Linear predictive coding

This method combines linear processing with scalar quantization.

The main idea of the method is to predict the value of the current sample by a linear combination of previous already reconstructed samples and then to quantize the difference between the actual value and the predicted value.

Linear prediction coefficients are weighting coefficients used in linear combination.

A simple predictive quantizer or differential pulse-coded modulator is shown in Fig. 5.1.

If the predictor is simply the last sample and the quantizer has only one bit, the system becomes a delta-modulator. It is shown in Fig. 5.2.
Differential pulse-coded modulator

\[ x(nT_s) \rightarrow e(nT_s) \rightarrow \text{QUANTIZER} \rightarrow q(nT_s) \rightarrow \text{DEQUANTIZER} \]

\[ - \hat{x}(nT_s) \]

\[ d(nT_s) \rightarrow \text{LINEAR PREDICTOR} \]

\[ x_R(nT_s) \]
Delta-modulator

\[ x(nT_s) \]}

\[ \Delta \]

\[ -\Delta \]

\[ x(nT_s) \geq \hat{x}(nT_s) \]

\[ x(nT_s) < \hat{x}(nT_s) \]
Linear predictive coding

The main feature of the quantizers shown in Figs 5.1, 5.2 is that they exploit not all advantages of predictive coding.

• **Prediction coefficients** used in these schemes are not optimal

• **Prediction** is based on past reconstructed samples and not true samples

• **Usually coefficients** of prediction are chosen by using some empirical rules and are not transmitted

For example quantizer in Fig.5.1 instead of actual value of error $e$ uses reconstructed values $d$ and instead of true sample values $x$ their estimates $x_R$ obtained via $d$. 
Linear predictive coding

The most advanced quantizer of linear predictive type represents a basis of the so-called Code Excited Linear Predictive (CELP) coder.

• It uses the optimal set of coefficients or in other words linear prediction coefficients of this quantizer are determined by minimizing the MSE between the current sample and its predicted value.

• It is based on the original past samples

• Using the true samples for prediction requires the “looking-ahead” procedure in the coder.

• The predictor coefficients are transmitted
Linear predictive coding

Assume that quantizer coefficients are optimized for each sample and that the original past samples are used for prediction. Let

\[ x(T_s), x(2T_s), \ldots \]

be a sequence of samples at the quantizer input. Then each sample \( x(nT_s) \) is predicted by the previous samples according to the formula

\[
\hat{x}(nT_s) = \sum_{k=1}^{m} a_k x(nT_s - kT_s), \quad (5.1)
\]

where \( \hat{x}(nT_s) \) is the predicted value, \( a_k \) are prediction coefficients, \( m \) denotes the order of prediction. The prediction error is

\[
e(nT_s) = x(nT_s) - \hat{x}(nT_s).
\]
Linear predictive coding

Prediction coefficients are determined by minimizing the sum of squared errors over a given interval

\[ E = \sum_{n=n_0}^{n_1} e^2(nT_s) \quad (5.2) \]

Inserting (5.1) into (5.2) we obtain

\[ E = \sum_{n=n_0}^{n_1} (x(nT_s) - a_1 x(nT_s - T_s) - \ldots - a_m x(nT_s - mT_s))^2 = \]

\[ = \sum_{n=n_0}^{n_1} \{x(nT_s)\}^2 - 2 \sum_{j=1}^{m} a_j \sum_{n=n_0}^{n_1} x(nT_s) x(nT_s - jT_s) + \]

\[ + \sum_{j=1}^{m} \sum_{k=1}^{m} a_j a_k \sum_{n=n_0}^{n_1} x(nT_s - jT_s) x(nT_s - kT_s). \quad (5.3) \]
Linear predictive coding

Differentiating (5.3) over $a_k$, $k = 1,2,\ldots, m$ yields

$$
\frac{\partial E}{\partial a_k} = \sum_{n=n_0}^{n_1} x(nT_s)x(nT_s - kT_s) - \sum_{j=1}^{m} a_j \sum_{n=n_0}^{n_1} x(nT_s - kT_s)x(nT_s - jT_s) = 0
$$

Thus we obtain a system of $m$ linear equations with $m$ unknown quantities $a_1, a_2, \ldots, a_m$

\[
\sum_{j=1}^{m} a_j c_{jk} = c_{ok}, \quad k = 1,2,\ldots, m \tag{5.4}
\]

where $c_{jk} = c_{kj} = \sum_{n=n_0}^{n_1} x(nT_s - jT_s)x(nT_s - kT_s)$.

\[
\tag{5.5}
\]

The system (5.4) is called the Yule-Walker equations.
Linear predictive coding

If $a_1, a_2, \ldots, a_m$ are solutions of (5.4) then we can find the minimal achievable prediction error. Insert (5.5) into (5.3). We obtain that

$$E = c_{00} - 2 \sum_{k=1}^{m} a_k c_{0k} + \sum_{k=1}^{m} a_k \sum_{j=1}^{m} a_j c_{jk}.$$  (5.6)

Using (5.3) we reduce (5.6) to

$$E = c_{00} - \sum_{k=1}^{m} a_k c_{0k}$$
Interpretation of the Yule-Walker equations like a digital filter

Eq. (5.1) describes the $m$ th order predictor with transfer function equal to

$$P(z) = \frac{\hat{X}(z)}{X(z)} = \sum_{k=1}^{m} a_k z^{-k}.$$  

$z$-transform for the prediction error is

$$E(z) = X(z) - \sum_{k=1}^{m} a_k X(z) z^{-k}.$$  

The prediction error is an output signal of the discrete-time filter with transfer function

$$A(z) = \frac{E(z)}{X(z)} = 1 - \sum_{k=1}^{m} a_k z^{-k}.$$  

The problem of finding the optimal set of prediction coefficients = problem of constructing $m$ th order FIR filter.
Interpretation of the Yule-Walker equations like a digital filter

Another name of the linear prediction (5.1) is the autoregressive model of signal $x(nT_s)$. It is assumed that the signal $x(nT_s)$ can be obtained as the output of the autoregressive filter with transfer function

$$H(z) = \frac{1}{1 - \sum_{k=1}^{m} a_k z^{-k}},$$

that is can be obtained as the output of the filter which is inverse with respect to the prediction filter. This filter is a discrete-time IIR filter.
Methods of finding coefficients

\[ c_{ij}, i = 0, 1, \ldots, m, j = 1, 2, \ldots, m \]

In order to solve the Yule-Walker eq. (5.4) it is necessary first to evaluate values \( c_{ij}, i = 0, 1, \ldots, m, j = 1, 2, \ldots, m \)

There are two approaches to estimating these values:

- The autocorrelation method
- The covariance method.

The complexity of solving (5.4) is proportional to \( m^2 \)

The complexity of solving (5.4) is proportional to \( m^3 \)
Autocorrelation method

The values $c_{ij}$ are computed as

$$c_{ij} = c_{ji} = \sum_{n=i_0}^{i_1} x(nT_s - iT_s)x(nT_s - jT_s). \quad (5.7)$$

We set $i_0 = -\infty, i_1 = \infty$ and $x(nT_s) = 0$ if $n < 0, n \geq N$, where $N$ is called the interval of analysis.

In this case we can simplify (5.7)

$$c_{ij} = \sum_{n=0}^{N-1} x(nT_s - iT_s)x(nT_s - jT_s) = \sum_{n=0}^{N-1-|i-j|} x(nT_s)x(nT_s + |i-j|T_s). \quad (5.8)$$

Normalized by $N$ they coincide with estimates of entries of covariance matrix for $x(nT_s)$

$$\hat{R}(|i - j|) = c_{ij} / N = 1 / N \sum_{n=0}^{N-1-|i-j|} x(nT_s)x(nT_s + |i-j|T_s).$$
Autocorrelation method
Autocorrelation method

The Yule-Walker equations for autocorrelation method have the form

$$\sum_{i=1}^{m} a_i \hat{R}(|i - j|) = \hat{R}(j), \quad j = 1, 2, \ldots, m. \quad (5.9)$$

Eq.(5.9) can be given by matrix equation

$$\mathbf{a} \times \mathbf{R} = \mathbf{b},$$

where  $\mathbf{a} = (a_1, a_2, \ldots, a_m)$,  $\mathbf{b} = (\hat{R}(1), \hat{R}(2), \ldots, \hat{R}(m))$,  

$$\mathbf{R} = \begin{bmatrix} \hat{R}(0) & \hat{R}(1) & \cdots & \hat{R}(m-1) \\ \hat{R}(1) & \hat{R}(0) & \cdots & \hat{R}(m-2) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{R}(m-1) & \hat{R}(m-2) & \cdots & \hat{R}(0) \end{bmatrix}.$$
Autocorrelation method

It is said that (5.9) relates the parameters of the autoregressive model of $m$ th order with the autocorrelation sequence.

Matrix $\mathbf{R}$ of the autocorrelation method has two important properties.

• It is symmetric, that is $\hat{R}(i, j) = \hat{R}(j, i)$

• It has Toeplitz property, that is $\hat{R}(i, j) = \hat{R}(|i - j|)$.

The Toeplitz property of $\mathbf{R}$ makes it possible to reduce the computational complexity of solving (5.4). The fast Levinson-Durbin recursive algorithm requires only $m^2$ operations.
Covariance method

We choose \( i_0 = 0 \) and \( i_1 = N - 1 \) and signal \( x(nT_s) \) is not constrained in time. In this case we have

\[
c_{ij} = \sum_{n=0}^{N-1} x(nT_s - iT_s)x(nT_s - jT_s).
\] (5.10)

Set \( k = n - i \) then (5.10) can be rewritten as

\[
c_{ij} = \sum_{k=-i}^{N-i-1} x(kT_s)x(kT_s + (i - j)T_s), i = 1,..., m, j = 0,...m.
\] (5.11)

(5.11) resembles (5.8) but it has other range of definition for \( k \).

- It uses signal values out of range \( 0 \leq k \leq N - 1 \),
- The method leads to the cross-correlation function between two similar but not exactly the same finite segments of \( x(kT_s) \).
Covariance method

\[ \hat{R}(i, j) = c_{ij} / N = 1 / N \sum_{n=0}^{N-1} x((n - i)T_s)x((n - j)T_s). \]

The Yule-Walker equations for the covariation method are

\[ \sum_{i=1}^{m} a_i \hat{R}(i, j) = \hat{R}(0, j), \quad j = 1, 2, ..., m. \quad (5.12) \]

Eq. (5.12) can be given by the matrix equation

\[ \mathbf{a} \times \mathbf{P} = \mathbf{c}, \]

where \( \mathbf{a} = (a_1, a_2, ..., a_m), \mathbf{c} = (\hat{R}(0,1), \hat{R}(0,2), ..., \hat{R}(0, m)), \)

\[ \mathbf{P} = \begin{bmatrix} \hat{R}(1,1) & \hat{R}(1,2) & \ldots & \hat{R}(1,m) \\ \hat{R}(2,1) & \hat{R}(2,2) & \ldots & \hat{R}(2,m) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{R}(m,1) & \hat{R}(m,2) & \ldots & \hat{R}(m,m) \end{bmatrix}. \]
Covariance method

Unlike the matrix $R$ of autocorrelation method the matrix $P$ is symmetric ($\hat{R}(i, j) = \hat{R}(j, i)$) but it is not Toeplitz.

Since computational complexity of solving an arbitrary system of linear equations of order $m$ is equal to $m^3$, then in this case to solve (5.12) it is necessary $m^3$ operations.
Algorithms for the solution of the Yule-Walker equations

The computational complexity of solving the Yule-Walker equations depends on the method of evaluating values $c_{ij}$.

Assume that $c_{ij}$ are found by the autocorrelation method.

In this case the Yule-Walker equations has the form (5.9) and the matrix $R$ is symmetric and the Toeplitz matrix.

These properties make it possible to find the solution of (5.9) by fast methods requiring $m^2$ operations.

There are a few methods of this type: the Levinson-Durbin algorithm, the Euclidean algorithm and the Berlekamp-Massey algorithm.
The Levinson-Durbin algorithm

It was suggested by Levinson in 1948 and then was improved by Durbin in 1960. Notice that this algorithm works efficiently if matrix of coefficients $R$ is simultaneously symmetric and Toeplitz. The Berlekamp-Massey and the Euclidean algorithms do not require the matrix of coefficients to be symmetric.

We sequentially solve equations (5.9) of order $l = 1, \ldots, m$. Let $\mathbf{a}^{(l)} = (a_1^{(l)}, a_2^{(l)}, \ldots, a_l^{(l)})$ denote the solution for the system of the $l$th order. Given $\mathbf{a}^{(l)}$ we find the solution for the $(l+1)$th order. At each step of the algorithm we evaluate the prediction error $E_l$ of the $l$th order system.
The Levinson-Durbin algorithm

Initialization: \( l = 0, E_0 = \hat{R}(0), a^{(0)} = 0. \)

Recurrent procedure:

For \( l = 1, \ldots, m \) compute

\[
a^{(l)}_l = (\hat{R}(l) - \sum_{i=1}^{l-1} a^{(l-1)}_i \hat{R}(l-i)) / E_{l-1},
\]

\[
a^{(l)}_j = a^{(l-1)}_j - a^{(l)}_l a^{(l-1)}_{l-j}, \quad 1 \leq j \leq l - 1,
\]

\[
E_l = E_{l-1} (1 - (a^{(l)}_l)^2).
\]

When \( l = m \) we obtain the solution

\[
a = (a_1, a_2, \ldots, a_m) = a^{(m)}, E = E_m.
\]
Example

\[
m = 1 \quad R(0)a_1^{(1)} = R(1), \quad k_1 = -R(1) / R(0),
\]
\[
a_1^{(1)} = R(1) / R(0),
\]
\[
E_1 = R(0) - a_1^{(1)} R(1)
\]
\[
E_1 = (R^2(0) - R^{(2)}(1)) / R(0).
\]

\[
m = 2 \quad \begin{cases} 
R(0)a_1^{(2)} + R(1)a_2^{(2)} = R(1) \\
R(1)a_1^{(2)} + R(0)a_2^{(2)} = R(2),
\end{cases}
\]
\[
a_1^{(2)} = \frac{R(1)}{R(0)} (1 - a_2^{(2)}) = a_1^{(1)} - a_2^{(2)} a_1^{(1)}
\]
\[
a_2^{(2)} = \frac{R(2) - R(1)a_1^{(1)}}{R(0) - a_1^{(1)} R(1)} = \frac{R(2) - R(1)a_1^{(1)}}{E_1}
\]
Example

\[ a_1^{(2)} = \frac{R(1)R(0) - R(1)R(2)}{R^2(0) - R^2(1)} \]

\[ a_2^{(2)} = \frac{R(0)R(2) - R(1)^2}{R^2(0) - R^{(2)}(1)}. \]

\[ E_2 = E_1 (1 - (a_2^{(2)})^2) \]

\[ E_2 = R(0) - a_1^{(2)} R(1) - a_2^{(2)} R(2) = (R(0) - a_1^{(1)} R(1))(1 - (a_2^{(2)})^2) = \]

\[ = E_1 (1 - (a_2^{(2)})^2). \]
Linear prediction analysis-by-synthesis (LPAS) coders

The most popular class of speech coders for bit rates between 4.8 and 16 kb/s are model-based coders that use an LPAS method.

A linear prediction model of speech production (adaptive linear prediction filter) is excited by an appropriate excitation signal in order to model the signal over time. The parameters of both the filter and the excitation are estimated and updated at regular time intervals (frames). The compressed speech file contains these model parameters estimated for each frame.

Each sound corresponds to a set of filter coefficients. Rather often this filter is also represented by the poles of its frequency response called formant frequencies or formants.
Model of speech generation

- Generator of periodical impulse train
- Random noise generator
- Pitch period
- Tone/noise
- $u(n)$
- $g$
- Adaptive linear prediction filter
- Original speech
- Synthesized speech
LPAS coders

The filter excitation depends on the type of the sound: voiced, unvoiced, vowel, hissing or nasal. The voiced sounds are generated by oscillation of vocal cords and represent a quasi-periodical impulse train. The unvoiced signals are generated by noise-like signals.

The period of vocal cords oscillation is called pitch period.

CELP coder is Code Excited Linear Predictive coder. It is a basis of all LPAS coders (G.729, G.723.1, G.728, IS-54, IS-96, RPE-LTP(GSM), FS-1016(CELP)).
CELP Standard

Stochastic codebook

Linear Prediction analysis

Interpolate by 4

LP filter

PW filter

Minimize PE

Input speech (8kHz sampling)

LSP

Adaptive codebook updating

Sync. bits

PE

g_a

i_s

g_s

i_a

i_s

g_s

i_a

g_a

256

512

...
CELP Standard

Main ideas:

• A 10\textsuperscript{th} order LP filter is used to model the speech signal short term spectrum, or formant structure.

• Long-term signal periodicity or pitch is modeled by an adaptive codebook VQ.

• The residual from the short-term LP and pitch VQ is vector quantized using a fixed stochastic codebook.

• The optimal scaled excitation vectors from the adaptive and stochastic codebooks are selected by minimizing a time-varying, perceptually weighted distorted measure that improves subjective speech quality by exploiting masking properties of human hearing.
CELP Standard

CELP uses input signals at sampling rate 8 kHz and 30 ms (240 samples) frame size with 4 subframes of size 7.5 ms (60 samples.)

• Short-term prediction

It is performed once per frame by open-loop analysis. We construct 10th order prediction filter using the autocorrelation method and the Levinson-Durbin procedure. The LP filter is represented by its linear spectral pairs (LPS) which are functions of the filter formants. The 10 LSPs are coded using 34-bit nonuniform scalar quantization. Because the LSPs are transmitted once per frame, but are needed for each subframe, they are linearly interpolated to form an intermediate set for each of the four subframes.
For the first frame we obtain:

Coefficients of the Yule-Walker equations (m=10) are:

1.0, 0.75, 0.17, -0.38, -0.65, -0.59, -0.35, -0.08, 0.17, 0.39, 0.52

$R(0) = 3.79 \times 10^7$

The prediction filter coefficients are:

1.989, -1.734, 0.412, 0.096, 0.128, -0.084, -0.378, 0.303, 0.131, -0.166

The prediction error is: $1.07 \times 10^6$
Linear spectral parameters

Let $H(z)$ be the transfer function of the inverse filter, that is $H(z) = 1/A(z)$, where $A(z)$ is the prediction filter. Then the frequencies $\omega_i, i = 1, 2..., m$ which correspond to the poles of function $H(z)$ or zeros of $A(z)$, are called formant frequencies or formants.

LSPs are functions of formant frequencies. They can be found using the algorithm shown in Fig.10.1
Amplitude function of the prediction filter

$A(\omega)$
Amplitude function of the inverse filter

$$H(\omega)$$
LSP

\[ m = 2 \]

\[ A(z) = A(2)z^{-2} + A(1)z^{-1} + 1 \]

\[ P(z) = 1 + (A(2) + A(1))z^{-1} + (A(2) + A(1))z^{-2} + z^{-3} \]

\[ Q(z) = 1 + (A(1) - A(2))z^{-1} - (A(1) - A(2))z^{-2} - z^{-3} \]

\[ PL(z) = P(z)/(z^{-1} + 1) = 1 + (A(1) + A(2) - 1)z^{-1} + z^{-2} \]

\[ QL(z) = Q(z)/(z^{-1} - 1) = 1 + (A(1) - A(2) + 1)z^{-1} + z^{-2} \]

\[ P^*(z) = (A(2) + A(1) - 1)/2 + z^{-1} \]

\[ Q^*(z) = (A(1) - A(2) + 1)/2 + z^{-1} \]

\[ z_p^{-1} = -(A(2) + A(1) - 1)/2 \]

\[ z_q^{-1} = -(A(1) - A(2) + 1)/2 \]

\[ \omega_p = \arccos(z_p^{-1}) \]

\[ \omega_q = \arccos(z_q^{-1}) \]
<table>
<thead>
<tr>
<th>LSP</th>
<th>Bits</th>
<th>Output Levels (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>100,170,225,250,280,340,420,500</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>210,235,265,295,325,360,400,440,480,520,560,610,670,740,810,880</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>420,460,500,540,585,640,705,775,850,950,1050,1150,1250,1350,1450,1550</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>620,660,720,795,880,970,1080,1170,1270,1370,1470,1570,1670,1770,1870,1970</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1000,1050,1130,1210,1285,1350,1430,1510,1590,1670,1750,1850,1950,2050,2150,2250</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1470,1570,1690,1830,2000,2200,2400,2600</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1800,1880,1960,2100,2300,2480,2700,2900</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>3190,3270,3350,3420,3490,3590,3710,3830</td>
</tr>
</tbody>
</table>
CELP

• Long-term signal periodicity is modeled by an adaptive codebook VQ. The adaptive codebook search is performed by closed-loop analysis using a modified minimum squared prediction error (MPSE) criterion of the perceptually weighted error signal.

The adaptive codebook contains 256 codewords. Each codeword is constructed by the previous excitation signal of length 60 samples delayed by $20 \leq M \leq 147$ samples.

For delays less than the subframe length (60 samples) the codewords contain the initial $M$ samples of the previous excitation vector. To complete the codeword to 60 elements, the short vector is replicated by periodic extension.
Adaptive codebook

<table>
<thead>
<tr>
<th>Index</th>
<th>Delay</th>
<th>Adaptive CB sample numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>147</td>
<td>-147, -146, -145, …, -89, -88</td>
</tr>
<tr>
<td>254</td>
<td>146</td>
<td>-146, -145, -144, …, -88, -87</td>
</tr>
<tr>
<td>253</td>
<td>145</td>
<td>-145, -144, -143, …, -87, -86</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>131</td>
<td>61</td>
<td>-61, -60, -59, …, -2, -1</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>-21, …, -2, -21, …, -2, -21, …, -2</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>-20, …, -1, -20, …, -1, -20, …, -1</td>
</tr>
</tbody>
</table>
Adaptive codebook
Adaptive codebook

To find the best excitation in the adaptive codebook the 1st order linear prediction is used. Let \( \mathbf{s} \) be the original speech vector and \( \mathbf{a}_i \) be a filtered codeword \( \mathbf{c}_i \) from the adaptive codebook then we search for

\[
\min_i \| \mathbf{s} - g_a \mathbf{a}_i \|_2^2 = \min_i \left\{ \| \mathbf{s} \|_2^2 - 2 g_a (\mathbf{s}, \mathbf{a}_i) + g_a^2 \| \mathbf{a}_i \|_2^2 \right\} \tag{10.1}
\]

where \( g_a \) is a prediction coefficient or the adaptive codebook gain. By taking derivative with respect to \( g_a \) and setting it to zero we find the optimal gain:

\[
g_a = \frac{(\mathbf{s}, \mathbf{a}_i)}{\| \mathbf{a}_i \|_2^2} \tag{10.2}
\]

Inserting (10.2) into (10.1) we obtain

\[
\min_i \| \mathbf{s} - g_a \mathbf{a}_i \|_2^2 = \min_i \left\{ \| \mathbf{s} \|_2^2 - \frac{2(\mathbf{s}, \mathbf{a}_i)^2}{\| \mathbf{a}_i \|_2^2} + \frac{(\mathbf{s}, \mathbf{a}_i)^2}{\| \mathbf{a}_i \|_2^2} \right\} = \min_i \left\{ \| \mathbf{s} \|_2^2 - \frac{(\mathbf{s}, \mathbf{a}_i)^2}{\| \mathbf{a}_i \|_2^2} \right\} \tag{10.3}
\]
Adaptive codebook

Minimizing (10.3) over $i$ is equivalent to maximizing the last term in (10.3) since the first term is independent of the codeword $a_i$. Thus the adaptive codebook search procedure finds codeword $c_i$ which maximizes the so-called match function $m_i$

$$m_i = \frac{(s, a_i)^2}{\|a_i\|^2}$$

The adaptive codebook index $i_a$ and gain $g_a$ are transmitted four times per frame (every 7.5 ms). The gain is coded between $-1$ and $+2$ using nonuniform, scalar, 5 bit quantization.
## Adaptive codebook

<table>
<thead>
<tr>
<th>-.993</th>
<th>-.831</th>
<th>-.693</th>
<th>-.555</th>
<th>-.414</th>
<th>-.229</th>
<th>0.00</th>
<th>0.139</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.255</td>
<td>0.368</td>
<td>0.457</td>
<td>0.531</td>
<td>0.601</td>
<td>0.653</td>
<td>0.702</td>
<td>0.745</td>
</tr>
<tr>
<td>0.780</td>
<td>0.816</td>
<td>0.850</td>
<td>0.881</td>
<td>0.915</td>
<td>0.948</td>
<td>0.983</td>
<td>1.02</td>
</tr>
<tr>
<td>1.062</td>
<td>1.117</td>
<td>1.193</td>
<td>1.289</td>
<td>1.394</td>
<td>1.540</td>
<td>1.765</td>
<td>1.991</td>
</tr>
</tbody>
</table>

Gain encoding levels
Stochastic codebook

The stochastic codebook (SC) contains 512 codewords. The special form of SC represents ternary quantized (-1,0,+1) Gaussian sequences of length 60.

The stochastic codebook search target is the original speech vector minus the filtered adaptive codebook excitation, that is, \( \mathbf{u} = \mathbf{s} - g_a \mathbf{a}_{opt} \).

The SC search is performed by closed-loop analysis using conventional MSPE criterion. We find such a codeword \( \mathbf{x}_i \) which maximizes the following match function

\[
\frac{(\mathbf{u}, \mathbf{y}_i)^2}{\|\mathbf{y}_i\|^2},
\]

where \( \mathbf{y}_i \) is the filtered codeword \( \mathbf{x}_i \).
Stochastic codebook

The stochastic codebook index and gain are transmitted four times per frame. The gain (positive and negative) is coded using 5-bit nonuniform scalar quantization.

The weighted sum of the found optimal codeword from the adaptive codebook and the optimal codeword from the stochastic codebook are used to update the adaptive codebook. It means that the most distant past samples are removed from the adaptive codebook, all codewords are shifted and the following new samples are placed as the first

\[ c_{i_a} g_a + x_{i_s} g_s \]
**Stochastic codebook**

<table>
<thead>
<tr>
<th>-1330</th>
<th>-870</th>
<th>-660</th>
<th>-520</th>
<th>-418</th>
<th>-340</th>
<th>-278</th>
<th>-224</th>
</tr>
</thead>
<tbody>
<tr>
<td>-178</td>
<td>-136</td>
<td>-98</td>
<td>-64</td>
<td>-35</td>
<td>-13</td>
<td>-3</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>13</td>
<td>35</td>
<td>64</td>
<td>98</td>
<td>136</td>
<td>178</td>
</tr>
<tr>
<td>224</td>
<td>278</td>
<td>340</td>
<td>418</td>
<td>520</td>
<td>660</td>
<td>870</td>
<td>1330</td>
</tr>
</tbody>
</table>

**Stochastic codebook gain encoding levels**
CELP

The total number of bits per frame can be computed as

\[ 4(b_{g_s} + b_{i_s} + b_{g_a} + b_{i_a}) + b_{LSP} = 4(5 + 8 + 5 + 9) + 34 = 142, \]

where \( b_{g_s}, b_{i_s}, b_{g_a}, b_{i_a} \) are numbers of bits for index and gain of the stochastic and adaptive codebook, respectively, \( b_{LSP} \) is the number of bits for linear spectral pairs.

Taking into account that a frame duration is 30 ms we obtain that the bit rate of the CELP coder is equal to

\[ R = \frac{142}{30 \cdot 10^{-3}} \approx 4733 \text{ b/s.} \]

Adding bits for synchronization and correcting errors we get that bit rate is equal to 4800 b/s.
CELP

The critical point of the standard is computational complexity of two codebook searches.

The number of multiplications for adaptive codebook search can be estimated as

$$(60 \times 10 + 60) \times 256 \approx 1.7 \times 10^5.$$ 

The stochastic codebook search requires

$$(60 \times 10 + 60) \times 512 \approx 3.4 \times 10^5$$

multiplications.
Speech coding for multimedia applications

The most important attributes of speech coding are:

• **Bit rate.** From 2.4 kb/s for secure telephony to 64 kb/s for network applications.

• **Delay.** For network applications delay below 150ms is required. For multimedia storage applications the coder can have unlimited delay.

• **Complexity.** Coders can be implemented on PC or on DSP chips. Measures of complexity for a DSP or a CPU are different. Also complexity depends on DSP architecture. It is usually expressed in MIPS.

• **Quality.** For secure telephony quality is synonymous with intelligibility. For network applications the goal is to preserve naturalness and subjective quality.
Standardization

- International Telecommunication Union (ITU)
- International Standards Organization (ISO)
- Telecommunication Industry Association (TIA), NA
- R&D Center for Radio systems (RCR), Japan
<table>
<thead>
<tr>
<th>Standard</th>
<th>Bit rate</th>
<th>Frame size / look-ahead</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ITU standards</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711 PCM</td>
<td>64 kb/s</td>
<td>0/0</td>
<td>0.01 MIPS</td>
</tr>
<tr>
<td>G.726 , G.727 ADPCM</td>
<td>16, 24, 32, 40 kb/s</td>
<td>0.125 ms/0</td>
<td>2 MIPS</td>
</tr>
<tr>
<td>G.722 Wideband coder</td>
<td>48, 56, 64 kb/s</td>
<td>0.125/1.5 ms</td>
<td>5 MIPS</td>
</tr>
<tr>
<td>G.728 LD-CELP</td>
<td>16 kb/s</td>
<td>0.625 ms/0</td>
<td>30 MIPS</td>
</tr>
<tr>
<td>G.729 CS-ACELP</td>
<td>8 kb/s</td>
<td>10/5 ms</td>
<td>20 MIPS</td>
</tr>
<tr>
<td>G.723.1 MPC-MLQ</td>
<td>5.3 &amp; 6.4 kb/s</td>
<td>30/7.5 ms</td>
<td>16 MIPS</td>
</tr>
<tr>
<td>G.729 CS-ACELP annex A</td>
<td>8 kb/s</td>
<td>10/5 ms</td>
<td>11 MIPS</td>
</tr>
<tr>
<td><strong>Cellular standards</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPE-LTP(GSM)</td>
<td>13 kb/s</td>
<td>20 ms/0</td>
<td>5 MIPS</td>
</tr>
<tr>
<td>IS-54 VSELP (TIA)</td>
<td>7.95 kb/s</td>
<td>20/5 ms</td>
<td>15 MIPS</td>
</tr>
<tr>
<td>PDC VSELP( RCR)</td>
<td>6.7 kb/s</td>
<td>20/5 ms</td>
<td>15 MIPS</td>
</tr>
<tr>
<td>IS-96 QCELP (TIA)</td>
<td>8.5/4/2/0.8 kb/s</td>
<td>20/5 ms</td>
<td>15 MIPS</td>
</tr>
<tr>
<td>PDC PSI-CELP (RCR)</td>
<td>3.45 kb/s</td>
<td>40/10 ms</td>
<td>40 MIPS</td>
</tr>
<tr>
<td><strong>U.S. secure telephony standards</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS-1015 LPC-10E</td>
<td>2.4 kb/s</td>
<td>22.5/90 ms</td>
<td>20 MIPS</td>
</tr>
<tr>
<td>FS-1016 CELP</td>
<td>4.8 kb/s</td>
<td>30/30 ms</td>
<td>20 MIPS</td>
</tr>
<tr>
<td>MELP</td>
<td>2.4 kb/s</td>
<td>22.5/20 ms</td>
<td>40 MIPS</td>
</tr>
</tbody>
</table>
History of speech coding

• 40s – PCM was invented and intensively developed in 60s

• 50s – delta modulation and differential PCM were proposed

• 1957 – $\mu$-law encoding was proposed (standardised for telephone networks in 1972 (G.711))

• 1974 – ADPCM was developed (1976 – standards G.726 and G.727)

• 1984 – CELP coder was proposed (majority of coding standards for speech coding today use a variation of CELP)

• 1995 – MELP coder was invented
Subjective quality of speech coders
Speech coders. Demo.

Original file 8kHz, 16 bits, file size is 73684 bytes

A-law 8 kHz, 8 bits, file size is 36842 bytes

ADPCM, 8 kHz, 4 bits, file size is 18688 bytes

GSM, file size is 7540 bytes

CELP-like coder, file size is 4896 bytes

MELP-like coder, file size 920 bytes

MELP-like coder, file size 690 bytes

MELP-like coder, file size 460 bytes
Direct sample-by-sample quantization: Standard G.711

The G.711 PCM coder is designed for telephone bandwidth speech signals. The input signal is speech signal sampled with rate 8 kHz. The coder does direct sample-by-sample quantization. Instead of uniform quantization one form of nonuniform quantization known as companding is used.

The name of the method is derived from the words “compressing-expanding”.

First the original signal is compressed using a memoryless nonlinear device.

The compressed signal is then uniformly quantized.

The decoded waveform must be expanded using a nonlinear function which is the inverse of that used in compression.
Standard G.711

Compression amplifier $F(x)$  →  Uniform quantizer  →  Uniform dequantizer  →  Expansion amplifier $F^{-1}(x)$

$F(x)$

$x$
Standard G.711

Companding is equivalent to quantizing with steps that start out small and get larger for higher signal levels.

There are different standards for companding.

North America and Japan have adopted a standard compression curve known as a $\mu$-law companding. Europe has adopted a different, but similar, standard known as A-law companding. The $\mu$ -law compression formula is

$$F(s) = F_{\text{max}} \, \text{sgn}(s) \frac{\ln(1 + \mu |s / s_{\text{max}}|)}{\ln(1 + \mu)},$$

where $s$ is the input sample value. G.711 uses $\mu = 255$. 
Implementation of G.711
Implementation of G.711

The $\mu = 255$ curve is approximated by a piecewise linear curve. When using this law in networks the suppression of the all zero character signal is required.

The positive portion of the curve is approximated by 8 straight line elements. We divide the positive output region into 8 equal segments. The input region is divided into 8 corresponding nonequal segments.

To identify which segment the sample lies we spend 3 bits.

The value of sample in each segment is quantized to 4 bits number.

1 bit shows the polarity of the sample.

In total we spend 4+3+1 bits for each sample.
Example

In order to quantize the value 66 we spend 1 bit for sign,
The number of the segment is 010,
The quantization level is 0000 (values 65,66,67,68 are quantized to 33).
At the decoder using the codeword 10100000 we reconstruct the approximating value \((64+68)/2=66\).
The same approximating value will be reconstructed instead of values 65,66,67,68.
Each segment of the input axis is twice as wide as the segment to its left. The values 129,130,131,132,133,134,135,136 are quantized to the value 49. The resolution of each next segment is twice as bad as of the previous one.
ADPCM coders: Standards G.726, G.727

Coders of this type are based on linear prediction method. The main feature of these coders is that they use nonoptimal prediction coefficients and prediction is based on past reconstructed samples. These coders provides rather low delay and have low computational complexity as well.

Delta-modulation is a simple technique for reducing the dynamic range of the numbers to be coded. Instead of sending each sample value, we send the difference between sample and a value of a staircase approximation function.

The staircase approximation can only either increase or decrease by step $\Delta$ at each sample point.
Delta-modulation

$x(n), s(n)$

$\Delta$
Granular noise
Slope overload

\[ x(n), s(n) \]

\begin{axis}
    \addplot [solid] coordinates {(0,-60) (10,-50) (20,-40) (30,-30) (40,0) (50,50) (60,40)};
    \addplot [dashed] coordinates {(0,-60) (10,-50) (20,-40) (30,-30) (40,0) (50,50) (60,40)};
\end{axis}
Delta-modulation

The choice of step size $\Delta$ and sampling rate is important. If steps are too small we obtain a slope overload condition where the staircase cannot trace fast changes in the input signal.

If the steps are too large, considerable overshoot will occur during periods when the signal is changing slowly. In that case we have significant quantization noise, known as granular noise.

Adaptive delta-modulation is a scheme which permits adjustment of the step size depending upon the characteristics of the input signal.
Adaptive delta-modulation

The idea of step size adaptation is the following:

If output bit stream contains almost equal number of 1’s and 0’s we assume that the staircase is oscillating about slowly varying analog signal and reduce the step size.

An excess of either 1’s or 0’s within an output bit stream indicates that staircase is trying to catch up with the function. In such cases we increase the step size.

Usually delta-modulators require sampling rate greater than the Nyquist rate.

They provide compression ratios 2-3 times.
G.726, G.727

The speech coders G.726 and G.727 are adaptive differential pulse-coded modulation (ADPCM) coders for telephone bandwidth speech. The input signal is 64 kb/s PCM speech signal (sampling rate 8 kHz and each sample is 8 bit integer number).

The format conversion block converts the input signal $x(n)$ from A-law or $\mu$-law PCM to a uniform PCM signal $x_u(n)$. The difference $d(n) = x_u(n) - x_p(n)$, here $x_p(n)$ is the predicted signal, is quantized. A 32-, 16-, 8- or 4 level non-uniform quantizer is used for operating at 40, 32, 24 or 16 kb/s, respectively. Prior to quantization $d(n)$ is converted and scaled

$$l(n) = \log_2 |d(n)| - y(n),$$

where $y(n)$ is computed by the scale factor adaptation block.
G726, G727

Input PCM format conversion

Adaptive quantizer

Inverse adaptive quantizer

Quantizer scale factor adaptation

$x(n)$

$x_u(n)$  $d(n)$

$q(n)$  $y(n)$

$-x_p(n)$

$Adaptive predictor$

$Adaptive quantizer$

$d_q(n)$  $x_r(n)$
The value $l(n)$ is then scalar quantized with a given quantization rate.

For bit rate 16 kb/s $|l(n)|$ is quantized with $R = 1$ bit/sample and one more bit is used to specify the sign. Two quantization intervals are: $(-\infty, 2.04)$ and $(2.04, \infty)$ They contain the approximating values 0.91 and 2.85, respectively.

A linear prediction is based on the two previous samples of the reconstructed signal $x_r(n) = x_p(n) + d_q(n)$ and the six previous samples of the reconstructed difference $d_q(n)$:

$$x_p(n) = \sum_{i=1}^{2} a_i (n-1) x_r(n-i) + e(n),$$

$$e(n) = \sum_{i=1}^{6} b_i (n-1) d_q(n-i).$$
G.726, G.727

The predictor coefficients as well as the scale factor are updated on sample-by-sample basis in a backward adaptive fashion. For the second order predictor:

\[ a_1(n) = (1 - 2^{-8})a_1(n-1) + (3 \cdot 2^{-8}) \text{sgn}(p(n)) \text{sgn}(p(n-1)) \]

\[ a_2(n) = (1 - 2^{-7})a_2(n-1) + 2^{-7} \{ \text{sgn}(p(n)) \text{sgn}(p(n-2)) \} - 2^{-7} f \{ a_1(n-1) \} \text{sgn}(p(n)) \text{sgn}(p(n-1)), \]

\[ p(n) = d_q(n) + e(n) \]

For the sixth order predictor:

\[ b_i(n) = (1 - 2^{-8})b_i(n-1) + 2^{-7} \text{sgn}(d_q(n)) \text{sgn}(d_q(n-i)), \]

\[ i = 1, 2, \ldots, 6. \]