2.2 The Structure of Analysis-by-Synthesis Codecs

A Block diagram of a synthesis-by-analysis codec is shown in the figure 1. Main functional parts are:

- Preprocessing.
- LPC-analysis, quantization and interpolation of LPC-coefficients.
- Determinization of the adaptive code vector which is the delay of the long-term predictor.
- Excitation search from fixed codebook.
- Weighting filter applying psychoacoustic masking.
- Postfiltering.

A detailed discussion follows later on. The basic idea is to search parameters: LP-coefficients, long-term predictor and excitation for each frame of speech. The parameters are chosen to minimize perceptually weighted error between the synthesized signal and the original.

![Figure 1](image.png)

**Figure 1:** A block diagram of a synthesis-by-analysis codec. The output of the codec is obtained by summing the excitations from the codebooks and by synthesizing it by the synthesis filter. To the decoder will be sent: quantized LP-coefficients, codebook indices and gains. In the picture AC is the adaptive codebook and FC the fixed codebook.

The biggest difference from multi-pulse codec is the adaptive codebook. Adaptive codebook is particularly useful in voiced sections of speech where the adaptive codebook delay is the pitch delay.
Perhaps the most crucial part of the synthesis-by-analysis codec is the fixed codebook. Most of the bits in the bitstream are consumed by the fixed codebook index and also the quality of speech is sensitive to the structure of the codebook.

Moreover the fixed codebook search is the most computationally complex operation in the codec. During the years of development of speech coding methods, a big bunch of different structures of the fixed codebook have been proposed of which a few will be explained.

## 2.3 Vector Quantization

Vector quantization refers to quantization of blocks (vectors) more than one sample at a time. Claude Shannon proved in forties that by quantizing more than one sample at a time yields always better results than quantization of the samples separately even if the samples were independent of each other. Deeper study of this phenomenon belongs to information theory, however intuitive explanation can be given already in dimension two.

Any quantizer maps the samples into values in a finite set of quantization points. Generally the coding maps the samples to the closest quantization points (in euclidean distance in this case, but can be any other suitable metric). There is always some amount of error between the quantization points and the original sample. The error depends on how close the quantization points are from the samples.

If two independent samples are quantized with two consecutive PCM-quantizers, the quantization points in the 2D plane are as in the figure 2. There is better way to distribute the quantization points on the plane to minimize the average euclidian distance. How? For instance so that the points are the points generated by an isosceles triangle as shown in 2.

## 2.4 Preprocessing and Windowing

The purpose of preprocessing is among other things to guarantee that the frequency band of the signal matches to that for which the codec is optimized. Typically this comprises removal of low frequencies by highpass filter and scaling of the signal to prevent overflows in the coming parts implemented with fixed precision DSP circuits. Also preprocessing can equalize the spectrum to compensate the use of possibly cheap low-quality microphones. Moreover noise removal should be implemented in the preprocessing stage to improve compressibility of the signal.

Typical highpass filter is the first degree FIR

\[ H(z) = 1 - 0.95z^{-1}, \]

having the amplitude response in the figure 3. To equalize the spectrum, one can use adaptive filter

\[ H(z) = 1 - \frac{r(1)}{r(0)}z^{-1}, \]

where \( r(k) \) is the autocorrelation of the signal in the current frame. (This is the optimal linear predictor of first degree.)

Preprocessing thus produces the edited speech signal \( s(n) \), which will be processed in the later parts of the codec. The original speech will not be used anymore.
2.5 LPC-analysis and quantization

The speech $s(n)$ produced by the preprocessing is analysed on frame basis to get LPC-polynomial $A(z)$ which will be used in synthesis and weighting filters. Reconstructed speech is obtained by filtering the excitation signal (described later) by the synthesis filter $\frac{1}{A(z)}$. Ideally the excitation should be

$$S(z)A(z),$$

producing the original signal after the synthesis filter. We should actually take into accordance the filter memory in the synthesis but it will be handled later. LPC-analysis is implemented on frame by frame basis as well as all other processing, but the LPC-analysis does not need to be in exact correspondence with synthesis windows in which the quantization of excitation is done. In other words: the goal is to code certain frame of speech $s(n)$, which is non-windowed speech signal. Therefore we need LP-coefficients but it is not good idea to analyze these from a rectangular window because the discontinuity in the borders would cause error in the coefficients. Also longer analysis window makes LP-parameters smoother and easier to quantize.

Thus a soft window is used in LPC-analysis. Yet another reason to is that LP-coefficients need not to updated as often as other parameters of speech since they change relatively slowly. This leads
to usage of subframes: LP-coefficients are calculated e.g. with window length 20 ms and the other parameters in shorter windows e.g. 5 ms.

In each subframe LPCs are used to give the synthesis filter. The speech quality will improve, if the LP-parameters are interpolated from the surrounding frames (if they are not sent every 5 ms or so) in the center subframe.

To minimize the delay caused by LPC-analysis, a non-symmetric window, having the weight on the last subframe, can be used.

In figure 4 is an example where 20 ms frame is divided in four 5 ms subframes and the window is non-symmetric in figure 5.

Figure 3: *Suitable first degree highpass filters for preprocessing.*

Figure 4: *LPC-analysis and synthesis-windows.*
LPC-analysis can be made directly from the analysis window, but subjective speech quality has been noticed to improve by the following method. Especially the problem is that the amplitude response of the LP-filter can be too steep in the formant locations meaning that the poles in \( \frac{1}{A(z)} \) are too close to the unit circle causing metallic sound in the coded speech. It is possible to get rid off this by:

- The windowing the coefficients of \( A(z) \) by exponential window as we showed earlier.
- The windowing of the autocorrelation \( r(k) \) before LPC-analysis with window

\[
w(k) = \exp\left(-\frac{1}{2}\left(\frac{2\pi f_0 k}{F_s}\right)^2\right), \quad k = 0, 1, \ldots, p,
\]

where \( f_0 \) is the desired increment to bandwidth.

Example is in the picture 6.

Since the LPC-filter is needed also in the decoder, the coefficients must be sent to the decoder and quantized before that. To get the coder work synchronized with decoder, the quantized coefficients \( \tilde{A}(z) \) must be used also in the coder. Next we describe transformation of LP-coefficients into line spectral frequencies (LSF) and vector quantization of them yielding much better results comparing to scalar quantization.

### 2.5.1 LSF-Quantization

The problem is to quantize coeffiicients of \( A(z) \). As mentioned the scalar quantization of the coefficients \( a_1, a_2, \ldots, a_p \) is bad idea among other things because small error in these parameters can make the
filter unstable. Another feature to take into account is that the subjective speech quality should be as good as possible after quantization. Roughly this means that the amplitude response of the quantized filter should be close to the original especially formant frequencies and the bandwidths. The small change in LP-coefficients may cause large change in the amplitude response.

Vector quantization of the coefficients would be a better idea because the stability could be guaranteed by putting only stable filters in the codebook. However this is not directly implementable because the codebook must have some \(67108864\) vectors to provide good speech quality (26 bits). It is not possible and sensible to store such a large codebook in the mobile phone memory.

Because of this there have been developed several vector quantization schemes making various compromises between quantization precision, memory requirements and search complexity. Some examples of these methods are:

- gain-shape VQ: Codevectors are obtained by multiplying a scalar quantizer with a vector quantizer. This method is used typically in the fixed codebook in residual quantization and in the adaptive codebook.

- split VQ (split VQ): the vector to be coded is split in smaller parts which will be vector quantized with smaller codobooks. The codevectors are cartesian product of the small quantizers. This method is utilized e.g. in quantization of LSF-parameters corresponding to the LPC-polynomial.

But what is this LSF-transformation?

LSF-transformation (Line Spectral Frequencies) converts LPC-polynomial into a form which is easy to quantize, interpolate and check if the filter is stable.
For simplicity assume that the LPC-polynomial is of even degree

\[ A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_p z^{-p} \]

i.e. \( p \) is even (for odd values the analysis goes similarly). Let’s define new polynomials

\[ P_1(z) = A(z) + z^{-p-1} A(z^{-1}) , \]

\[ Q_1(z) = A(z) - z^{-p-1} A(z^{-1}) , \]

obtained from \( A(z) \) which correspond to adding to the reflection coefficients 1 (\( P_1(z) \)) and \(-1\) (\( Q_1(z) \)). Because

\[ z^{-p} A(z^{-1}) = z^{-p} (1 + a_1 (z^{-1})^{-1} + a_2 (z^{-1})^{-2} + \ldots + a_p (z^{-1})^{-p}) \]

\[ = z^{-p} (1 + a_1 z^1 + a_2 z^2 + \ldots + a_p z^p) \]

\[ = a_p + a_{p-1} z^1 + \ldots + a_1 z^{p-1} + z^{-p} , \]

so the coefficients of \( P_1(z) \) are obtained by adding to the coefficients the coefficients delayd with one sample and flipped upside down. By subtracting the flipped and delayd vector yields \( Q_1(z) \). If the zeros of \( A(z) \) are inside the unit circle, the zeros of \( P_1(z) \) and \( Q_1(z) \) are on the unit circle and moreover they appear on turns (see figure 7). Moreover \( P_1(z) \) has zero always in \( z = -1 \) and \( Q_1(z) \) has zero in \( z = 1 \). Thus by dividing them out we obtain new polynomials

\[ P(z) = \frac{P_1(z)}{1 + z^{-1}} , \]

\[ Q(z) = \frac{Q_1(z)}{1 - z^{-1}} . \]

Polynomials \( P(z) \) and \( Q(z) \) are fully determinized by the angles of the roots on the circle and since all the roots appear as complex conjugate pairs, it is suffiecient to handle only the half of the zeros. Let us denote these zeros of \( P(z) \) by \( \omega_1, \omega_3, \omega_5, \ldots, \omega_{p-1} \) and for \( Q(z) \) by \( \omega_2, \omega_4, \omega_6, \ldots, \omega_p \). So it holds that

\[ \omega_1 < \omega_2 < \omega_3 < \ldots < \omega_p , \]

if the zeros of \( A(z) \) are inside the unit circle. The parameters \( \omega_1, \ldots, \omega_p \) are the LSF-coefficients. The name ’line spectral frequencies’ is derived from the property that the poles of \( \frac{1}{P(z)} \) and \( \frac{1}{Q(z)} \) are on the unit circle and the frequency response consists of discrete lines.

According to the parameters \( \omega_1, \ldots, \omega_p \), the \( A(z) \) is obtained by

\[ P(z) = \prod_{i=1,3,5,\ldots,p-1} (1 - e^{j\omega_i} z^{-1})(1 - e^{-j\omega_i} z^{-1}) , \]

\[ Q(z) = \prod_{i=2,4,6,\ldots,p} (1 - e^{j\omega_i} z^{-1})(1 - e^{-j\omega_i} z^{-1}) , \]

22
Figure 7: The zeros of polynomials \( P_1(z) \) and \( Q_1(z) \) which are on the unit circle appear in turns and \( P_1(-1) = 0, Q_1(1) = 0 \).

\[
P_1(z) = P(z)(1 + z^{-1}),
\]

\[
Q_1(z) = Q(z)(1 - z^{-1}),
\]

\[
A(z) = \frac{1}{2}(P_1(z) + Q_1(z)).
\]

The LSF vector \( \Omega = (\omega_1, \omega_2, \ldots, \omega_p) \) can now be split in 4 parts for instance and quantized with 3 different codebooks. Stability is easy to check: just see if the zeros are in increasing order. If they are not, they can be reordered before transformation back to \( A(z) \). Similar trick cannot be done in split VQ of LPC-parameters (why not?).

LSF-vectors are also handy in interpolation, like reflection coefficients. In certain subframes it is useful to interpolate the LSF-coefficients from the surrounding subframe. For instance if the quantized LSF-vector in the previous frame was \( \Omega_{n-1} \) and in the current frame it is \( \Omega_n \), then in the four subframes the LSF-vectors could be

\[
0.75\Omega_{n-1} + 0.25\Omega_n, 0.5\Omega_{n-1} + 0.5\Omega_n, 0.25\Omega_{n-1} + 0.75\Omega_n, \Omega_n.
\]

These are automatically stable if \( \Omega_{n-1} \) and \( \Omega_n \) are. (why?)
LSF-coefficients have also the feature that if two consequent values $\omega_i$ and $\omega_{i+1}$ are close to each other, the filter $\frac{1}{A(z)}$ has a resonance (corresponding to a formant) on this frequency and it should be quantized carefully. In fact in quantization, the choice of quantization vector is made by finding a vector which makes the smallest error on these critical frequencies.

### 2.6 Adaptive Codebook

Especially in voiced speech, good prediction of the current sample can be obtained by using a long-term predictor

$$P(z) = -\alpha z^{-T},$$

where $\alpha$ is the prediction coefficient and $T$ is delay, which is close to the pitch delay or its multiple. The long-term filter is understood as a adaptive codebook, which makes things easier since the search can be implemented in similar way with the fixed codebook.

Analysis-by-synthesis codecs use so that first the adaptive codebook is searched to get the parameters $T, \alpha$. Then calculate the residual error and quantize it by the fixed codebook. In principal to get optimal result, all the parameters $\omega, \alpha, T, \phi, c_f$ ($\phi$ gain and the fixed codebook vector $\nu$) should be tested at the same time. This would require considerable amount of computation because all the parameters should be synthesized to get the error measure. For this reason we must settle to a partial optimization, in which we first quantize LSF-parameters then the adaptive codebook and then the fixed codebook.

It has been observed that LT-prediction can be calculated more accurately on the excitation signal rather than on the synthesized speech signal.

The prediction of the LT-predictor to the current excitation is simply the excitation delay with $T$ and multiplied by $\alpha$. Thus with different values of $T$ we get different excitations starting from the index \textit{Current} $- T$. Considering vectors having the length of the subframe we can think different values of $T$ to correspond to codevectors in the adaptive codebook.

If we want to search the LT-predictor with delays 20..147 (corresponding to frequencies 54...400 Hz, when $F_s = 8$ kHz) and the index of the first sample of the current subframe is $n$, then the adaptive codevector corresponding to the delay $T = 20$ is

$$[e(n-20) \ e(n-20+1) \ e(n-20+2) \ \cdots \ e(n-20+M-1)]^T,$$

where $e(n)$ is the excitation signal produced in the earlier subframes and $M$ is the length of the subframe. The codevector corresponding to the delay 21 is

$$[e(n-21) \ e(n-21+1) \ e(n-21+2) \ \cdots \ e(n-21+M-1)]^T,$$

and similarly up to the last vector

$$[e(n-147) \ e(n-147+1) \ e(n-147+2) \ \cdots \ e(n-147+M-1)]^T.$$

Notice that this codebook contains $128 = 2^7$ whose indices can be represented by 7 bits. The codebook consists of earlier reconstructed excitation signal because is must be in use also in the decoder. The ‘adaptive’ suits well because this codebook is chancing all the time.
If it happens that $M > 20$, then the first vectors in the codebook can be made periodic so that the first vector is

$$
\begin{bmatrix}
e([0 \mod 20] + n - 20) \\
e([1 \mod 20] + n - 20) \\
e([2 \mod 20] + n - 20) \\
\vdots \\
e([M - 1 \mod 20] + n - 20)
\end{bmatrix}
$$

where $k \mod 20$ is the remainder of division of $\frac{k}{20}$. In the second vector we use 21 instead of 20 and so forth. The search of the codebook are similar with the fixed codebook and it will be explained later.

### 2.7 The Fixed Codebook

After the adaptive codebook the residual signal is quantized with the fixed codebook. The aim is to find that vector which after summing with the adaptive codebook vector will be synthesized as close as possible to the original speech (producing minimal weighted error).

There are several different codebook structures proposed:

- **multipulse excitation, MPE**, where the excitation is modeled by certain amount of pulses in different locations with different amplitudes.

- **regular pulse excitation, RPE**, where the pulses reside on regular intervals (every fourth index e.g.) but the amplitude vary. The excitation can be represented by the delay of the pulse train and the amplitudes. This was used in the original GSM RPE-LTP codec.

- **stochastic codebook**, which contains random vectors and the scaling coefficient. This was used in the original CELP-codec.

- **Simplified CELP**, for instance ACELP (algebraic CELP) and VSELP (vector sum-ELP), where the codevectors have been constrained to allow fast search. This is used in current GSM-standard GSM EFR (enhanced full rate)-codec.

The multipulse method is already well-known so let’s see other methods more carefully.

#### 2.7.1 RPE

The problem in the multipulse method is the large amount of bits required by the positions of the pulses, which is

$$\log_2 \left( \frac{M}{N} \right),$$

where $M$ is the length of the subframe and $N$ is the number of pulses. In the RPE-codebook the pulses are constrained on periodic indices, for instance if the $M = 40$ and we use every fourth index, we obtain the following index sets:

$$\mathcal{I}_0 = \{0, 4, 8, 12, \ldots, 32, 36\},$$
\[ I_1 = \{1, 5, 9, 13, \ldots, 33, 37\}, \]
\[ I_2 = \{2, 6, 10, 14, \ldots, 34, 38\}, \]
\[ I_3 = \{3, 7, 11, 15, \ldots, 35, 39\}. \]

In this case the index set can be represented with 2 bits. As we saw the amplitudes can be optimized by using standard linear algebra whereas for the pulse locations there is no any easy optimal method. This generalizes also to several pulses: optimal amplitudes can be calculated from matrix equation if the pulse locations are known. Because there are only 4 choices for locations, we can go through all of them and take the best one.

### 2.7.2 CELP

The speech quality of MPE and RPE-codebook decreases when the bitrate is below 9.6 kbps approximately. This is due to large portion of bits going to the amplitudes by which it is possible to generate a large number of vectors which do not even appear very likely in speech signals.

Interestingly enough, a good choice for codebook are normal distributed random vectors. It is so because the LPC-filtering and the adaptive codebook removes the largest part of the interdependencies between the samples yielding relatively white noise-like residual.

In CELP-method the codebook comprises of e.g. 1024 excitation vectors (10 bit codebook) for 40 sample subframe with 8 kHz sampling frequency. The search is carried out as before: Find the index and the gain which produces the best reconstruction in the current subframe.

### 2.7.3 CELP-variants

The problem in basic CELP is the large amount of memory and the computational complexity in the search. To avoid this problem, several codebook structures have been proposed. The common priniciple is to add lots of zeros in the vectors thus minimizing the amount of arithmetical operations in the search. Moreover, codebooks with many zeros can be more effectively stored. Another possibility is to construc the codebook so that evaluation of the next codevector can be easily carried out after the previous one.

The alternatieves are:

- Sparse codebook, large part of the samples are zero.
- Ternary codebooks, where pulses get only values $-1, 0, 1$.
- Overlapping codebook, where the codevector is obtained from the previous by shifting it and by adding a few new samples.
- Algebraic codebook, where excitation vectors comprise of values $-1, 0, 1$ in certain indices. The codebook can be algebraicly generated (no memory consumption at all) and the search is fast due to sparseness.
2.8 Weighting Filter

The purpose of the weighting filter is to transform the spectrum of the error so that masking effect of
the formants would make the error disappear to human ear.

This can be realized by weighting the error unequally on different frequencies based on the speech
spectrum. The weighting must be small at the formant frequencies and large elsewhere. The spectrum
of the synthesis filter \( \frac{1}{A(z)} \) is a good approximation for the spectrum of the subframe. Thus it is utilized.

Commonly used weighting filter is

\[
\frac{A(z/\gamma_1)}{A(z/\gamma_2)},
\]

where \( 0 < \gamma_2 \leq \gamma_1 \leq 1 \), for instance. \( \gamma_1 = 0.94, \gamma_2 = 0.6 \) is a good choice. The weightin is required
only in the coder, thus it is possible to use unquantized \( A(z) \)-polynomial.

2.9 The Search in Adaptive and Fixed Codebooks

The most challenging phase in the analysis-by-synthesis coding is the codebook search and the re-
sulting parameters comprise the largest part of the bitstream.

So let’s look how we can find the indices of the adaptive and the fixed codebooks and the scaling
factors. At this point LPC-analysis and quantization have been already carried out. In the figure 8 the
different signals have been marked appearing in different parts of the coder. In the figure \( \mathbf{H} \) is the
convolution matrix of the synthesis filter(quantized), \( \mathbf{W} \) the convolution matrix of the weighting filter
\( \mathbf{s} \) is the original speech frame, \( \mathbf{a} \) the adaptive codebook vector, \( \alpha \) its scaling, \( \mathbf{f} \) the fixed codebook
vector and \( \phi \) its scaling, \( \mathbf{e} \) is the excitation vector (the weighted sum \( \alpha \mathbf{a} + \phi \mathbf{f} \)), \( z_0 \) is the zero input
response of the (quantized) synthesis filter and \( z_1 \) the zero-input response of the weighting filter.

![Figure 8: Decoder and the weighting filter inside analysis-by-synthesis–codec.](image)

Let’s first clarify the notion of zero-input response. A linear filter has memory elements, in which
is stored previous input samples and output samples. When we ‘star’ the filter, the memories are
empty, which is they contain 0. In filtering of speech and in non-stationary signals, the coefficients
change from time to time. If the filter memories are cleared when the coefficients are changed, there
is a serious discontinuity in the output. For this reason it is better to simply retain the contents of the
memory elements and change only the coefficients.

Now the filtering is no longer time-invariant operation and it cannot be represented as a matrix
multiplication. However, due to linearity, filtering with memory update has the following linear prop-
erty: if the output vector of the filter with memory is \( \mathbf{v} \) when the input is \( 0 \)-vector (all the samples
zero), then the output of the input $x$ is $y + v$ where $y$ is the output of the filtering the signal $x$ without memory.

In other words the output of the filter (with non-zero memory) having the input $x$, is

$$Hx + v,$$

where $v$ is the zero-input response (logical name) and $H$ is the convolution matrix of the filter.

What comes to codebook search, the situation is that we would like to find vectors and scaling factors which after quantized synthesis filtering (with non-zero memory) would produce the smallest possible error in the decoder. In other words we try minimize term

$$\|W(s - H(\alpha a + \phi f) - z_0) + z_1\|^2,$$

which can be as well written as

$$\|W(H(\alpha a + \phi f) + z_0 - s) - z_1\|^2.$$

The search is carried out by finding the best possible values for $a$ and $\alpha$.

How?

Let’s see first seemingly simpler situation, where we want to find a vector $\nu$ and a scalar $\nu$ s.t. we minimize

$$\|\nu Sv - d\|^2,$$

where $S$ is some given matrix and $d$ some given vector. This minimization is easy to solve by using linear algebra:

$$\|\nu Sv - d\|^2 = (\nu Sv - d)^T(\nu Sv - d) = \nu^2 v^T S^T Sv - 2\nu d^T Sv + d^T d,$$

(1)

(2)

giving the (similarly as in the multipulse codec) optimal value of $\nu$

$$\nu_{\text{min}} = \frac{d^T Sv}{v^T S^T Sv}.$$ 

By substituting this back to the (2), we obtain

$$\left(\frac{d^T Sv}{v^T S^T Sv}\right)^2 v^T S^T Sv - 2\frac{d^T Sv}{v^T S^T Sv} d^T Sv + d^T d = -\left(\frac{(d^T Sv)^2}{v^T S^T Sv}\right) + d^T d.$$

To minimize this it is sufficient to maximize the following term

$$\frac{(d^T Sv)^2}{v^T S^T Sv}.$$ 

In principle we evaluate this for every codevector $v$ and take the one with the biggest value. Then we calculate $\nu_{\text{min}}$.

Now we can apply this to the original problem by noticing that when we take

$$S = WH, \quad \nu = \alpha, \quad v = a, \quad d = W(s - z_0) + z_1,$$
the parameters $a$ and $\alpha$ can be found precisely as we described above. At this point the fixed codevector is 0.

When the parameters $a$ and $\alpha$ have been found, we try to find the fixed codevector minimizing the remaining error. This is solved by the same method by choosing

$$S = WH, \ \nu = \phi, \ v = f, \ d = W (s - z_0 - H(\alpha a)) + z_1,$$

and be selecting the vector in the fixed codebook which maximizes the term (3).

What remains to explain is the update of the filter memories. The principle is that all the filters work continuously. In other words when the subframe changes, the coefficients change, the memory elements are copied from the end of the previous subframe. When the optimal excitation has been found to the current subframe, the excitation signal $e = \alpha a + \phi f$ is filtered through the synthesis filter $\frac{1}{A(z)}$ and the memory is saved. After these filter operations the zero-input responses for the next subframe can be calculated. After this the difference of the original speech and the synthesized speech is filtered through the weighting filter and the memory is saved for the future use (see the diagram).

### 2.10 Postprocessing

Postprocessing is carried out only in the decoder. The idea is to improve the speech quality by removing quantization noise. The key idea is the same as in the weighting filter: try to minimize quantization noise on the frequencies where the speech amplitude is low.

How then?

Postprocessing splits in two parts. First noise suppression between formants with the familiar weighting filter

$$\frac{A(z/\gamma_1)}{A(z/\gamma_2)},$$

where $0 < \gamma_2 \leq \gamma_1 \leq 1$, for instance $\gamma_1 = 0.55$, $\gamma_2 = 0.7$ are this time good choices. (and in use in AMR-codec standard with low bitrates.)

Noise suppression between with harmonic frequencies is carried out by using the long term predictor:

$$\frac{1 + \gamma_3 z^{-T}}{1 - \gamma_4 z^{-T}}.$$

The parameters $\gamma_3$ and $\gamma_4$ cannot be fixed since this filter enforces periodicity and thus voiceness. Voiced speech becomes brighter but for non-voiced speech appears voiced noise which is not good. To determine these parameters there are lot of methods. May be the simplest method is to look at first reflection coefficient. For voiced speech it is close to 1 and smaller in the non-voiced speech. (why?)

Enhancement of coded and decoded speech compared to speech enhancement of natural speech, is easy because the coded speech is synthesized speech. We know precisely its parameters. On the other hand it is challenging to remove background noise which has come from the coder and has gone through the codec optimized for speech signals. For this reason the background noise removal belongs to the preprocessing and the quantization noise suppression to the postprocessing.