solid line in Fig. 13.3. This waveform is the sum $\cos(\omega t) + \cos(2\omega t + \pi/2)$.

If we did not wait before striking the second note (1/4 ms, in Fig. 13.3), our waveform would be $\cos(\omega t) + \cos(2\omega t)$. But perceptually, the two notes would sound the same, even though in actuality they would be shifted in phase.

Hence, if we can get the energy spectrum right—where we hear loudness and quiet—then we do not really have to worry about the exact waveform.

13.3.2 Channel Vocoder

Subband filtering is the process of applying a bank of band-pass filters to the analog signal, thus actually carrying out the frequency decomposition indicated in a Fourier

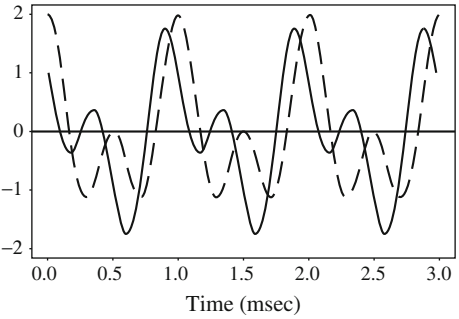

Fig. 13.3 The *solid line* shows the superposition of two cosines, with a phase shift. The *dashed line* shows the same with no phase shift. The wave is very different, yet the sound is the same, perceptually

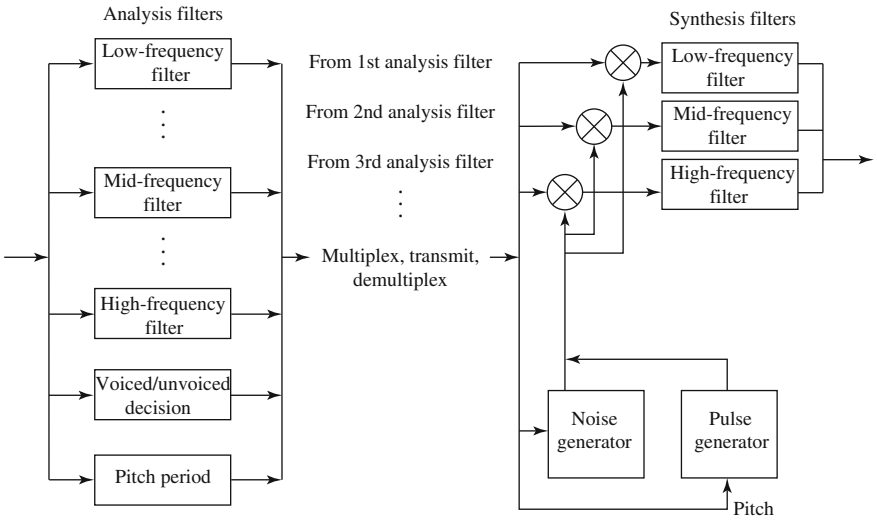

Fig. 13.4 Channel vocoder

analysis. *Subband coding* is the process of making use of the information derived from this filtering to achieve better compression.

For example, an older ITU recommendation, G.722, uses subband filtering of analog signals into just two bands: voice frequencies in 50 to 3.5 kHz and 3.5 to 7 kHz. Then the set of two signals is transmitted at 48 kbps for the low frequencies, where we can hear discrepancies well, and at only 16 kbps for the high frequencies.

Vocoders can operate at low bitrates, just 1–2 kbps. To do so, a *channel vocoder* first applies a filter bank to separate out the different frequency components, as in Fig. 13.4. However, as we saw above, only the energy is important, so first the waveform is "rectified" to its absolute value. The filter bank derives relative power levels for each frequency range. A subband coder would not rectify the signal and would use wider frequency bands.

A channel vocoder also analyzes the signal to determine the general pitch of the speech—low (bass), or high (tenor)—and also the *excitation* of the speech. Speech excitation is mainly concerned with whether a sound is *voiced* or *unvoiced*. A sound is unvoiced if its signal simply looks like noise: the sounds *s* and *f* are unvoiced. Sounds such as the vowels *a*, *e*, and *o* are voiced, and their waveform looks periodic. The *o* at the end of the word "audio" in Fig. 13.1 is fairly periodic. During a vowel sound, air is forced through the vocal cords in a stream of regular, short puffs, occurring at the rate of 75–150 pulses per second for men and 150–250 per second for women.

Consonants can be voiced or unvoiced. For the nasal sounds of the letters *m* and *n*, the vocal cords vibrate, and air is exhaled through the nose rather than the mouth. These consonants are therefore voiced. The sounds *b*, *d*, and *g*, in which the mouth starts closed but then opens to the following vowel over a transition lasting a few milliseconds, are also voiced. The energy of voiced consonants is greater than that of unvoiced consonants but less than that of vowel sounds. Examples of unvoiced consonants include the sounds *sh*, *th*, and *h* when used at the front of a word.

A channel vocoder applies a vocal-tract transfer model to generate a vector of excitation parameters that describe a model of the sound. The vocoder also guesses whether the sound is voiced or unvoiced and, for voiced sounds, estimates the period (i.e., the sound's pitch). Figure 13.4 shows that the decoder also applies a vocal-tract model.

Because voiced sounds can be approximated by sinusoids, a periodic pulse generator recreates voiced sounds. Since unvoiced sounds are noise-like, a pseudo-noise generator is applied, and all values are scaled by the energy estimates given by the band-pass filter set. A channel vocoder can achieve an intelligible but synthetic voice using 2,400 bps.

13.3.3 Formant Vocoder

It turns out that not all frequencies present in speech are equally represented. Instead, only certain frequencies show up strongly, and others are weak. This is a direct consequence of how speech sounds are formed, by resonance in only a few chambers of the mouth, throat, and nose. The important frequency peaks are called *formants* [2].

Figure 13.5 shows how this appears: only a few, usually just four or so, peaks of energy at certain frequencies are present. The peak locations however change in time, as speech continues. For example, two different vowel sounds would activate different sets of formants—this reflects the different vocal-tract configurations necessary to form each vowel. Usually, a small segment of speech is analyzed, say 10–40 ms, and formants are found. A *Formant Vocoder* works by encoding only the most important frequencies. Formant vocoders can produce reasonably intelligible speech at only 1,000 bps.

13.3.4 Linear Predictive Coding (LPC)

LPC vocoders extract salient features of speech directly from the waveform rather than transforming the signal to the frequency domain. LPC coding uses a time-

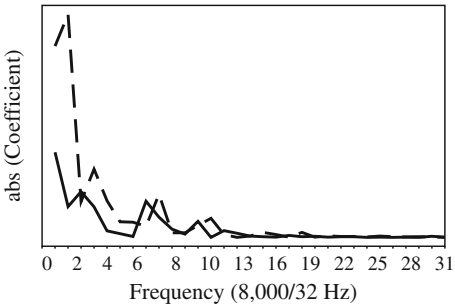

Fig. 13.5 Formants are the salient frequency components present in a sample of speech. Here, the *solid line* shows frequencies present in the first 40 ms of the speech sample in Fig. 6.16. The *dashed line* shows that while similar frequencies are still present 1 s later, they have shifted

varying model of vocal-tract sound generated from a given excitation. What is transmitted is a set of parameters modeling the shape and excitation of the vocal tract, not actual signals or differences.

Since what is sent is an analysis of the sound rather than sound itself, the bitrate using LPC can be small. This is like using a simple descriptor such as MIDI to generate music: we send just the description parameters and let the sound generator do its best to create appropriate music. The difference is that as well as pitch, duration, and loudness variables, here we also send vocal-tract excitation parameters.

After a block of digitized samples, called a *segment* or *frame*, is analyzed, the speech signal generated by the output vocal-tract model is calculated as a function of the current speech output plus a second term linear in previous model coefficients. This is how "linear" in the coder's name arises. The model is adaptive—the encoder side sends a new set of coefficients for each new segment.

The typical number of sets of previous coefficients used is $N = 10$ (the "model order" is 10), and such an LPC-10 [3] system typically uses a rate of 2.4 kbps. The model coefficients a_i act as predictor coefficients, multiplying previous speech output sample values.

LPC starts by deciding whether the current segment is voiced or unvoiced. For unvoiced speech, a wide-band noise generator is used to create sample values $f(n)$ that act as input to the vocal-tract simulator. For voiced speech, a pulse-train generator creates values $f(n)$. Model parameters a_i are calculated by using a least-squares set of equations that minimize the difference between the actual speech and the speech generated by the vocal-tract model, excited by the noise or pulse-train generators that capture speech parameters.

If the output values generated are denoted $s(n)$, then for input values $f(n)$, the output depends on p previous *output* sample values, via

$$s(n) = \sum_{i=1}^{p} a_i s(n-i) + Gf(n). \tag{13.7}$$

Here, G is known as the *gain* factor. Note that the coefficients a_i act as values in a linear predictor model. The pseudo-noise generator and pulse generator are as discussed above and depicted in Fig. 13.4 in regard to the channel vocoder.

The speech encoder works in a blockwise fashion. The input digital speech signal is analyzed in some small, fixed-length segments, called speech frames. For the LPC speech coder, the frame length is usually selected as 22.5 ms, which corresponds to 180 samples for 8 kHz sampled digital speech. The speech encoder analyzes the speech frames to obtain the parameters such as LP coefficients $a_i, i = 1 \ldots p$, gain G, pitch P, and voiced/unvoiced decision U/V.

To calculate LP coefficients, we can solve the following minimization problem for a_j:

$$\min E\{[s(n) - \sum_{j=1}^{p} a_j s(n-j)]^2\}. \tag{13.8}$$

By taking the derivative of a_i and setting it to zero, we get a set of p equations:

$$E\{[s(n) - \sum_{j=1}^{p} a_j s(n-j)]s(n-i)\} = 0, \qquad i = 1 \ldots p \tag{13.9}$$

Letting $\phi(i, j) = E\{s(n-i)s(n-j)\}$, we have

$$\begin{bmatrix} \phi(1,1) & \phi(1,2) & \cdots & \phi(1,p) \\ \phi(2,1) & \phi(2,2) & \cdots & \phi(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(p,1) & \phi(p,2) & \cdots & \phi(p,p) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} \phi(0,1) \\ \phi(0,2) \\ \vdots \\ \phi(0,p) \end{bmatrix} \tag{13.10}$$

The *autocorrelation method* is often used to calculate LP coefficients, where

$$\phi(i, j) = \sum_{n=p}^{N-1} s_w(n-i)s_w(n-j) / \sum_{n=p}^{N-1} s_w^2(n) \qquad i = 0 \ldots p,\ j = 1 \ldots p \tag{13.11}$$

$s_w(n) = s(n+m)w(n)$ is the windowed speech frame starting from time m. Since $\phi(i, j)$ is determined only by $|i-j|$, we define $\phi(i, j) = R(|i-j|)$. Since we also have $R(0) \geq 0$, the matrix $\{\phi(i, j)\}$ is positive symmetric, and thus a fast scheme to calculate the LP coefficients is as follows:

Procedure 13.1 (LPC Coefficients).

$E(0) = R(0), i = 1$

while $i \leq p$
  $k_i = [R(i) - \sum_{j=1}^{i-1} a_j^{i-1} R(i-j)]/E(i-1)$
  $a_i^i = k_i$
  for $j = 1$ to $i-1$
    $a_j^i = a_j^{i-1} - k_i a_{i-j}^{i-1}$
  $E(i) = (1 - k_i^2)E(i-1)$
  $i \leftarrow i + 1$
for $j = 1$ to p
  $a_j = a_j^p$

After getting the LP coefficients, gain G can be calculated as:

$$\begin{aligned} G &= E\{[s(n) - \sum_{j=1}^{p} a_j s(n-j)]^2\} \\ &= E\{[s(n) - \sum_{j=1}^{p} a_j s(n-j)]s(n)\} \\ &= \phi(0,0) - \sum_{j=1}^{p} a_j \phi(0,j) \end{aligned} \tag{13.12}$$

For the autocorrelation scheme, $G = R(0) - \sum_{j=1}^{p} a_j R(j)$. Order-10 LP analysis is found to be enough for speech coding applications.

The pitch P of the current speech frame can be extracted by the correlation method by finding the index of the peak of

$$\begin{aligned} v(i) &= \sum_{n=m}^{N-1+m} s(n)s(n-i) \Big/ [\sum_{n=m}^{N-1+m} s^2(n) \sum_{n=m}^{N-1+m} s^2(n-i)]^{1/2} \\ i &\in [P_{\min}, P_{\max}]. \end{aligned} \tag{13.13}$$

The searching range $[P_{\min}, P_{\max}]$ is often selected as $[12, 140]$ for 8 kHz sampling speech. Denote P as the peak lag. If $v(P)$ is less than some given threshold, the current frame is classified as an unvoiced frame and will be reconstructed in the receiving end by stimulating with a white-noise sequence. Otherwise, the frame is determined as voiced and stimulated with a periodic waveform at the reconstruction stage. In practical LPC speech coders, the pitch estimation and U/V decision procedure are usually based on a dynamic programming scheme, so as to correct the often occurring errors of pitch doubling or halving in the single frame scheme.

In LPC-10, each segment is 180 samples, or 22.5 ms at 8 kHz. The speech parameters transmitted are the coefficients a_k; G, the gain factor; a voiced/unvoiced flag (1 bit); and the pitch period if the speech is voiced.

13.3.5 Code Excited Linear Prediction (CELP)

CELP, *Code Excited Linear Prediction* (sometimes *Codebook Excited*), is a more complex family of coders that attempts to mitigate the lack of quality of the simple LPC model by using a more complex description of the excitation. An entire set (a codebook) of excitation vectors is matched to the actual speech, and the index of the best match is sent to the receiver. This complexity increases the bitrate to 4,800–9,600 bps, typically.

In CELP, since all speech segments make use of the same set of templates from the template codebook, the resulting speech is perceived as much more natural than the two-mode excitation scheme in the LPC-10 coder. The quality achieved is considered sufficient for audio conferencing.

In CELP coders two kinds of prediction, Long Time Prediction (LTP) and Short-Time Prediction (STP), are used to eliminate the redundancy in speech signals. STP is an analysis of *samples*—it attempts to predict the next sample from several previous ones. Here, redundancy is due to the fact that usually one sample will not change drastically from the next. LTP is based on the idea that in a *segment* of speech, or perhaps from segment to segment, especially for voiced sounds, a basic periodicity or pitch will cause a waveform that more or less repeats. We can reduce this redundancy by finding the pitch.

For concreteness, suppose we sample at 8,000 samples per second and use a 10 ms frame, containing 80 samples. Then we can roughly expect a pitch that corresponds to an approximately repeating pattern every 12–140 samples or so. (Notice that the pitch may actually be longer than the chosen frame size.)

STP is based on a short-time LPC analysis, discussed in the last section. It is "short-time" in that the prediction involves only a few samples, not a whole frame or several frames. STP is also based on minimizing the residue error over the whole speech frame, but it captures the correlation over just a short range of samples (10 for order-10 LPC).

After STP, we can subtract signal minus prediction to arrive at a differential coding situation. However, even in a set of errors $e(n)$, the basic pitch of the sequence may still remain. This is estimated by means of LTP. That is, LTP is used to further eliminate the periodic redundancy inherent in the voiced speech signals. Essentially, STP captures the formant structure of the short-term speech spectrum, while LTP recovers the long-term correlation in the speech signal that represents the periodicity in speech.

Thus there are always two stages—and the order is in fact usually STP followed by LTP, since we always start off assuming zero error and then remove the pitch component. (If we use a closed-loop scheme, STP usually is done first). LTP proceeds using whole frames—or, more often, subframes equal to one quarter of a frame. Figure 13.6 shows these two stages.

LTP is often implemented as *adaptive codebook searching*. The "codeword" in the adaptive codebook is a shifted speech residue segment indexed by the lag τ corresponding to the current speech frame or subframe. The idea is to look in a codebook of waveforms to find one that matches the current subframe. We generally look in the codebook using a *normalized* subframe of speech, so as well as a speech segment match, we also obtain a scaling value (the *gain*). The gain corresponding to the codeword is denoted as g_0.

There are two types of codeword searching: *open-loop* and *closed-loop*. Open-loop adaptive codebook searching tries to minimize the long-term prediction error but not the perceptual weighted reconstructed speech error,

$$E(\tau) = \sum_{n=0}^{L-1} [s(n) - g_0 s(n-\tau)]^2 \tag{13.14}$$

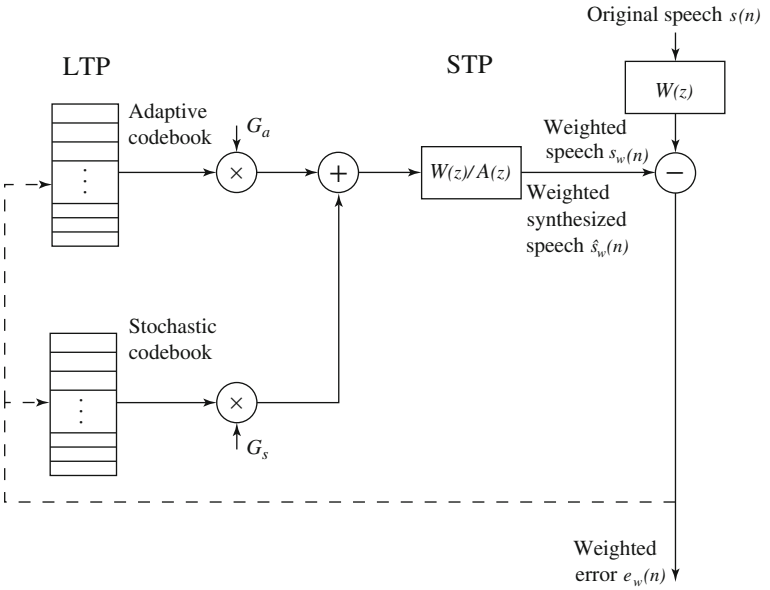

Fig. 13.6 CELP analysis model with adaptive and stochastic codebooks

By setting the partial derivative of g_0 to zero, $\partial E(\tau)/\partial g_0 = 0$, we get

$$g_0 = \frac{\sum_{n=0}^{L-1} s(n)s(n-\tau)}{\sum_{n=0}^{L-1} s^2(n-\tau)} \tag{13.15}$$

and hence a minimum summed-error value

$$E_{\min}(\tau) = \sum_{n=0}^{L-1} s^2(n) - \frac{[\sum_{n=0}^{L-1} s(n)s(n-\tau)]^2}{\sum_{n=0}^{L-1} s^2(n-\tau)}. \tag{13.16}$$

Notice that the sample $s(n-\tau)$ could be in the previous frame.

Now, to obtain the optimum adaptive codebook index τ, we can carry out a search exclusively in a small range determined by the pitch period. More often, CELP coders use a closed-loop search. Rather than simply considering sum of squares, speech is reconstructed, with perceptual error minimized via an adaptive codebook search. So in a closed-loop, adaptive codebook search, the best candidate in the adaptive codebook is selected to minimize the distortion of locally reconstructed speech. Parameters are found by minimizing a measure (usually the mean square) of the difference between the original and the reconstructed speech. Since this means that we are simultaneously incorporating synthesis as well as analysis of the speech segment, this method is also called *analysis-by-synthesis*, or *A-B-S*.

The residue signal after STP based on LPC analysis and LTP based on adaptive codeword searching is like white noise and is encoded by codeword matching in the stochastic (random or probabilistic) codebook. This kind of sequential optimization

of the adaptive codeword and stochastic codeword methods is used because jointly optimizing the adaptive and stochastic codewords is often too complex to meet real-time demands.

The decoding direction is just the reverse of the above process and works by combining the contribution from the two types of excitations.

DOD 4.8 Kbps CELP (FS1016)*

DOD 4.8 kbps CELP [4] is an early CELP coder adopted as a U.S. federal standard to update the 2.4 kbps LPC-10e (FS1015) vocoder. This vocoder is now a basic benchmark to test other low-bitrate vocoders. FS1016 uses an 8 kHz sampling rate and 30 ms frame size. Each frame is further split into four 7.5 ms subframes. In FS1016, STP is based on an open-loop order-10 LPC analysis.

To improve coding efficiency, a fairly sophisticated type of transform coding is carried out. Then, quantization and compression are done in terms of the transform coefficients.

First, in this field it is common to use the *z-transform*. Here, z is a complex number and represents a kind of complex "frequency." If $z = e^{-2\pi i/N}$, then the discrete z-transform reduces to a discrete Fourier transform. The z-transform makes Fourier transforms look like polynomials. Now we can write the error in a prediction equation

$$e(n) = s(n) - \sum_{i=1}^{p} a_i s(n-i) \tag{13.17}$$

in the z domain as

$$E(z) = A(z)S(z) \tag{13.18}$$

where $E(z)$ is the z-transform of the error and $S(z)$ is the transform of the signal. The term $A(z)$ is the transfer function in the z domain, and equals

$$A(z) = 1 - \sum_{i=1}^{p} a_i z^{-i} \tag{13.19}$$

with the same coefficients a_i as appear in Eq. (13.7). How speech is reconstructed, then, is via

$$S(z) = E(z)/A(z) \tag{13.20}$$

with the estimated error. For this reason, $A(z)$ is usually stated in terms of $1/A(z)$.

The idea of going to the z-transform domain is to convert the LP coefficients to *Line Spectrum Pair* (*LSP*) coefficients, which are given in this domain. The reason is that the LSP space has several good properties with respect to quantization. LSP representation has become standard and has been applied to nearly all the recent LPC-based speech coders, such as G.723.1, G.729, and MELP. To get LSP coefficients, we construct two polynomials

$$\begin{aligned} P(z) &= A(z) + z^{-(p+1)} A(z^{-1}) \\ Q(z) &= A(z) - z^{-(p+1)} A(z^{-1}) \end{aligned} \tag{13.21}$$

where p is the order of the LPC analysis and $A(z)$ is the transform function of the LP filter, with z the transform domain variable. The z-transform is just like the Fourier transform but with a complex "frequency."

The roots of these two polynomials are spaced around the unit circle in the z plane and have mirror symmetry with respect to the x axis. Assume p is even and denote the phase angles of the roots of $P(z)$ and $Q(z)$ above the x axis as $\theta_1 < \theta_2 < \ldots < \theta_{p/2}$ and $\varphi_1 < \varphi_2 < \ldots < \varphi_{p/2}$, respectively. Then the vector $\{\cos(\theta_1), \cos(\varphi_1), \cos(\theta_2), \cos(\varphi_1) \ldots \cos(\theta_{p/2}), \cos(\varphi_{p/2})\}$ is the LSP coefficient vector, and vector $\{\theta_1, \varphi_1, \theta_2, \varphi_1 \ldots, \theta_{p/2}, \varphi_{p/2}\}$ is usually called *Line Spectrum Frequency*, or *LSF*. Based on the relationship $A(z) = [P(z) + Q(z)]/2$, we can reconstruct the LP coefficients at the decoder end from the LSP or LSF coefficients.

Adaptive codebook searching in FS1016 is via a closed-loop search based on perceptually weighted errors. As opposed to considering just the mean squared error, here errors are weighted so as to take human perception into account. In terms of the z-transform, it is found that the following multiplier does a good job:

$$W(z) = \frac{A(z)}{A(z/\gamma)} = \frac{1 - \sum_{i=1}^{p} a_i z^{-i}}{1 - \sum_{i=1}^{p} a_i \gamma^i z^{-i}} \quad 0 < \gamma < 1 \tag{13.22}$$

with a constant parameter γ.

The adaptive codebook has 256 codewords for 128 integer delays and 128 non-integer delays (with half-sample interval, for better resolution), the former ranging from 20 to 147. To reduce searching complexity, even subframes are searched in an interval relative to the previous odd subframe, and the difference is coded with 6 bits. The gain is nonuniformly scalar coded between -1 and 2 with 5 bits.

Stochastic codebook search is applied for each of the four subframes. The stochastic codebook of FS1016 is generated by clipping a unit variance Gaussian distribution random sequence to within a threshold of absolute value 1.2 and quantizing to three values -1, 0, and 1. The stochastic codebook has 512 codewords. The codewords are overlapped, and each is shifted by 2 with respect to the previous codeword. This kind of stochastic design is called an *Algebraic Codebook*. It has many variations and is widely applied in recent CELP coders.

Denoting the excitation vector as $v^{(i)}$, the periodic component obtained in the first stage is $v^{(0)}$. $v^{(1)}$ is the stochastic component search result in the second stage. In closed-loop searching, the reconstructed speech can be represented as:

$$\widehat{s} = \widehat{s_0} + (u + v^{(i)})H \tag{13.23}$$

where u is equal to zero at the first stage and $v^{(0)}$ at the second stage, and $\widehat{s_0}$ is the zero response of the LPC reconstructing filter. Matrix H is the truncated LPC reconstructing filter unit impulse-response matrix

$$H = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{L-1} \\ 0 & h_0 & h_1 & \cdots & h_{L-2} \\ 0 & 0 & h_0 & \cdots & h_{L-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & h_0 \end{bmatrix} \tag{13.24}$$

where L is the length of the subframe (this simply represents a convolution). Similarly, defining W as the unit response matrix of the perceptual weighting filter, the perceptually weighted error of reconstructed speech is:

$$e = (s - \widehat{s})W = e_0 - v^{(i)}HW \tag{13.25}$$

where $e_0 = (s - \widehat{s_0})W - uHW$. The codebook searching process is to find a codeword $y^{(i)}$ in the codebook and corresponding $a^{(i)}$ such that $v^{(i)} = a^{(i)}y^{(i)}$ and ee^T is minimized. To make the problem tractable, adaptive and stochastic codebooks are searched sequentially. Denoting a quantized version by $\tilde{a}^{(i)} = Q[\hat{a}^{(i)}]$, then the criterion of codeword searching in the adaptive codebook or stochastic codebook is to minimize ee^T over all $y^{(i)}$ in terms of an expression in $\tilde{a}^{(i)}$, e_0, and $y^{(i)}$.

The decoder of the CELP codec is a reverse process of the encoder. Because of the unsymmetrical complexity property of vector quantization, the complexity in the decoder side is usually much lower.

G.723.1*

G723.1 [5] is an ITU standard aimed at multimedia communication. It has been incorporated into H.324 for audio encoding in videoconference applications. G.723.1 is a dual-rate CELP-type speech coder that can work at bitrates of 5.3 and 6.3 kbps.

G.723.1 uses many techniques similar to FS1016, discussed in the last section. The input speech is again 8 kHz, sampled in 16-bit linear PCM format. The speech frame size is also 30 msec and is further divided into four equal-sized subframes. Order-10 LPC coefficients are estimated in each subframe. LP coefficients are further converted to LSP vectors and quantized by predictive splitting VQ. LP coefficients are also used to form the perceptually weighted filter.

G.723.1 first uses an open-loop pitch estimator to get a coarse pitch estimation in a time interval of every two subframes. Closed-loop pitch searching is done in every speech subframe by searching the data in a range of the open-loop pitch. After LP filtering and removing the harmonic components by LTP, the stochastic residue is quantized by *Multipulse Maximum Likelihood Quantization* (*MP-MLQ*) for the 5.3 kbps coder or *Algebraic-Code-Excited Linear Prediction* (*ACELP*) for the 6.3 kbps coder, which has a slightly higher speech quality. These two modes can be switched at any boundary of the 30 ms speech frames.

In MP-MLQ, the contribution of the stochastic component is represented as a sequence of pulses

$$v(n) = \sum_{i=1}^{M} g_i \delta(n - m_i) \tag{13.26}$$

where M is the number of pulses and g_i is gain of the pulse at position m_i. The closed-loop search is done by minimizing

$$e(n) = r(n) - \sum_{i=1}^{M} g_i h(n - m_i) \tag{13.27}$$

where $r(n)$ is the speech component after perceptual weighting and eliminating the zero-response component and periodic component contributions. Based on methods similar to those presented in the last section, we can sequentially optimize the gain and position for each pulse. Say we first assume there is only one pulse and find the best gain and position. After removing the contribution from this pulse, we can get the next optimal pulse based on the same method. This process is done recursively until we get all M pulses.

The stochastic codebook structure for the ACELP model is different from FS1016. The following table shows the ACELP excitation codebook:

Sign	*Positions*
±1	0, 8, 16, 24, 32, 40, 48, 56
±1	2, 10, 18, 26, 34, 42, 50, 58
±1	4, 12, 20, 28, 36, 44, 52, 60
±1	6, 14, 22, 30, 38, 46, 54, 62

(13.28)

There are only four pulses. Each can be in eight positions, coded by three bits each. Also, the sign of the pulse takes one bit, and another bit is to shift all possible positions to odd. Thus, the index of a codeword has 17 bits. Because of the special structure of the algebraic codebook, a fast algorithm exists for efficient codeword searching.

Besides the CELP coder we discussed above, there are many other CELP-type codecs, developed mainly for wireless communication systems. The basic concepts of these coders are similar, except for different implementation details on parameter analysis and codebook structuring.

Some examples include the 12.2 kbps GSM *Enhanced Full Rate* (*EFR*) algebraic CELP codec [6] and IS-641EFR [7], designed for the North American digital cellular IS-136 TDMA system. G.728 [8] is a low-delay CELP speech coder. G.729 [9] is another CELP based ITU standard aimed at toll-quality speech communications.

G.729 is a *Conjugate-Structure Algebraic-Code-Excited-Linear-Prediction* (*CS-ACELP*) codec. G.729 uses a 10 ms speech analysis frame and thus has lower delay than G.723.1, which uses a 30 ms speech frame. G.729 also has some inherent protection schemes to deal with packet loss in applications such as VoIP.

13.3.6 Hybrid Excitation Vocoders*

Hybrid Excitation Vocoders are another large class of speech coders. They are different from CELP, in which the excitation is represented as the contributions of the adaptive and stochastic codewords. Instead, hybrid excitation coders use model-based methods to introduce multimodel excitation.

Multiband Excitation (MBE)

The *Multiband Excitation* (*MBE*) [10] vocoder was developed by MIT's Lincoln Laboratory. The 4.15 kbps IMBE codec [11] has become the standard for IMMSAT.

MBE is also a blockwise codec, in which a speech analysis is done in a speech frame unit of about 20–30 ms. In the analysis part of the MBE coder, a spectrum analysis such as FFT is first applied for the windowed speech in the current frame. The short-time speech spectrum is further divided into different spectrum bands. The bandwidth is usually an integer times the basic frequency that equals the inverse of the pitch. Each band is described as "voiced" or "unvoiced."

The parameters of the MBE coder thus include the spectrum envelope, pitch, unvoiced/voiced (U/V) decisions for different bands. Based on different bitrate demands, the phase of the spectrum can be parameterized or discarded. In the speech decoding process, voiced bands and unvoiced bands are synthesized by different schemes and combined to generate the final output.

MBE utilizes the analysis-by-synthesis scheme in parameter estimation. Parameters such as basic frequency, spectrum envelope, and subband U/V decisions are all done via closed-loop searching. The criteria of the closed-loop optimization are based on minimizing the perceptually weighted reconstructed speech error, which can be represented in the frequency domain as:

$$\varepsilon = \frac{1}{2\pi}\int_{-\pi}^{+\pi} G(\omega)|S_w(\omega) - S_{wr}(\omega)|d\omega \tag{13.29}$$

where $S_w(\omega)$ and $S_{wr}(\omega)$ are the original speech short-time spectrum and reconstructed speech short-time spectrum, and $G(\omega)$ is the spectrum of the perceptual weighting filter.

Similar to the closed-loop searching scheme in CELP, a sequential optimization method is used to make the problem tractable. In the first step, all bands are assumed voiced bands, and the spectrum envelope and basic frequency are estimated. Rewriting the spectrum error with the all-voiced assumption, we have

$$\check{\varepsilon} = \sum_{m=-M}^{M}[\frac{1}{2\pi}\int_{\alpha_m}^{\beta_m} G(\omega)|S_w(\omega) - A_m E_{wr}(\omega)|^2 d\omega] \tag{13.30}$$

in which M is band number in $[0, \pi]$, A_m is the spectrum envelope of band m, $E_{wr}(\omega)$ is the short-time window spectrum, and $\alpha_m = (m - \frac{1}{2})\omega_0$, $\beta_m = (m + \frac{1}{2})\omega_0$. Setting $\partial\check{\varepsilon}/\partial A_m = 0$, we get

$$A_m = \frac{\int_{\alpha_m}^{\beta_m} G(\omega)S_w(\omega)E_{wr}^*(\omega)d\omega}{\int_{\alpha_m}^{\beta_m} G(\omega)|E_{wr}(\omega)|^2 d\omega} \tag{13.31}$$

The basic frequency is obtained at the same time by searching over a frequency interval to minimize $\check{\varepsilon}$. Based on the estimated spectrum envelope, an adaptive thresholding scheme tests the matching degree for each band. We label a band as voiced if there is a good matching; otherwise, we declare the band as unvoiced and re-estimate the envelope for the unvoiced band as:

$$A_m = \frac{\int_{\alpha_m}^{\beta_m} G(\omega)|S_w(\omega)|d\omega}{\int_{\alpha_m}^{\beta_m} G(\omega)d\omega}. \tag{13.32}$$

The decoder uses separate methods to synthesize unvoiced and voiced speech, based on the unvoiced and voiced bands. The two types of reconstructed components are then combined to generate synthesized speech. The final step is overlapping the sum of the synthesized speech in each frame to get the final output.

Multiband Excitation Linear Predictive (MELP)

The Multiband Excitation Linear Predictive (MELP) speech codec is a new U.S. federal standard to replace the old LPC-10 (FS1015) standard, with the application focus on low-bitrate safety communications. At 2.4 kbps, MELP [12] has comparable speech quality to the 4.8 kbps DOD-CELP (FS1016) and good robustness in a noisy environment.

MELP is also based on LPC analysis. Different from the hard-decision voiced/unvoiced model adopted in LPC-10, MELP uses a multiband soft-decision model for the excitation signal. The LP residue is band-passed, and a voicing strength parameter is estimated for each band. The decoder can reconstruct the excitation signal by combining the periodic pulses and white noises, based on the voicing strength in each band. Speech can be then reconstructed by passing the excitation through the LPC synthesis filter.

Different from MBE, MELP divides the excitation into five fixed bands of 0–500, 500–1000, 1000–2000, 2000–3000, and 3000–4000 Hz. It estimates a voice degree parameter in each band based on the normalized correlation function of the speech signal and the smoothed, rectified signal in the non-DC band. Let $s_k(n)$ denote the speech signal in band k, and $u_k(n)$ denote the DC-removed smoothed rectified signal of $s_k(n)$. The correlation function is defined as

$$R_x(P) = \frac{\sum_{n=0}^{N-1} x(n)x(n+P)}{[\sum_{n=0}^{N-1} x^2(n) \sum_{n=0}^{N-1} x^2(n+P)]^{1/2}} \tag{13.33}$$

where P is the pitch of the current frame, and N is the frame length. Then the voicing strength for band k is defined as $\max(R_{s_k}(P), R_{u_k}(P))$.

To further remove the buzziness of traditional LPC-10 speech coders for the voiced speech segment, MELP adopts a jittery voiced state to simulate the marginal voiced speech segments. The jittery state is indicated by an aperiodic flag. If the aperiodic flag is set in the analysis end, the receiver adds a random shifting component to the periodic pulse excitation. The shifting can be as big as $P/4$. The jittery state is determined by the peakiness of the full-wave rectified LP residue $e(n)$,

$$\text{peakiness} = \frac{[\frac{1}{N}\sum_{n=0}^{N-1} e(n)^2]^{1/2}}{\frac{1}{N}\sum_{n=0}^{N-1} |e(n)|}. \tag{13.34}$$

If peakiness is greater than some threshold, the speech frame is determined as jittered.

To better reconstruct the short-time spectrum of the speech signal, the spectrum of the residue signal is not assumed to be flat, as it is in the LPC-10 speech coder. After normalizing the LP residue signal, MELP preserves the magnitudes corresponding to the first $\min(10, P/4)$ basic frequency harmonics. Basic frequency is the inverse

of the pitch period. The higher harmonics are discarded and assumed to be unity spectrum.

The 10-d magnitude vector is quantized by 8-bit vector quantization, using a perceptual weighted distance measure. Similar to most modern LPC quantization schemes, MELP also converts LPC parameters to LSF and uses four-stage vector quantization. The bits allocated for the four stages are 7, 6, 6, and 6, respectively. Apart from integral pitch estimation similar to LPC-10, MELP applies a fractional pitch refinement procedure to improve the accuracy of pitch estimation.

In the speech reconstruction process, MELP does not use a periodic pulse to represent the periodic excitation signal but uses a dispersed waveform. To disperse the pulses, a *finite impulse-response* (*FIR*) filter is applied to the pulses. MELP also applies a perceptual weighting filter postfilter to the reconstructed speech so as to suppress the quantization noise and improve the subject's speech quality.

13.4 Exercises

1. In Sect. 13.3.1 we discuss phase insensitivity. Explain the meaning of the term "phase" in regard to individual frequency components in a composite signal.
2. Input a speech segment, using C or MATLAB, and verify that formants indeed exist—that any speech segment has only a few important frequencies. Also, verify that formants change as the interval of speech being examined changes. A simple approach to coding a frequency analyzer is to reuse the DCT coding ideas we have previously considered in Sect. 8.5. In one dimension, the DCT transform reads

$$F(u) = \sqrt{\frac{2}{N}} C(u) \sum_{i=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N} f(i) \tag{13.35}$$

where $i, u = 0, 1, \ldots, N-1$, and the constants $C(u)$ are given by:

$$C(u) = \begin{cases} \frac{\sqrt{2}}{2} & \text{if } u = 0 \\ 1 & \text{otherwise} \end{cases} \tag{13.36}$$

If we use the speech sample in Fig. 6.16, then taking the one-dimensional DCT of the first, or last, 40 ms (i.e., 32 samples), we arrive at the absolute frequency components as in Fig. 13.5.
3. Write code to read a WAV file. You will need the following set of definitions: a WAV file begins with a 44-byte header, in unsigned byte format. Some important parameter information is coded as follows:
4. Write a program to add fade in and fade-out effects to sound clips (in WAV format). Specifications for the fades are as follows: The algorithm assumes a linear envelope; the fade-in duration is from 0 to 20 % of the data samples; the fade-out duration is from 80 to 100 % of the data samples. If you like, you can make your code able to handle both mono and stereo WAV files. If necessary, impose a limit on the size of the input file, say 16 mb.

Byte[22 ... 23] Number of channels
Byte[24 ... 27] Sampling rate
Byte[34 ... 35] Sampling bits
Byte[40 ... 43] Data length

5. In the text, we study an adaptive quantization scheme for ADPCM. We can also use an adaptive prediction scheme. We consider the case of one tap prediction, $\hat{s}(n) = a \cdot s(n-1)$. Show how to estimate the parameter a in an open-loop method. Estimate the SNR gain you can get, compared to the direct PCM method based on a uniform quantization scheme.
6. Linear prediction analysis can be used to estimate the shape of the envelope of the short-time spectrum. Given ten LP coefficients $a_1, \ldots, a_{10}$, how do we get the formant position and bandwidth?
7. Download and implement a CELP coder (see the textbook web site). Try out this speech coder on your own recorded sounds.
8. In quantizing LSP vectors in G.723.1, splitting vector quantization is used: if the dimensionality of LSP is 10, we can split the vector into three subvectors of length 3, 3, and 4 each and use vector quantization for the subvectors separately. Compare the codebook space complexity with and without split vector quantization. Give the codebook searching time complexity improvement by using splitting vector quantization.
9. Discuss the advantage of using an algebraic codebook in CELP coding.
10. The LPC-10 speech coder's quality deteriorates rapidly with strong background noise. Discuss why MELP works better in the same noisy conditions.
11. Give a simple time domain method for pitch estimation based on the autocorrelation function. What problem will this simple scheme have when based on one speech frame? If we have three speech frames, including a previous frame and a future frame, how can we improve the estimation result?
12. On the receiver side, speech is usually generated based on two frames' parameters instead of one, to avoid abrupt transitions. Give two possible methods to obtain smooth transitions. Use the LPC codec to illustrate your idea.

References

1. N.S. Jayant, P. Noll, *Digital Coding of Waveforms* (Prentice-Hall, Upper Saddle River, 1984)
2. J.C. Bellamy, *Digital Telephony* (Wiley, Hoboken, 2000)
3. T.E. Tremain, The government standard linear predictive coding algorithm: LPC-10. Speech Technol. 1(**2**), 40–49 (1982)
4. J.P. Campbell Jr., T.E. Tremain, V.C. Welch, in *Advances in Speech Coding*, The DOD 4.8 kbps Standard (Proposed Federal Standard 1016), (Kluwer Academic Publishers, Boston, 1991)
5. Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s. ITU-T recommendation G.723.1 (1996), http://www.itu.int/rec/T-REC-G.723.1/e
6. GSM enhanced full rate (EFR) speech transcoding (GSM 06.60). European Telecommunications Standards Institute v.8.0.1 (1999)

7. TDMA Cellular/PCS radio interface-enhanced full rate speech codec standard. TIA/EIA/IS-641-A (1998), http://engineers.ihs.com/document/abstract/OVXADAAAAAAAAAAA
8. Coding of speech at 16 kbit/s using low-delay code excited linear programming. ITU-T Recommendation G.728 (1992), http://www.itu.int/rec/T-REC-G.728/e
9. Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP). ITU-T Recommendation G.729 (1996), http://www.itu.int/rec/T-REC-G.729/e
10. D.W. Griffin, J.S. Lim, Multi-band excitation vocoder. IEEE Trans. ASSP **36**(8), 1223–1235 (1988)
11. M.S. Brandstein, P.A. Monta, J.C. Hardwick, J.S. Lim, A real-time implementation of the improved MBE speech coder. Int. Conf. on Acoustics, Speech, and Signal Proc. (1990), pp. 5–8
12. T.P. Barnwellm III, A.V. McCree, Mixed excitation LPC vocoder model for low bit rate speech coding. IEEE Trans. Speech Audio Proc. **3**(4), 242–250 (1995)