Chapter 1

How is speech processed in a cell phone conversation?

T. Dutoit(°), N. Moreau(*), P. Kroon(+)

(°) Faculté Polytechnique de Mons, Belgium
(*) Ecole Nationale Supérieure des Télécommunications, Paris, France
(+) LSI, Allentown, PA, USA

Every cell phone solves 10 linear equations in 10 unknowns every 20 milliseconds

Although most people see the cell phone as an extension of conventional wired phone service or POTS (plain old telephone service), the truth is that cell phone technology is extremely complex and a marvel of technology. Very few people realize that these small devices perform hundreds of millions of operations per second to be able to maintain a phone conversation. If we take a closer look at the module that converts the electronic version of the speech signal into a sequence of bits, we see that for every 20 ms of input speech, a set of speech model parameters is computed and transmitted to the receiver. The receiver converts these parameters back into speech. In this chapter, we will see how linear predictive (LP) analysis–synthesis lies at the very heart of mobile phone transmission of speech. We first start with an introduction to linear predictive speech modeling and follow with a MATLAB-based proof of concept.

1.1 Background – Linear predictive processing of speech

Speech is produced by an excitation signal generated in our throat, which is modified by resonances produced by different shapes of our vocal, nasal, and pharyngeal tracts. This excitation signal can be the glottal pulses produced by the periodic opening and closing of our vocal folds (which creates *voiced* speech such as the vowels in "*voice*"), or just some continuous air flow pushed by our lungs (which creates *unvoiced* speech such as the last sound in "*voice*"), or even a combination of both at the same time (such as the first sound in "*voice*").

The periodic component of the glottal excitation is characterized by its fundamental frequency F_0 (Hz) called *pitch*[1]. The resonant frequencies of the vocal, oral, and pharyngeal tracts are called *formants*. On a spectral plot of a speech frame, pitch appears as narrow peaks for fundamental and harmonics; formants appear as wide peaks of the envelope of the spectrum (Fig. 1.1).

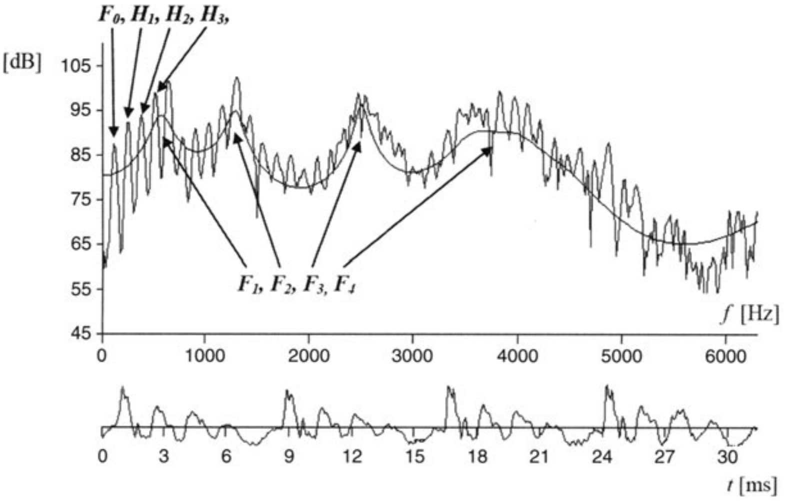

Fig. 1.1 A 30- ms frame of voiced speech (*bottom*) and its spectrum (shown here as the magnitude of its FFT). Harmonics are denoted as H_1, H_2, H_3, etc.; formants are denoted as F_1, F_2, F_3, etc. The spectral envelope is shown here for convenience; it implicitly appears only in the regular FFT

1.1.1 The LP model of speech

As early as 1960, Fant proposed a linear model of speech production (Fant 1960), termed as the *source-filter model*, based on the hypothesis that the

[1] Strictly speaking, pitch is defined as the *perceived* fundamental frequency.

glottis and the vocal tract are fully uncoupled. This model led to the well-known *autoregressive* (AR) or *linear predictive* (LP)[2] model of speech production (Rabiner and Shafer 1978), which describes speech $s(n)$ as the output $\tilde{s}(n)$ of an *all-pole* filter $1/A(z)$ excited by $\tilde{e}(n)$:

$$\tilde{S}(z)=\tilde{E}(z)\frac{1}{\sum_{i=0}^{p} a_i z^{-i}}=\tilde{E}(z)\frac{1}{A_p(z)} \qquad (a_0=1) \tag{1.1}$$

where $\tilde{S}(z)$ and $\tilde{E}(z)$ are the Z transforms of the speech and excitation signals, respectively, and p is the *prediction order*. The excitation of the LP model (Fig. 1.2) is assumed to be either a sequence of regularly spaced pulses (whose period T_0 and amplitude σ can be adjusted) or white Gaussian noise (whose variance σ^2 can be adjusted), thereby implicitly defining the so-called voiced/unvoiced (V/UV) decision. The filter $1/A_p(z)$ is termed as the *synthesis filter* and $A_p(z)$ is called the *inverse filter.*

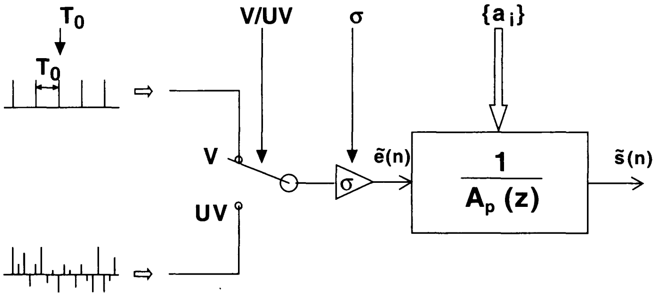

Fig. 1.2 The LP model of speech production

Equation (1.1) implicitly introduces the concept of linear predictability of speech (hence the name of the model), which states that each speech sample can be expressed as a weighted sum of the p previous samples, plus some excitation contribution:

$$\tilde{s}(n)=\tilde{e}(n)-\sum_{i=1}^{p} a_i\tilde{s}(n-i) \tag{1.2}$$

[2] Sometimes it is denoted as the *LPC model (linear predictive coding)* because it has been widely used for speech coding.

1.1.2 The LP estimation algorithm

From a given signal, a practical problem is to find the best set of prediction coefficients – that is, the set that minimizes modeling errors – by trying to minimize audible differences between the original signal and the one that is produced by the model of Fig. 1.2. This implies to estimate the value of the LP parameters: pitch period T_0, gain σ, V/UV switch position, and prediction coefficients $\{a_i\}$.

Pitch and voicing (*V/UV*) determination is a difficult problem. Although speech seems periodic, it is never truly the case. Glottal cycle amplitude varies from period to period (*shimmer*) and its period itself is not constant (*jitter*). Moreover, the speech waveform reveals only filtered glottal pulses rather than glottal pulses themselves. This makes a realistic measure of T_0 even more complex. In addition, speech is rarely completely voiced; its additive noise components make pitch determination even harder. Many techniques have been developed to estimate T_0 (see Hess 1992; de la Cuadra 2007).

The estimation of σ and of the prediction coefficients can be performed simultaneously and fortunately independently of the estimation of T_0.

For a given speech signal *s(n)*, imposing the value of the $\{a_i\}$ coefficients in the model results in the *prediction residual* signal, *e(n)*:

$$e(n)=s(n)+\sum_{i=1}^{p}a_i s(n-i) \tag{1.3}$$

which is simply the output of the inverse filter excited by the speech signal (Fig. 1.3).

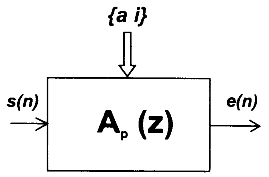

Fig. 1.3 Inverse filtering of speech

The principle of AR estimation is to choose the set $\{a_1,a_2,...a_p\}$, which minimizes the expectation $E(e^2(n))$ of the residual energy:

$$\{a_i\}^{opt} = \underset{a_i}{\arg\min}\left(E(e^2(n))\right) \tag{1.4}$$

As a matter of fact, it can be shown that, if $s(n)$ is stationary, the synthetic speech $\tilde{s}(n)$ produced by the LP model (Fig. 1.2) using this specific set of prediction coefficients in Equation (1.2) will exhibit the same spectral envelope as $s(n)$. Since the excitation of the LP model (pulses or white noise) has a flat spectral envelope, this means that the frequency response of the synthesis filter will approximately match the spectral envelope of $s(n)$ and that the spectral envelope of the LP residual will be approximately flat. In a word, inverse filtering decorrelates speech.

Developing the LMSE (least mean squared error) criterion (1.4) easily leads to the so-called set of p Yule-Walker linear equations:

$$\begin{bmatrix} \phi_{xx}(0) & \phi_{xx}(1) & \dots & \phi_{xx}(p-1) \\ \phi_{xx}(1) & \phi_{xx}(0) & \dots & \phi_{xx}(p-2) \\ \dots & \dots & \dots & \dots \\ \phi_{xx}(p-1) & \phi_{xx}(p-2) & \dots & \phi_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_p \end{bmatrix} = - \begin{bmatrix} \phi_{xx}(1) \\ \phi_{xx}(2) \\ \dots \\ \phi_{xx}(p) \end{bmatrix} \tag{1.5}$$

in which $\phi_{xx}(k)$ $(k = 0 \dots p)$ are the p+1 first autocorrelation coefficients of $s(n)$. After solving this set of equations, the optimal value of σ is then given by the following equation:

$$\sigma^2 = \sum_{i=0}^{p} a_i \phi_{xx}(i) \tag{1.6}$$

It should be noted that since Equations (1.5) are based only on the autocorrelation function of $s(n)$, the model does not try to imitate the exact speech waveform, but rather its spectral envelope (based on the idea that our ear is more sensitive to the amplitude spectrum than to the phase spectrum).

1.1.3 LP processing in practice

Since speech is nonstationary, the LP model is applied on speech *frames* (typically 30 ms long, with an overlap of 20 ms; Fig. 1.4) in which the signal is assumed to be stationary given the inertia of the articulatory muscles[3].

Speech samples are usually weighted using a *weighting window* (typically a 30-ms-long Hamming window). This prevents the first samples of each frame, which cannot be correctly predicted, from having too much weight in Equation (1.4) by producing higher values of $e^2(n)$.

The $\phi_{xx}(k)$ $(k=0...p)$ autocorrelation coefficients are then estimated on a limited number of samples (typically 240 samples, for 30 ms of speech with a sampling frequency of 8 kHz). The prediction order p (which is also the number of poles in the all-pole synthesis filter) is chosen such that the resulting synthesis filter has enough degrees of freedom to copy the spectral envelope of the input speech. Since there is approximately one formant per kilohertz of bandwidth of speech, at least $2B$ poles are required (where B is the signal bandwidth in kHz, i.e., half the sampling frequency). Two more poles are usually added for modeling the glottal cycle waveform (and also empirically, because the resulting LPC speech sounds better). For telephone-based applications, working with a sampling frequency of 8 kHz, this leads to $p=10$.

Although Equation (1.5) can be solved with any classical matrix inversion algorithm, the so-called *Levinson–Durbin* algorithm is preferred for its speed, as it takes into account the special structure of the matrix (all elements on diagonals parallel to the principal diagonal are equal; this characterizes a *Toeplitz* matrix). See Rabiner and Schafer (1978) or Quatieri (2002) for details.

The *prediction coefficients* $\{a_i\}$ are finally computed for every frame (i.e., typically every 10–20 ms).

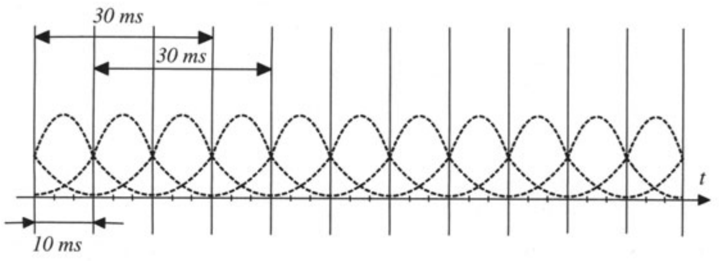

Fig. 1.4 Frame-based processing of speech (shown here with a frame length of 30 ms and a shift of 10 ms)

[3] In practice, this is only an approximation, which tends to be very loose for plosives, for instance.

The complete block diagram of an LPC speech analysis–synthesis system is given in Fig. 1.5.

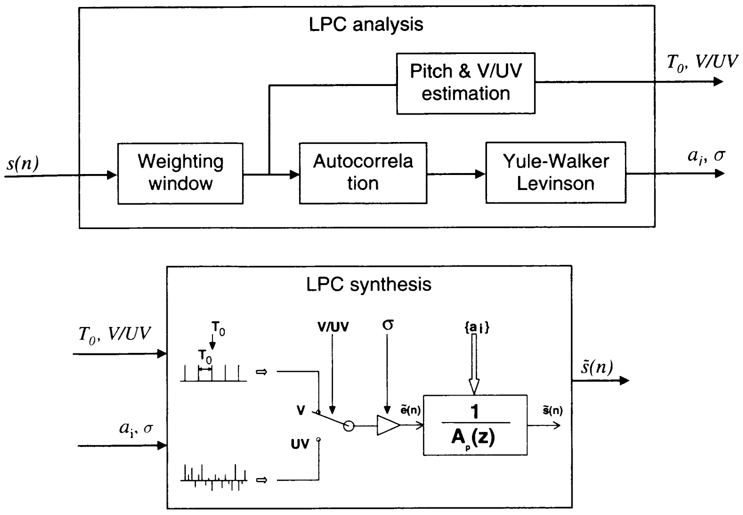

Fig. 1.5 A linear predictive speech analysis–synthesis system

1.1.4 Linear predictive coders

The LPC analysis–synthesis system, which has been described above, is not exactly the one embedded in cell phones.

It is, however, implemented in the so-called NATO LPC10 standard (NATO, 1984), which was used for satellite transmission of speech communications until 1996. This norm makes it possible to encode speech with a bit rate as low as 2,400 bits/s (frames are 22.5 ms long, and each frame is coded with 54 bits: 7 bits for pitch and *V/UV* decision, 5 bits for the gain, and 42 bits for the prediction coefficients[4]). In practice, prediction coefficients are actually not used as such; the related *reflection coefficients* or *log area ratios* are preferred, since they have better quantization properties. Quantization of prediction coefficients can result in unstable filters.

The number of bits in LPC10 was chosen such that it does not bring audible artifacts to the LPC speech. The example LPC speech produced in Section 1.2 is therefore a realistic example of typical LPC10 speech.

[4] Advanced LP coders, such as CELP, have enhanced prediction coefficients coding down to 30 bits.

Clearly this speech coder suffers from the limitations of the poor (and binary!) excitation model. Voiced fricatives, for instance, cannot be adequately modeled since they exhibit voiced *and* unvoiced features simultaneously. Moreover, the LPC10 coder is very sensitive to the efficiency of its voiced/unvoiced detection and F_0 estimation algorithms. Female voices, whose higher F_0 frequency sometimes results in a second harmonic at the center of the first formant, often lead to F_0 errors (the second harmonic being mistaken for F_0).

One way of enhancing the quality of LPC speech is obviously to reduce the constraints on the LPC excitation so as to allow for a better modeling of the prediction residual $e(n)$ by the excitation $\tilde{e}(n)$. As a matter of fact, passing this residual through the synthesis filter $1/A(z)$ produces the original speech (Fig. 1.6, which is the inverse of Fig. 1.3).

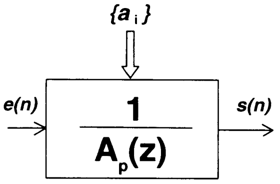

Fig. 1.6 Passing the prediction residual through the synthesis filter produces the original speech signal

The *multipulse excited* (MPE; Atal and Remde 1982) was an important step in this direction, as it was the first approach to implement an analysis-by-synthesis process (i.e., a closed loop) for the estimation of the excitation features. The MPE excitation is characterized by the positions and amplitudes of a limited number of pulses per frame (typically 10 pulses per 10 ms frame; Fig. 1.7). Pitch estimation and voiced/unvoiced decision are no longer required. Pulse positions and amplitudes are chosen iteratively (Fig. 1.8) so as to minimize the energy of the modeling error (the difference between the original speech and the synthetic speech). The error is filtered by a *perceptual filter* before its energy is computed:

$$P(z) = \frac{A(z)}{A(z/\gamma)} \tag{1.7}$$

The role of this filter, whose frequency response can be set to any intermediate between all pass response (γ=1) and the response of the inverse filter ($\gamma = 0$), is to reduce the contributions of the formants to the estimation of the error. The value of γ is typically set to 0.8.

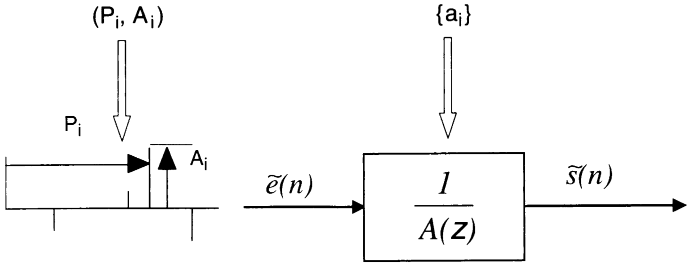

Fig. 1.7 The MPE decoder

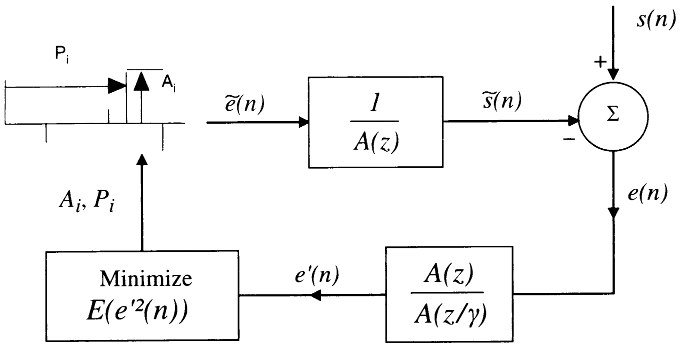

Fig. 1.8 Estimation of the MPE excitation by an analysis-by-synthesis loop in the MPE encoder

The *code-excited linear prediction* (CELP) coder (Schroeder and Atal, 1985) further extended the idea of analysis-by-synthesis speech coding by using the concept of *vector quantization* (VQ) for the excitation sequence. In this approach, the encoder selects one excitation sequence from a predefined *stochastic codebook* of possible sequences (Fig. 1.9) and sends only the index of the selected sequence to the decoder, which has a similar codebook. Although the lowest quantization rate for scalar quantization is 1 bit per sample, VQ allows fractional bit rates. For example, quantizing two samples simultaneously using a 1-bit codebook will result in 0.5 bits per sample. More typical values are a 10-bit codebook with codebook vectors of dimension 40, resulting in 0.25 bits per sample. Given the very high variability of speech frames, however (due to changes in glottal excitation *and* vocal tract), vector-quantized speech frames would be possible only with a very large codebook. The great idea of CELP is precisely to perform VQ on LP residual sequences: as we have seen in Section 1.1.2, the LP residual has a flat spectral envelope, which makes it easier to produce a small but somehow exhaustive codebook of LP residual sequences. CELP can thus be seen as an *adaptive vector quantization* scheme of speech frames (adaptation being performed by the synthesis filter).

CELP additionally takes advantage of the periodicity of voiced sounds to further improve predictor efficiency. A so-called *long-term predictor* filter is cascaded with the synthesis filter, which enhances the efficiency of the codebook. The simplest long-term predictor consists of a simple variable delay with adjustable gain (Fig. 1.10).

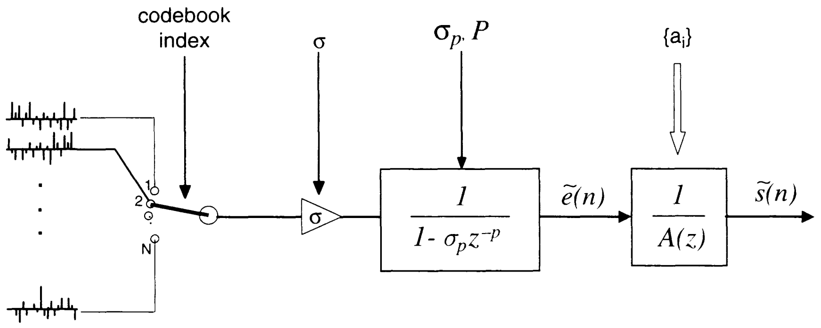

Fig. 1.9 The CELP decoder

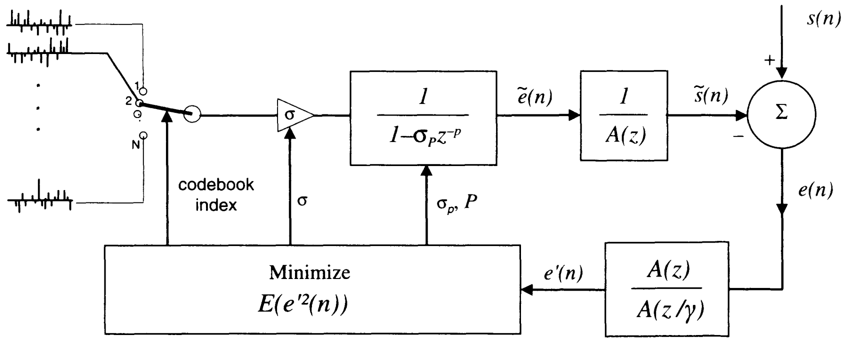

Fig. 1.10 Estimation of the CELP excitation by an analysis-by-synthesis loop in the CELP encoder

Various coders have been developed after MPE and CELP using the same analysis-by-synthesis principle with the goal of enhancing CELP quality while further reducing bit rate, among which are the mixed-excitation linear prediction (MELP; McCree and Barnwell, 1995) and the harmonic and vector excitation coding (HVXC; Matsumoto et al. 1997). In 1996, LPC-10 was replaced by MELP to be the United States Federal Standard for coding at 2.4 kbps.

From 1992 to 1996, GSM (global system for mobile communication) phones embedded a particular form of MPE, the *RPE-LPC* (regular pulse excited; Kroon et al. 1986) coder, with additional constraints on the positions of the pulses: the RPE pulses were evenly spaced (but their amplitude, as well as the position of the first pulse, is left open). Speech is divided into 20 ms frames, each of which is encoded as 260 bits, giving a total bit rate of 13 kbps. In 1996, this so-called *full-rate* (FR) codec was replaced by the *enhanced full-rate* (EFR) codec, implementing a variant of CELP termed as algebraic-CELP (ACELP, Salami et al. 1998). The ACELP codebook structure allows efficient searching of the optimal codebook index thereby eliminating one of the main drawbacks of CELP which is its complexity. The EFR coder operates at 11.2 kbps and produces better speech quality than the FR coder at 13 kb/s. A variant of the ACELP coder has been standardized by ITU-T as G.729 for operation at a bit rate of 8 kbps. Newer generations of coders that are used in cell phones are all based on the CELP principle and can operate at bit rates as low as 4.75 – 11.2 kbps.

1.2 MATLAB proof of concept : ASP_cell_phone.m

We will first examine the contents of a speech file (Section 1.2.1) and perform LP analysis and synthesis on a voiced (Section 1.2.2) and an unvoiced frame (Section 1.2.3). We will then generalize this approach to the complete speech file by first synthesizing all frames as voiced and imposing a constant pitch (Section 1.2.4), then by synthesizing all frames as unvoiced (Section 1.2.5), and finally by using the original pitch[5] and voicing information as in LPC10 (Section 1.2.6). We will conclude this section by changing LPC10 into CELP (Section 1.2.7).

[5] By "original pitch," we mean the pitch that can be measured on the original signal.

1.2.1 Examining a speech file

Let us load file "speech.wav," listen to it, and plot its samples (Fig. 1.11). This file contains the sentence "Paint the circuits" sampled at 8 kHz, with 16 bits.[6]

```
speech=wavread('speech.wav');
plot(speech)
xlabel('Time (samples)'); ylabel('Amplitude');
sound(speech,8000);
```

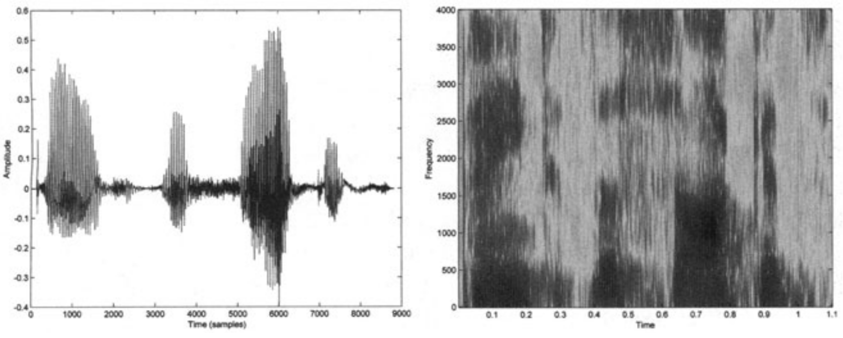

Fig. 1.11 Input speech: the speech.wav file (*left*: waveform; *right*: spectrogram)

The file is about 1.1 s long (9,000 samples). One can easily spot the position of the four vowels appearing in this plot, since vowels usually have higher amplitude than other sounds. The vowel "e" in "the", for instance, is approximately centered on sample 3,500.

As such, however, the speech waveform is not "readable," even by an expert phonetician. Its information content is hidden. In order to reveal it to the eyes, let us plot a spectrogram of the signal (Fig. 1.11). We then choose a *wideband spectrogram*[7] by imposing the length of each frame to be approximately 5 ms long (40 samples) and a hamming weighting window.

```
specgram(speech,512,8000,hamming(40))
```

In this plot, pitch periods appear as vertical lines. As a matter of fact, since the length of analysis frames is very small, some frames fall on the

[6] This sentence was taken from the Open Speech Repository on the web.

[7] A wideband spectrogram uses a small amount of samples (typically less than the local pitch period) so as to better reveal formants.

peaks (resp., on the valleys) of pitch periods and thus appear as a darker (resp., lighter) vertical lines.

In contrast, formants (resonant frequencies of the vocal tract) appear as dark (and rather wide) horizontal traces. Although their frequency is not easy to measure with precision, experts looking at such a spectrogram can actually often read it (i.e., guess the corresponding words). This clearly shows that formants are a good indicator of the underlying speech sounds.

1.2.2 Linear prediction synthesis of 30 ms of voiced speech

Let us extract a 30-ms frame from a voiced part (i.e., 240 samples) of the speech file and plot its samples (Fig. 1.12).

```
input_frame=speech(3500:3739);
plot(input_frame);
```

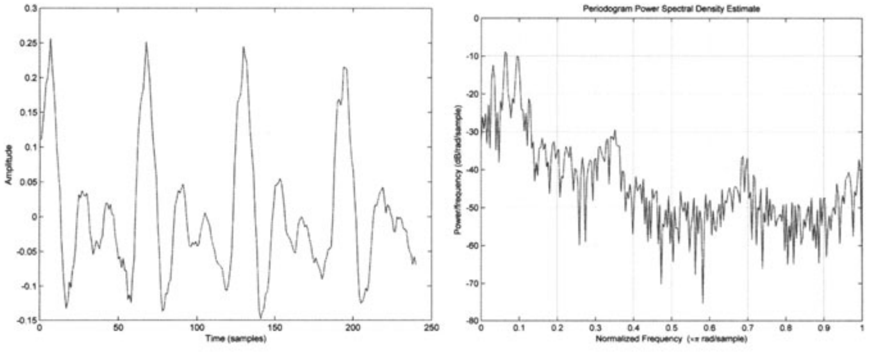

Fig. 1.12 A 30-ms-long voiced speech frame taken from a vowel (*left*: waveform; *right*: periodogram)

As expected this sound is approximately periodic (period=65 samples, i.e., 80 ms; fundamental frequency = 125 Hz). Note, though, that this is only apparent; in practice, no sequence of samples can be found more than once in the frame.

Now let us see the spectral content of this speech frame (Fig. 1.12) by plotting its *periodogram* on 512 points (using a normalized frequency axis; remember π corresponds to $F_s/2$, i.e., to 4,000 Hz here).

```
periodogram(input_frame,[],512);
```

The fundamental frequency appears again at around 125 Hz. One can also roughly estimate the position of formants (peaks in the spectral envelope) at ± 300, 1,400, and 2,700 Hz.

Let us now fit an LP model of order 10 to our voiced frame.[8] We obtain the prediction coefficients (ai) and the variance of the residual signal (sigma_square).

```
[ai, sigma_square]=lpc(input_frame,10);
sigma=sqrt(sigma_square);
```

The estimation parameter inside LPC is called the Levinson–Durbin algorithm. It chooses the coefficients of an FIR filter *A(z)* so that when passing the input frame into *A(z)*, the output, termed as the prediction residual, has minimum energy. It can be shown that this leads to a filter which has anti-resonances wherever the input frame has a formant. For this reason, the *A(z)* filter is termed as the "inverse" filter. Let us plot its frequency response (on 512 points) and superimpose it to that of the "synthesis" filter *1/A(z)* (Fig. 1.13).

```
[HI,WI]=freqz(ai, 1, 512);
[H,W]=freqz(1,ai, 512);
plot(W,20*log10(abs(H)),'-',WI,20*log10(abs(HI)),'--');
```

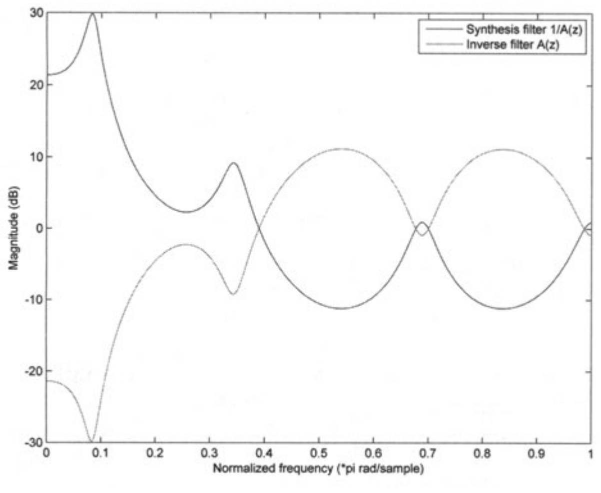

Fig. 1.13 Frequency responses of the inverse and synthesis filters

[8] We do not apply windowing prior to LP analysis now, as it has no tutorial benefit. We will add it in subsequent sections.

In other words, the frequency response of the filter *1/A(z)* matches the spectral amplitude envelope of the frame. Let us superimpose this frequency response to the periodogram of the vowel (Fig. 1.14).[9]

```
periodogram(input_frame,[],512,2)
hold on;
plot(W/pi,20*log10(sigma*abs(H)));
hold off;
```

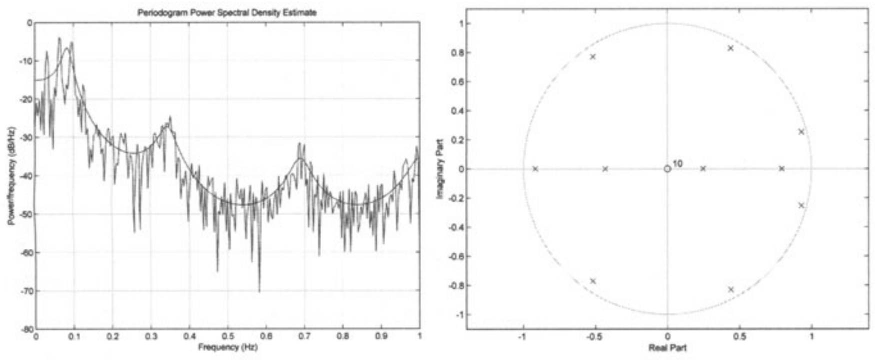

Fig. 1.14 *Left*: Frequency response of the synthesis filter superimposed with the periodogram of the frame; *right*: poles and zeros of the filter

In other words, the LPC fit has automatically adjusted the poles of the synthesis filter close to the unit circle at angular positions chosen to imitate formant resonances (Fig. 1.14).

```
zplane(1,ai);
```

If we apply the inverse of this filter to the input frame, we obtain the prediction residual (Fig. 1.15).

```
LP_residual=filter(ai,1,input_frame);
plot(LP_residual)
periodogram(LP_residual,[],512);
```

[9] The `periodogram` function of MATLAB actually shows the so-called *one-sided periodogram*, which has twice the value of the two-sided periodogram in [0, $F_s/2$]. In order to force MATLAB to show the real value of the two-sided periodogram in [0, $F_s/2$], we claim $F_s = 2$.

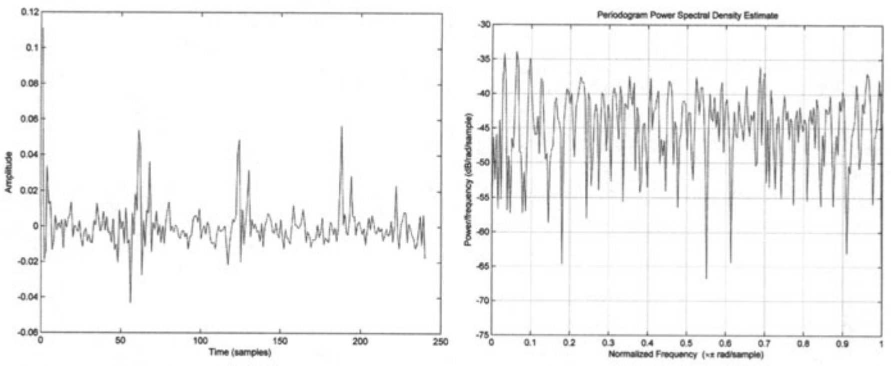

Fig. 1.15 The prediction residual (*left*: waveform; *right*: periodogram)

Let us compare the spectrum of this residual to the original spectrum. The new spectrum is approximately flat; its fine spectral details, however, are the same as those of the analysis frame. In particular, its pitch and harmonics are preserved.

For obvious reasons, applying the synthesis filter to this prediction residual results in the analysis frame itself (since the synthesis filter is the inverse of the inverse filter).

```
output_frame=filter(1, ai,LP_residual);
plot(output_frame);
```

The LPC model actually models the prediction residual of voiced speech as an impulse train with adjustable pitch period and amplitude. For the speech frame considered, for instance, the LPC ideal excitation is a sequence of pulses separated by 64 zeros (so as to impose a period of 65 samples; Fig. 1.16). Note we multiply the excitation by some gain so that its variance matches that of the residual signal.

```
excitation = [1;zeros(64,1);1;zeros(64,1);1;zeros(64,1);…
              1;zeros(44,1)];
gain=sigma/sqrt(1/65);
plot(gain*excitation);
periodogram(gain*excitation,[],512);
```

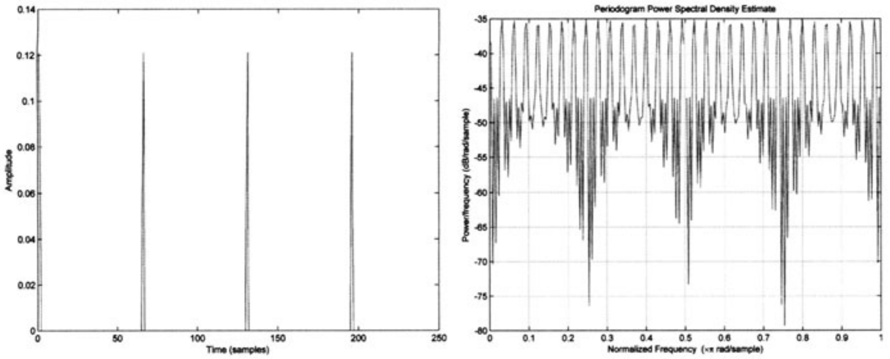

Fig. 1.16 The LPC excitation (*left*: waveform; *right*: periodogram)

Clearly, as far as the waveform is concerned, the LPC excitation is far from similar to the prediction residual. Its spectrum (Fig. 1.16), however, has the same broad features as that of the residual: flat envelope and harmonic content corresponding to F_0. The main difference is that the excitation spectrum is "over-harmonic" compared to the residual spectrum.

Let us now use the synthesis filter to produce an artificial "e."

```
synt_frame=filter(gain,ai,excitation);
plot(synt_frame);
periodogram(synt_frame,[],512);
```

Although the resulting waveform is obviously different from the original one (this is due to the fact that the LP model does not account for the phase spectrum of the original signal), its spectral envelope is identical. Its fine harmonic details, though, also widely differ: the synthetic frame is actually "over-harmonic" compared to the analysis frame (Fig. 1.17).

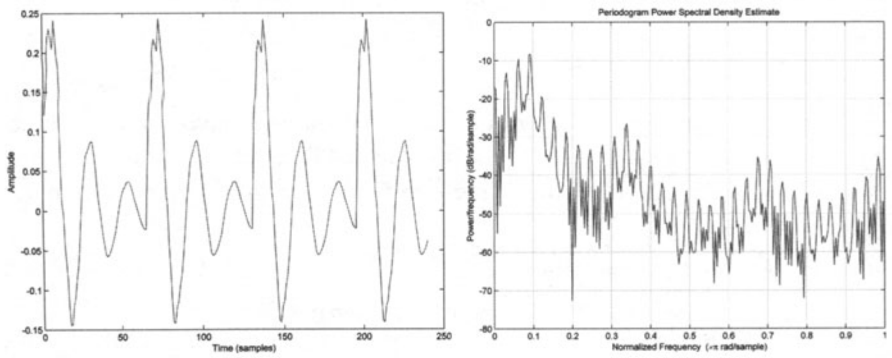

Fig. 1.17 Voiced LPC speech (*left*: waveform; *right*: periodogram)

1.2.3 Linear prediction synthesis of 30 ms of unvoiced speech

It is easy to apply the same process to an unvoiced frame and compare the final spectra again. Let us first extract an unvoiced frame and plot it (Fig. 1.18). As expected, no clear periodicity appears.

```
input_frame=speech_HF(4500:4739);
plot(input_frame);
```

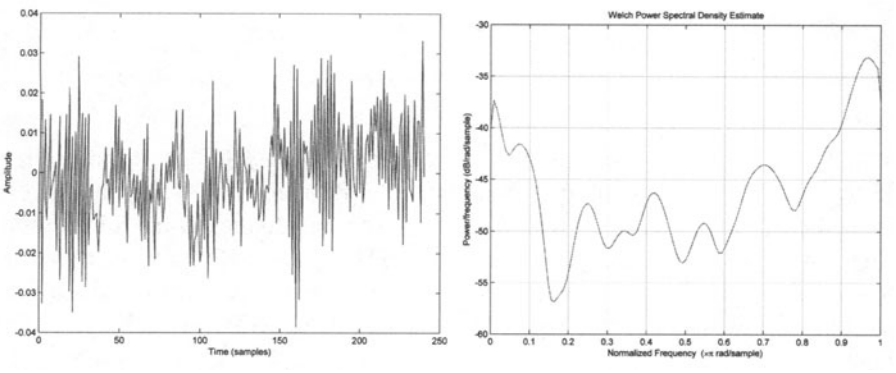

Fig. 1.18 A 30-ms-long frame of unvoiced speech (*left*: waveform; *right*: power spectral density)

Now let us see the spectral content of this speech frame. Note that, since we are dealing with noisy signals, we use the *averaged periodogram* to estimate power spectral densities, although with less-frequency resolution than using a simple periodogram. The MATLAB pwlech function does this with eight subframes by default and 50% overlap.

```
pwelch(input_frame);
```

Let us now apply an LP model of order 10 and synthesize a new frame. Synthesis is performed by all-pole filtering a Gaussian white noise frame with standard deviation set to the prediction residual standard deviation, σ.

```
[ai, sigma_square]=lpc(input_frame,10);
sigma=sqrt(sigma_square);
excitation=randn(240,1);
synt_frame=filter(sigma,ai,excitation);
plot(synt_frame);

pwelch(synt_frame);
```

The synthetic waveform (Fig. 1.19) has no sample in common with the original waveform. The spectral envelope of this frame, however, is still similar to the original one, enough at least for both the original and synthetic signals to be perceived as the same colored noise.[10]

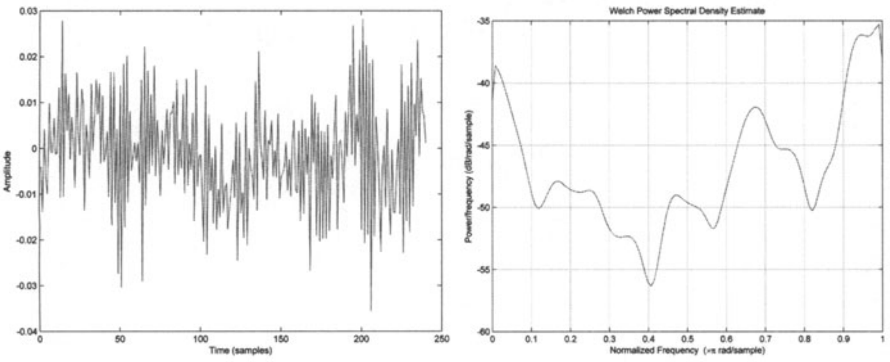

Fig. 1.19 Unvoiced LPC speech (*left*: waveform; *right*: psd)

1.2.4 Linear prediction synthesis of a speech file, with fixed F_0

We will now loop the previous operations for the complete speech file using 30 ms analysis frames overlapping by 20 ms. Frames are now weighted with a Hamming window. At synthesis time, we simply synthesize 10 ms of speech and concatenate the resulting synthetic frames to obtain the output speech file. Let us choose 200 Hz as synthesis F0, for convenience: this way each 10-ms excitation frame contains exactly two pulses.

```
for i=1:(length(speech)-160)/80; % number of frames

    % Extracting the analysis frame
    input_frame=speech_HF((i-1)*80+1:(i-1)*80+240);

    % Hamming window weighting and LPC analysis
    [ai, sigma_square]=lpc(input_frame.*hamming(240),10);
    sigma=sqrt(sigma_square);

    % Generating 10 ms of excitation
    % = 2 pitch periods at 200 Hz
    excitation=[1;zeros(39,1);1;zeros(39,1)];
    gain=sigma/sqrt(1/40);

    % Applying the synthesis filter
```

[10] Although both power spectral densities have identical spectral slopes, one should not expect them to exhibit a close match in terms of their details, since only LPC modeling reproduces the smooth spectral envelope of the original signal.

```
        synt_frame=filter(gain, ai,excitation);

        % Concatenating synthesis frames
        synt_speech_HF=[synt_speech_HF;synt_frame];

    end
```

The output waveform basically contains a sequence of LP filter impulse responses. Let us zoom on 30 ms of LPC speech (Fig. 1.20).

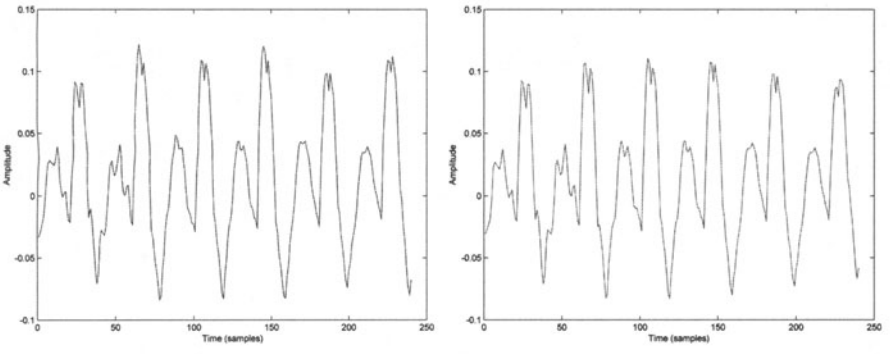

Fig. 1.20 Zoom on 30 ms of LPC speech (*left*: with internal variable reset; *right*: with internal variable memory)

It appears that in many cases the impulse responses have been cropped to the 10-ms synthetic frame size. As a matter of fact, since each synthesis frame was composed of two identical impulses, one should expect our LPC speech to exhibit pairs of identical pitch periods. This is not the case, because for producing each new synthetic frame, the internal variables of the synthesis filter are implicitly reset to zero. We can avoid this problem by maintaining the internal variables of the filter from the end of each frame to the beginning of the next one.

We initialize a vector z with 10 zeros and change the synthesis code into

```
    % Applying the synthesis filter
    % Taking care of the internal variables of the filter
    gain=sigma/sqrt(1/40);
    [synt_frame,z]=filter(gain, ai, excitation, z);
```

This time the end of each impulse response is properly added to the beginning of the next one, which results in more smoothly evolving periods (Fig. 1.20).

If we want to synthesize speech with constant pitch period length different from a submultiple of 80 samples (say, N0=65 samples), we additionally need to take care of a possible pitch period offset in the excitation signal. After initializing this `offset` to zero, we simply change the excitation code into

```
% Generating 10 ms of excitation
% taking a possible offset into account

% if pitch period length > excitation frame length
if offset>=80
    excitation=zeros(80,1);
    offset=offset-80;
else
    % complete the previously unfinished pitch period
    excitation=zeros(offset,1);
    % for all pitch periods in the remaining of the frame
    for j=1:floor((80-offset)/N0)
        % add one excitation period
        excitation=[excitation;1;zeros(N0-1,1)];
    end;
    % number of samples left in the excitation frame
    flush=80-length(excitation);
    if flush~=0
        % fill the frame with a partial pitch period
        excitation=[excitation;1;zeros(flush-1,1)];
        % remember to fill the remaining of the period in
        % next frame
        offset=N0-flush;
    else offset=0;
    end
end
gain=sigma/sqrt(1/N0);
```

1.2.5 Unvoiced linear prediction synthesis of a speech file

Synthesizing the complete speech file as LPC unvoiced speech is easy. Periodic pulses are simply replaced by white noise, as in Section 1.2.3.

```
% Generating 10 ms of excitation
excitation=randn(80,1); % White Gaussian noise
gain=sigma;
```

As expected, the resulting speech sounds like whisper.

1.2.6 Linear prediction synthesis of a speech file, with original F_0

We will now synthesize the same speech using the original F_0. We will thus have to deal with the additional problems of pitch estimation (on a frame-by-frame basis), including voiced/unvoiced decision. This approach is similar to that of the LPC10 coder (except that we do not quantize coefficients here). We change the excitation generation code into

```
% local synthesis pitch period (in samples)
N0=pitch(input_frame);
```

```
% Generating 10 ms of excitation
if N0~=0 % voiced frame
    % Generate 10 ms of voiced excitation
    % taking a possible offset into account

    (same code as in Section 1.2.4)

else
    % Generate 10 ms of unvoiced voiced excitation

    (same code as in Section 1.2.5)

    offset=0; % reset for subsequent voiced frames
end;
```

MATLAB function involved:

- `T0=pitch(speech_frame)` returns the pitch period T_0 (in samples) of a speech frame (T_0 is set to zero when the frame is detected as unvoiced). T_0 is obtained from the maximum of the (estimated) autocorrelation of the LPC residual. Voiced/unvoiced decision is based on the ratio of this maximum to the variance of the residual. This simple algorithm is not optimal but will do the job for this proof of concept.

The resulting synthetic speech (Fig. 1.21) is intelligible. It shows the same formants as the original speech. It is therefore acoustically similar to the original except for the additional buzziness that has been added by the LP model.

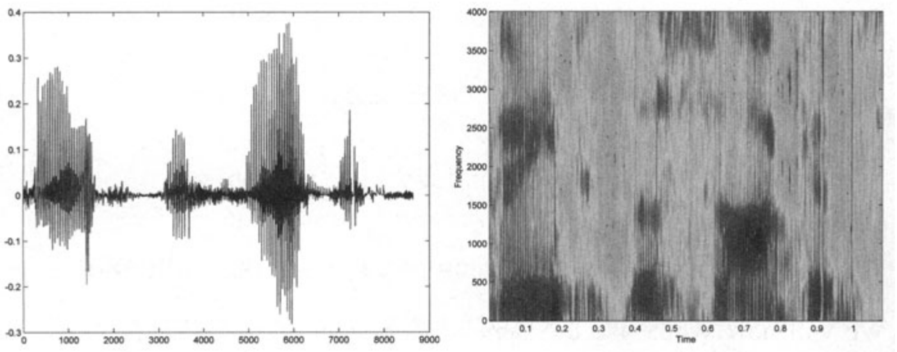

Fig. 1.21 LPC10 speech

1.2.7 CELP analysis–synthesis of a speech file

Our last step will be to replace the LPC10 excitation by a more realistic code-excited linear prediction (CELP) excitation obtained by selecting the best linear combination of excitation components from a codebook. Component selection is performed in a closed loop so as to minimize the difference between the synthetic and original signals.

We start with 30-ms LP analysis frames, shifted every 5 ms, and a codebook size of 512 vectors from which 10 components are chosen for every 5-ms synthesis frame.[11]

MATLAB function involved:

-
  ```
  [gains, indices] = find_Nbest_components(signal, ...
                     codebook_vectors, codebook_norms , N)
  ```

  This function finds the `N` best components of signal from the vectors in `codebook_vectors`, so that the residual error
  `error = signal - codebook_vectors(indices)*gains`
  is minimized. Components are found one-by-one using a greedy algorithm. When components in `codebook_vectors` are not orthogonal, the search is therefore suboptimal.

```
frame_length=240; % length of the LPC analysis frame
frame_shift=40; % length of the synthesis frames
codebook_size = 512; % number of vectors in the codebook
N_components= 10;  % number of codebook components per frame

speech=wavread('speech.wav');

% Initializing internal variables
z_inv=zeros(10,1);  % inverse filter
z_synt=zeros(10,1); % synthesis filter
synt_speech_CELP=[];

% Generating the stochastic excitation codebook
codebook = randn(frame_shift,codebook_size);

for i=1:(length(speech)-frame_length+frame_shift)/frame_shift;

    input_frame=speech((i-1)*frame_shift+1:...
                      (i-1)*frame_shift+frame_length);

    % LPC analysis of order 10
    ai = lpc(input_frame.*hamming(frame_length), 10);

    % Extracting frame_shift samples from the LPC analysis frame
    speech_frame = input_frame((frame_length-frame_shift)/2+1:...
```

[11] These values actually correspond to a rather high bit rate, but we will show in the next paragraphs how to lower the bit rate while maintaining the quality of synthetic speech.

```
                    (frame_length-frame_shift)/2+frame_shift);

    % Filtering the codebook (all column vectors)
    codebook_filt = filter(1, ai, codebook);

    % Finding speech_frame components in the filtered codebook
    % taking into account the transient stored in the internal
    % variables of the synthesis filter
    ringing = filter(1, ai, zeros(frame_shift,1), z_synt);
    signal = speech_frame - ringing;
    [gains, indices] = find_Nbest_components(signal, ...
       codebook_filt, N_components);

    % Generating the corresponding excitation as a weighted sum
    % of codebook vectors
    excitation = codebook(:,indices)*gains;

    % Synthesizing CELP speech, and keeping track of the
    % synthesis filter internal variables
    [synt_frame, z_synt] = filter(1, ai, excitation, z_synt);
    synt_speech_CELP=[synt_speech_CELP;synt_frame];

end
```

Note that this analysis–synthesis simulation is implemented as mentioned in Section 1.2.4 as an adaptive vector quantization system. This is done by passing the whole codebook through the synthesis filter, for each new frame, and searching for the best linear decomposition of the speech frame in terms of filtered codebook sequences.

Also note our use of `ringing`, which stores the natural response of the synthesis filter due to its nonzero internal variables. This response should not be taken into account in the adaptive VQ.

The resulting synthetic speech sounds more natural than in LPC10. Plosives are much better rendered, and voiced sounds are no longer buzzy, but speech sounds a bit noisy. Note that pitch and V/UV estimation are no longer required.

One can see that the closed-loop optimization leads to excitation frames, which can somehow differ from the LP residual, while the resulting synthetic speech is similar to its original counterpart (Fig. 1.22).

In the above script, though, each new frame was processed independently of past frames. Since voiced speech is strongly self-correlated, it makes sense to incorporate a long-term prediction filter in cascade with the LPC (short-term) prediction filter. In the example below, we can reduce the number of stochastic components from 10 to 5 while still increasing speech quality, thanks to long-term prediction.

```
N_components= 5;  % number of codebook components per frame
```

Since CELP excitation frames are only 5 ms long, we store them in a 256 samples circular buffer (i.e., a bit more than 30 ms of speech) for finding the best long-term prediction delay in the range [0–256] samples.

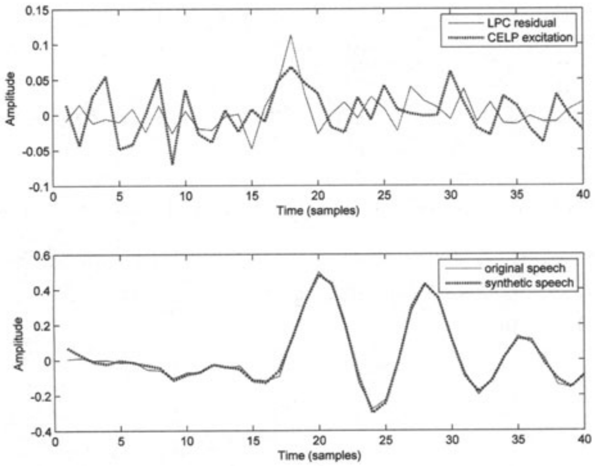

Fig. 1.22 CELP analysis–synthesis of frame #140. *Top*: CELP excitation compared to linear prediction residual. *Bottom*: CELP synthetic speech compared to original speech

```
LTP_max_delay=256; % maximum long-term prediction delay
excitation_buffer=zeros(LTP_max_delay+frame_shift,1);
```

Finding the delay itself (inside the frame-based loops) is achieved in a way very similar to finding the *N* best stochastic components in our previous example: we create a long-term prediction codebook, pass it through the synthesis filter, and search for *the* best excitation component in this filtered codebook.

```
% Building the long-term prediction codebook and filtering it
for j = 1:LTP_max_delay
    LTP_codebook(:,j) = excitation_buffer(j:j+frame_shift-1);
end
LTP_codebook_filt = filter(1, ai, LTP_codebook);

% Finding the best predictor in the LTP codebook
ringing = filter(1, ai, zeros(frame_shift,1), z_synt);
signal = speech_frame - ringing;
[LTP_gain, LTP_index] = find_Nbest_components(signal, ...
    LTP_codebook_filt, 1);

% Generating the corresponding prediction
LT_prediction= LTP_codebook(:,LTP_index)*LTP_gain;
```

Stochastic components are then searched *in the remaining signal* (i.e., the original signal minus the long-term predicted *signal*).

```
% Finding speech_frame components in the filtered codebook
% taking long term prediction into account
signal = signal - LTP_codebook_filt(:,LTP_index)*LTP_gain;
[gains, indices] = find_Nbest_components(signal, ...
    codebook_filt, N_components);
```

The final excitation is computed as the sum of the long-term predicted *excitation*.

```
excitation = LT_prediction + codebook(:,indices)*gains;
```

As can be seen in Fig. 1.23, the resulting synthetic speech is still similar to the original one, notwithstanding the reduction of the number of stochastic components.

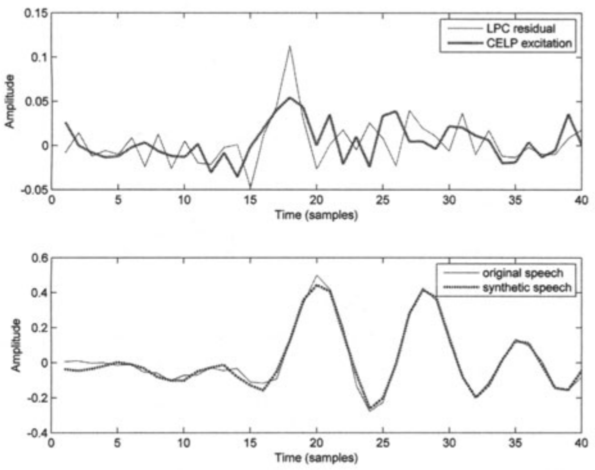

Fig. 1.23 CELP analysis–synthesis of frame #140 with long-term prediction and only five stochastic components. *Top*: CELP excitation compared to linear prediction residual. *Bottom*: CELP synthetic speech compared to original speech

Although the search for the best components in the previous scripts aims at minimizing the energy of the difference between original and synthetic speech samples, it makes sense to use the fact that the ear will be more tolerant to this difference in parts of the spectrum that are louder and vice versa. This can be achieved by applying a perceptual filter to the error, which enhances spectral components of the error in frequency bands with less energy and vice versa (Fig. 1.24).

In the following example, we still decrease the number of components from 5 to 2, with the same overall synthetic speech quality.

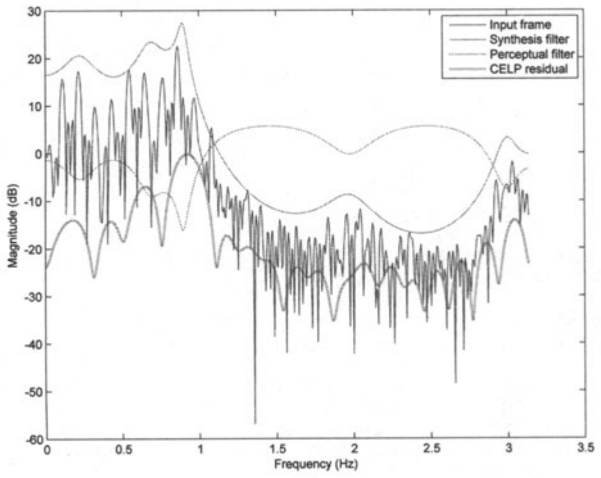

Fig. 1.24 CELP analysis–synthesis of frame #140: the frequency response of the perceptual filter approaches the inverse of that of the synthesis filter. As a result, the spectrum of the CELP residual somehow follows that of the speech frame

```
N_components= 2;  % number of codebook components per frame
```

We will apply perceptual filter $A(z)/A(z/\gamma)$ to the input frame, and filter $1/A(z/\gamma)$ to the stochastic and long-term prediction codebook vectors.[12] We will therefore need to handle their internal variables.

```
gamma = 0.8;   % perceptual factor
z_inv=zeros(10,1);  % inverse filter
z_synt=zeros(10,1); % synthesis filter
z_gamma_s=zeros(10,1); % perceptual filter for speech
z_gamma_e=zeros(10,1); % perceptual filter for excitation
```

Finding the coefficients of $A(z/\gamma)$ is easy.

```
    ai_perceptual = ai.*(gamma.^(0:(length(ai)-1)) );
```

One can then filter the input frame and each codebook.

[12] In the previous examples, the input frame was not perceptually filtered, and codebooks were passed through the synthesis filter 1/A(z).

```
% Passing the central 5ms of the input frame through
% A(z)/A(z/gamma)
[LP_residual, z_inv] = filter(ai, 1, speech_frame, z_inv);
[perceptual_speech, z_gamma_s] = filter(1, ...
                       ai_perceptual, LP_residual, z_gamma_s);

% Filtering both codebooks
LTP_codebook_filt = filter(1, ai_perceptual, LTP_codebook);
codebook_filt = filter(1, ai_perceptual, codebook);
```

The search for the best long-term predictor is performed as before, except that the perceptually filtered speech input is used as the reference from which to find codebook components.

```
% Finding the best predictor in the LTP codebook
ringing = filter(1, ai_perceptual, ...,
          zeros(frame_shift,1), z_gamma_e);
signal = perceptual_speech - ringing;
[LTP_gain, LTP_index] = find_Nbest_components(signal, ...
    LTP_codebook_filt, 1);

% Generating the corresponding prediction
LT_prediction= LTP_codebook(:,LTP_index)*LTP_gain;

% Finding speech_frame components in the filtered codebook
% taking long term prediction into account
signal = signal - LTP_codebook_filt(:,LTP_index)*LTP_gain;
[gains, indices] = find_Nbest_components(signal, ...
    codebook_filt, N_components);
```

Last but not least, one should not forget to update the internal variables of the perceptual filter applied to the excitation.

```
[ans, z_gamma_e] = filter(1, ai_perceptual, excitation, ...
    z_gamma_e);
```

While using less stochastic components than in the previous example, synthetic speech quality is maintained, as revealed by listening. The synthetic speech waveform also looks much more similar to original speech than its LPC10 counterpart (Fig. 1.25).

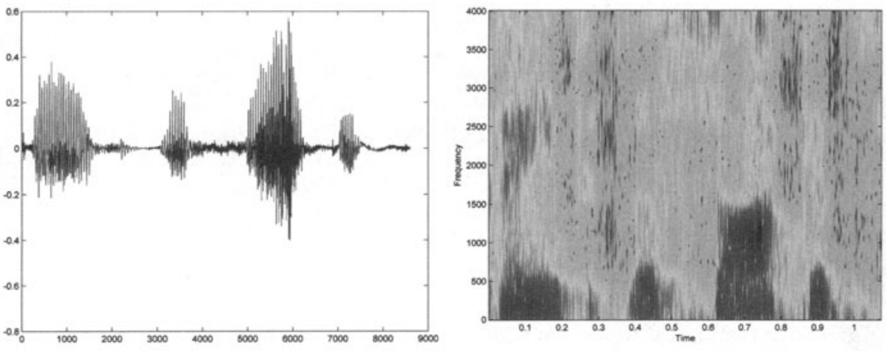

Fig. 1.25 CELP speech

One can roughly estimate the corresponding bit rate. Assuming 30 bits are enough for the prediction coefficients and each gain factor is quantized on 5 bits, we have to send for each frame: 30 bits [ai] + 7 bits [LTP index] + 5 bits [LTP gain] + 2 [stochastic components] *(9 bits [index] + 5 bits [gain]) = 70 bits every 5 ms, i.e., 14 kbps.

Note that G729 reaches a bit rate as low as 8 kbps by sending prediction coefficients only once every four frame.

1.3 Going further

Various tools and interactive tutorials on LP modeling of speech are available on the web (see Fellbaum 2007, for instance).

MATLAB code by A. Spanias for the LPC10e coder can be found on the web (Spanias and Painter 2002).

Another interesting MATLAB-based project on LPC coding, applied to wideband speech this time, can be found on the dspexperts.com website (Khan and Kashif 2003).

D. Ellis provides interesting MATLAB-based audio processing examples on his web pages (Ellis 2006), among which are a sinewave speech analysis/synthesis demo (including LPC) and a spectral warping of LPC demo.

For a broader view of speech coding standards, one might refer to Woodard (2007) or to the excellent book by Goldberg and Riek (2000).

1.4 Conclusion

We now understand how every cell phone solves a linear system of 10 equations in 10 unknowns every 20 ms which is the basis of the estimation of the LP model through Yule-Walker equations. The parameters that are actually sent from one cell phone to another are vocal tract coefficients related to the frequency response of the vocal tract and source coefficients related to the residual signal.

The fact that the vocal tract coefficients are very much related to the geometric configuration of the vocal tract for each frame of 10 ms of speech calls for an important conclusion: cell phones, in a way, transmit a picture of our vocal tract rather than the speech it produces.

In fact, the reach of LP speech modeling goes far beyond the development of cell phones. As shown by Gray (2006), its history is intermixed with that of Arpanet, the ancestor of Internet.

References

Atal BS, Remde JR (1982) A New Model LPC Excitation for Producing Natural Sounding Speech at Low Bit Rates. In: Proc. ICASSP'82, pp 614–617

de la Cuadra P (2007) Pitch Detection Methods Review [online] Available: http://www-ccrma.stanford.edu/~pdelac/154/m154paper.htm [20/2/1007]

Ellis D (2006) Matlab Audio Processing Examples [online] Available: http://www.ee.columbia.edu/%7Edpwe/resources/matlab/ [20/2/2007]

Fant G (1960) Acoustic Theory of Speech Production. The Hague: Mouton

Fellbaum K (2007) Human Speech Production Based on a Linear Predictive Vocoder [online] Available: http://www.kt.tu-cottbus.de/speech-analysis/ [20/2/2007]

Goldberg RG, Riek L (2000) Speech Coders. CRC Press: Boca Raton, FL

Gray RM (2006) Packet speech on the Arpanet: A history of early LPC speech and its accidental impact on the Internet Protocol [online] Available: http://www.ieee.org/organizations/society/sp/ Packet_Speech.pdf [20/2/2007]

Hess W (1992) Pitch and Voicing Determination. In: Advances in Speech Signal Processing, S. Furui, M. Sondhi, eds., Dekker, New York, pp 3–48

Khan A, Kashif F (2003) Speech Coding with Linear Predictive Coding (LPC) [online] Available: http://www.dspexperts.com/dsp/projects/lpc [20/2/2007]

Kroon P, Deprettere E, Sluyter R (1986) Regular-pulse excitation – A novel approach to effective and efficient multipulse coding of speech. IEEE Transactions on Acoustics, Speech, and Signal Processing 34(5): 1054–1063

Matsumoto J, Nishiguchi M, Iijima K (1997) Harmonic Vector Excitation Coding at 2.0 kbps. In: Proc. IEEE Workshop on Speech Coding, pp 39–40

McCree AV, Barnwell TP (1995) A mixed excitation LPC vocoder model for low bit rate speech coding. IEEE Transactions on Speech and Audio Processing, 3(4):242–250

NATO (1984) Parameters and coding characteristics that must be common to assure interoperability of 2400 bps linear predictive encoded speech. NATO Standard STANAG-4198-Ed1

Quatieri T (2002) Discrete-Time Speech Signal Processing: Principles and Practice. Prentice-Hall, Inc.: Upper Saddle River, NJ

Rabiner LR, Schafer RW (1978) Digital Processing of Speech Signals. Prentice-Hall, Inc.: Englewood Cliffs, NJ

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

Schroeder MR, Atal B (1985) Code-Excited Linear Prediction(CELP): High Quality Speech at Very Low Bit Rates. In: Proc. IEEE ICASSP-85, pp 937–940

Spanias A, Painter T (2002) Matlab simulation of LPC10e vocoder [online] Available: http://www.cysip.net/lpc10e_FORM.htm [19/2/2007]

Woodard J (2007) Speech coding [online] Available: http://www-mobile.ecs.soton.ac.uk/speech_codecs/ [20/2/2007]

Chapter 2

How are bits played back from an audio CD?

T. Dutoit (°), R. Schreier (*)

(°) Faculté Polytechnique de Mons, Belgium
(*) Analog Devices, Inc., Toronto, Canada

An audio digital-to-analog converter adds noise to the signal, by requantizing 16-bit samples to one-bit. It does it...on purpose.

Loading a CD player with one's favorite CD has become an ordinary action. It is taken for granted that the stream of 16-bit digital information it contains can easily be made available to our ears, i.e., in the analog world in which we live. The essential tool for this is the digital-to-analog converter (DAC).

In this chapter we will see that, contrary to what might be expected, many audio DACs (including those used in CD and MP3 players, for instance, or in cell phones) first requantize the 16-bit stream into a *1-bit* stream[1] with very high sampling frequency, using a signal processing concept known as delta–sigma[2] (ΔΣ) modulation, and then convert the resulting bipolar signal back to an audio waveform.

The same technique is used in ADCs for digitizing analog waveforms. It is also the heart of the direct stream digital (DSD) encoding system implemented in super audio CDs (SACDs).

---

[1] In practice, 1-bit ΔΣ DACs have been superseded by multiple-bit DACs, but the principle remains the same.

[2] Sometimes also referred to as *sigma–delta.*

T. Dutoit, F. Marqués (eds.), *Applied Signal Processing*,
DOI 10.1007/978-0-387-74535-0_2, © Springer Science+Business Media, LLC 2009