CHAPTER 5. ADAPTIVE LINE-SPECTRAL FREQUENCIES

In this chapter we apply the CRLS-SA structure to adaptively estimate the line-spectral frequencies (LSF). There are two schemes for this purpose. The first is a cascade of second order linear predictors (LP) adapted using CRLS-SA, for which the LP coefficients are converted to the corresponding LSF using a look-up table. The second scheme modifies the CRLS-SA structure to directly adaptively estimate the corresponding LSF parameters. We also look at the vector quantization of the LSF.

5.1. Introduction

Using a cascade of second order adaptive filters to estimate the second order section LSF, denoted as LSF\(_2\), is straightforward because the LSF\(_2\) can be obtained directly from the coefficients of each section, with very little additional computation. Hence, the computation of the LSF\(_2\) using a cascade structure is less complex than in the case of the higher order direct form. Furthermore, from the cascade structure we can get true information about the formant frequencies and their bandwidths because there is no influence from other roots [21]. There have been some attempts to compute the line-spectral pair frequencies (LSP), another form of the LSF, using a cascade adaptive filter. Some use the LMS adaptive filter [13], or a recursive least squares adaptive filter [69]. However, LMS is very slow to adapt, especially for speech signals, and recursive least squares (RLS) is computationally very expensive. When the gradient of the \(k\)-th section is computed by passing the input signal to all sections but the \(k\)-th section [69], this increases the computational load significantly.
We propose the computation of the LSF\(_2\) using the cascade recursive-least squares with subsection adaptation (CRLS-SA) adaptive filter. CRLS-SA is a least-squares based cascade adaptive filter, where each section is adapted independently [85] with the aim of reaching a global minimum, not a local minimum [65]. Since each section is a second order filter, the computational effort required by CRLS-SA is approximately \(20^*L^*N/2\), where \(N\) is the order of the filter, and \(L\) is the number of data. Also, for short data records, CRLS-SA gives a result closer to the known signal when compared to the widely used auto-correlation method [85].

There are two methods for using CRLS-SA to estimate LSF. First, we can obtain LSF\(_2\) by directly converting the LP coefficients for each section of the cascade. Second, when each section of the cascade structure uses the symmetric and anti-symmetric polynomials, \(P(z)\) and \(Q(z)\) respectively, the LSF\(_2\) coefficients of the section can be adapted directly. We reiterate the advantages of LSF computation using CRLS-SA, which are that its required computational effort is less or comparable to that of the auto-correlation method with LSF searching and approximation, that we can get the true formant frequencies and their bandwidths; and that for short data (around 20-30 ms speech segments) it gives an estimate closer to the true values than the estimate from the auto-correlation method. Furthermore we can also use the cascade LSF\(_2\) to directly synthesize the AR process signal.

5.2. Line Spectral Frequencies for a Cascade of Second Order Filters

A linear prediction (LP) error filter \(\tilde{A}(z)\) of order \(N\) can be denoted as:

\[
\tilde{A}(z) = 1 - \hat{A}(z) = 1 - \sum_{k=1}^{N} \hat{a}_k z^{-k}
\]  
(5.2.1a)
\[
\hat{A}(z) = \sum_{k=0}^{N} \hat{a}_k z^{-k}
\]  
(5.2.1b)

where the \( \hat{a}_k \)'s are the LP coefficients. \( \hat{A}(z) \) can also be written as a cascade of several second order LP filters:

\[
\hat{A}(z) = \prod_{k=1}^{M} 1 - \hat{A}_k(z)
\]  
(5.2.2)

where

\[
\hat{A}_k(z) = \sum_{i=1}^{N/2} \hat{a}_{k,i} z^{-i}
\]  
(5.2.3)

and \( M = N/2 \), with \( N \) being the order of the filter.

From \( 1 - \hat{A}(z) \), we can form the associated symmetric and anti-symmetric second order filters, \( P_k(z) \) and \( Q_k(z) \), defined as follows:

\[
P_k(z) = 1 - \hat{A}_k(z) + z^{-3} \left( 1 - \hat{A}_k(z^{-1}) \right) \\
= 1 - \left( \hat{a}_{k,1} + \hat{a}_{k,2} \right) z^{-1} - \left( \hat{a}_{k,1} + \hat{a}_{k,2} \right) z^{-2} + z^{-3}
\]  
(5.2.4)

\[
Q_k(z) = 1 - \hat{A}_k(z) - z^{-3} \left( 1 - \hat{A}_k(z^{-1}) \right) \\
= 1 - \left( \hat{a}_{k,1} - \hat{a}_{k,2} \right) z^{-1} - \left( \hat{a}_{k,2} - \hat{a}_{k,1} \right) z^{-2} - z^{-3}
\]  
(5.2.5)

\( P_k(z) \) and \( Q_k(z) \) have one root at \( z = -1 \) and \( z = 1 \) respectively, so that \( P_k(z) \) and \( Q_k(z) \) can be written as:

\[
P_k(z) = (1 + z^{-1}) \tilde{P}_k(z)
\]  
(5.2.6a)

\[
Q_k(z) = (1 - z^{-1}) \tilde{Q}_k(z)
\]  
(5.2.6b)

where
\[ \tilde{P}_k(z) = 1 + p_k z^{-1} + z^{-2} \]  \hspace{1cm} (5.2.6c)

\[ \tilde{Q}_k(z) = 1 + q_k z^{-1} + z^{-2} \]  \hspace{1cm} (5.2.6d)

Factoring \((1 + z^{-1})\) out of \(P_k(z)\) and \((1 - z^{-1})\) out of \(Q_k(z)\), we can show that

\[ p_k = -(1 + \hat{a}_{k,1} + \hat{a}_{k,2}) \]  \hspace{1cm} (5.2.7a)

\[ q_k = (1 - \hat{a}_{k,1} + \hat{a}_{k,2}) \]  \hspace{1cm} (5.2.7b)

We know that a second order filter with complex conjugate roots can be represented by

\[ B(z) = 1 - 2r \cos \theta \ z^{-1} + r^2 z^{-2} \]  \hspace{1cm} (5.2.8)

where \(\theta\) relates to the frequency and \(r\) to the radius of the root pair. If \(r = 1\), \(B(z)\) has its roots on the unit circle. So, from \(p_k\) and \(q_k\) we can get the frequencies of \(\tilde{P}_k(z)\) and \(\tilde{Q}_k(z)\) as

\[ \theta_{p,k} = \cos^{-1} \left( -\frac{p_k}{2} \right) \]  \hspace{1cm} (5.2.9a)

\[ \theta_{q,k} = \cos^{-1} \left( -\frac{q_k}{2} \right) \]  \hspace{1cm} (5.2.9b)

By using table look-up, the LSF \(\theta_{p,k}\) and \(\theta_{q,k}\) can be obtained directly from \(p_k\) and \(q_k\) respectively without additional computational load. No approximation to find the roots of the polynomial is needed due to the second order section approach. For \(A_k(z)\) to be minimum phase, the LSF must alternate on the unit circle, which for the second order sections implies

\[ 0 < \theta_{p,k} < \theta_{q,k} < \pi \ ; \ k = 1, \ldots, M \]  \hspace{1cm} (5.2.9c)

where \(M\) equals \(N/2\) [13].

The frequencies of \(P_k(z)\) and \(Q_k(z)\) implicitly reveals the formant bandwidth of the \(k\)-th section. The proof is as follows. From summing and subtracting (5.2.7a) and (5.2.7b) we get the following equalities:
\[
\hat{a}_{k,1} = -\frac{(q_k + p_k)}{2}
\]
\[
\hat{a}_{k,2} = 1 - \frac{q_k - p_k}{2}
\]

(5.2.10)

We know that, for a second order polynomial \(1 - \hat{A}(z)\) with \(\hat{A}(z)\) in (5.2.3), the frequency of the pole is \(\omega_k = \cos^{-1}\left(\frac{a_{k,1}}{2\sqrt{1 - a_{k,2}}}\right)\) and \(r^2 = -a_{k,2}\). Assuming \(z_k = re^{j\omega_k}\) is the pole of the \(k\)-th section, then from \(p_k\) and \(q_k\) we can compute the formant frequency \(f_k\) and its 3 dB bandwidth \(BW_3\) as [38]:

\[
f_k = \frac{\omega_k}{2\pi}
\]
\[
BW = \log_e\left(\frac{r}{\pi}\right)
\]
\[
BW_3 = \frac{-BW}{2\pi}
\]

(5.2.11)

Since \(r\) is less than 1, the bigger \(r\) the narrower the 3 dB bandwidth.

### 5.3. Direct LSF Adaptation

Note from (5.2.4) and (5.2.5) that the 2\(^{nd}\) order predictor polynomial \(\hat{A}_k(z)\) can be used to write the 2\(^{nd}\) order prediction error filter \(1 - \hat{A}_k(z)\) as

\[
1 - \hat{A}_k(z) = \frac{1}{2}\left[(1 + z^{-1})\tilde{P}_k(z) + (1 - z^{-1})\tilde{Q}_k(z)\right]
\]

(5.3.1a)

Consequently, each second order section in the cascade can be expressed in terms of the symmetric and anti-symmetric filter polynomials \(P_k(z)\) and \(Q_k(z)\), as shown in Figure 5.3.1. The structure in Figure 5.3.1 implements (5.3.1a) using (5.2.6a-5.2.6d).
Figure 5.3.1 Direct LSF Adaptation Section $1 - \hat{A}_k(z)$. 

We can thus represent the overall $1 - \hat{A}_k(z)$ as follows:

$$1 - \hat{A}(z) = \prod_{k=1}^{M} \left\{ \frac{1}{2} \left[ (1 + z^{-1}) \tilde{P}_k(z) + (1 - z^{-1}) \tilde{Q}_k(z) \right] \right\}$$

(5.3.1b)

The gradients $\varphi_{n,k,p}$ and $\varphi_{n,k,q}$ for adapting $\tilde{P}_k(z)$ and $\tilde{Q}_k(z)$ respectively are

$$\varphi_{n,k,p} = -\frac{\partial e_{n,M}}{\partial p_k} = -\frac{1}{2} (1 + z^{-1}) z^{-1} \prod_{i=1}^{M} \left\{ \frac{1}{2} \left[ (1 + z^{-1}) \tilde{P}_i(z) + (1 - z^{-1}) \tilde{Q}_i(z) \right] \right\} y_n$$

(5.3.2a)

and analogously

$$\varphi_{n,k,q} = -\frac{e_{n,M}}{2(1 - \hat{A}_k(z))}$$
\[
\varphi_{n,k,q} = -(1 - z^{-1}) z^{-1} \frac{e_{n,M}}{2 \left(1 - A_k(z)\right)} \quad (5.3.2b)
\]

We can use the CRLS-SA algorithm in the Chapter 2 to adaptively compute the LSF with the following changes:

\[
\varphi_{n,k} = [\varphi_{n,k,p} \quad \varphi_{n,k,q}]^T
\quad (5.3.3a)
\]
and using

\[
\hat{a}_{n,k} = [p_{n,k} \quad q_{n,k}]^T
\quad (5.3.3b)
\]

The complete Direct LSF adaptation algorithm is shown in Table 5.3.1.

**Table 5.3.1 Direct LSF Adaptation Using CRLS-SA.**

For \( k = 1, 2, \ldots, M \), compute the following

\[
d_{n,k} = y_{n,k}
\]
\[
y_{n-1,k} = [y_{n-1,k} \quad y_{n-2,k}]^T
\]
\[
\varphi_{n,k,p} = -(1 + z^{-1}) z^{-1} \frac{e_{n,M}}{2 \left(1 - \hat{A}_k(z)\right)}
\]
\[
\varphi_{n,k,q} = -(1 - z^{-1}) z^{-1} \frac{e_{n,M}}{2 \left(1 - \hat{A}_k(z)\right)}
\]
\[
\varphi_{n,k} = [\varphi_{n,k,p} \quad \varphi_{n,k,q}]^T
\]
\[
\kappa_{n,k} = \frac{P_{n-1,k} \varphi_{n,k}}{\lambda + \varphi_{n,k}^T P_{n-1,k} \varphi_{n,k}}
\]
\[
P_{n,k} = \lambda^{-1} P_{n-1,k} - \lambda^{-1} \kappa_{n,k} \varphi_{n,k}^T P_{n-1,k}
\]
\[
e_{n,k} = y_{n,k+1} = d_{n,k} - \hat{a}_{n-1,k}^T y_{n-1,k}
\]
\[
\hat{a}_{n,k} = \hat{a}_{n-1,k} + \kappa_{n,k} e_{n,M}
\]
5.4. Direct Synthesis

Furthermore, the second order section LSF parameters can be used directly for synthesis as follows.

\[
Y_k(z) = \frac{E_k(z)}{1 - \hat{A}_k(z)} = \frac{E_k(z)}{\frac{1}{2}\{P_k(z) + Q_k(z)\}} = \frac{2E_k(z)}{P_k(z) + Q_k(z)} = \frac{2E_k(z)}{(1 + z^{-1})\hat{P}_k(z) + (1 - z^{-1})\hat{Q}_k(z)}
\] (5.4.1)

Multiplying out we get:

\[
2E_k(z) = Y_k(z)[(1 + z^{-1})\hat{P}_k(z) + (1 - z^{-1})\hat{Q}_k(z)] = Y_k(z)\{\hat{P}_k(z) + \hat{Q}_k(z)\} + z^{-1}Y_k(z)\{\hat{P}_k(z) - \hat{Q}_k(z)\}
\] (5.4.2)

Replacing \( \hat{P}_k(z) \) using (5.2.6c) and \( \hat{Q}_k(z) \) using (5.2.6d) yields

\[
Y_k(z)\{2 + (p_k + q_k)z^{-1} + 2z^{-2}\} = 2E_k(z) - z^{-2}Y_k(p_k - q_k)
\] (5.4.3)

An inverse \( z \)-transform then yields the desired relation:

\[
y_{n,k} = \frac{1}{2}\left\{2e_{n,k} - y_{n-1,k}(p_k + q_k) - y_{n-2,k}(2 + p_k - q_k)\right\} = e_{n,k} - \frac{1}{2}y_{n-1,k}(p_k + q_k) - \frac{1}{2}y_{n-2,k}(p_k - q_k) - y_{n-2,k}
\] (5.4.4)

which is reflected in the diagram of Figure 5.4.1.
5.5. Performance Results

To evaluate the performance of LSF \(_2\) estimation using the CRLS-SA structure, 100 speech-like signal realizations are generated. The spectrum of the underlying tenth order AR process is shown in Figure 5.5.1.

The performance of the LSF estimator is evaluated by measuring the Itakura distance [19] between the estimated AR (10) process, obtained by converting the LSF\(_2\) back to an AR (10) process, and the actual AR (10) process. The length of the data records is 240 samples. For comparison, we also compute the LSF\(_{10}\) using the auto-correlation method, where from the auto-correlation estimates the polynomials \(P(z)\) and \(Q(z)\) are formed. Next the roots of \(P(z)\) and \(Q(z)\) are computed.
Figure 5.5.1 LSF$_{10}$ Speech Spectrum, and Underlying AR Process.

Typical results for LSF$_{10}$ generated with the Direct LSF CRLS-SA (DLSF CRLS-SA) algorithm are shown in Table 5.5.1. Note that, to facilitate comparison, the LSF$_2$ for the Direct LSF CRLS-SA come from the following process: second order section LSF are converted to AR(2) polynomials, the AR(2) polynomials are combined into an AR(10) polynomial, and then the LSF$_{10}$ are computed.

From the results in Table 5.4.1, based on 100 realizations, we see that both methods produce consistent results in terms of mean and standard deviation. Note however that Direct LSF CRLS-SA generally (not always) yields less bias and less variance in terms of the corresponding LSF root estimates. Recall too that for the auto-correlation method, the roots of $P(z)$ and $Q(z)$ are obtained using Chebyshev approximation [39] [42]. By contrast, for Direct LSF CRLS-SA the second order section line spectral frequencies themselves have been adapted so that these can be transmitted without further computation.
Table 5.5.1 Statistical Performance Comparison between the Direct LSF and Auto-correlation Approaches.

<table>
<thead>
<tr>
<th>Method</th>
<th>Direct LSF</th>
<th>Auto-correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSF root</td>
<td>Actual Mean/Std.Dev.</td>
<td>Mean/ Std.Dev.</td>
</tr>
<tr>
<td>P1</td>
<td>0.0445/0.0067</td>
<td>0.0550/0.0110</td>
</tr>
<tr>
<td>Q1</td>
<td>0.0679/0.0089</td>
<td>0.0685/0.0127</td>
</tr>
<tr>
<td>P2</td>
<td>0.0991/0.0052</td>
<td>0.1050/0.0079</td>
</tr>
<tr>
<td>Q2</td>
<td>0.1296/0.0072</td>
<td>0.1253/0.0058</td>
</tr>
<tr>
<td>P3</td>
<td>0.2273/0.0058</td>
<td>0.2299/0.0069</td>
</tr>
<tr>
<td>Q3</td>
<td>0.2569/0.0061</td>
<td>0.2496/0.0099</td>
</tr>
<tr>
<td>P4</td>
<td>0.2737/0.0055</td>
<td>0.2856/0.0049</td>
</tr>
<tr>
<td>Q4</td>
<td>0.3038/0.0049</td>
<td>0.2860/0.0169</td>
</tr>
<tr>
<td>P5</td>
<td>0.3219/0.0080</td>
<td>0.3350/0.0217</td>
</tr>
<tr>
<td>Q5</td>
<td>0.3807/0.0065</td>
<td>0.3612/0.0288</td>
</tr>
</tbody>
</table>

To provide another measure of comparison, the histograms of the Itakura distance to the actual process, associated with the AR process estimated by CRLS-SA and with the auto-correlation method, are shown in Figures 5.5.2 and 5.5.3 respectively.

Figure 5.5.2 Itakura Distance Histogram Distance Produced by CRLS-SA.
We see in Figures 5.5.2 and 5.5.3 that both the mean and the standard deviation of the Itakura distance for the AR process obtained by CRLS-SA are smaller than those obtained from the auto-correlation method. Also, the histogram in Figure 5.5.2 shows negative and positive values indicating the possibility of unbiased estimation. In Figure 5.5.3 on the other hand, the Itakura distance is exclusively positive, indicative of the biased estimates produced by the auto-correlation method. A possible explanation for the better CRLS-SA estimator is that, for our relatively short data record, the auto-correlation method has not yet been able to estimate the underlying AR process as well as CRLS-SA [85].

In the next experiment, the DLSF CRLS-SA structure is used to compute the LSF of the speech signal. The segment of the speech signal itself is shown in Figure 5.5.4. The LSF2 from each section are shown in Figures 5.5.5-5.5.9, where the LSF of P(z) are solid and the LSF of Q(z) are dotted. The LSF2 are obtained at the end of each iteration, where each iteration uses a
segment of 240 speech samples or 30 ms at 8 kHz sampling frequency. Then each LSF is expanded 240 times, so that the plot can be aligned with the speech signal.

We see in Figures 5.5.5-5.5.9 that the changes in LSF are usually small, which shows the smoothness of the speech from one segment to the next. An abrupt change in the LSF usually happens at the onset or offset of a spurt of speech. This is a true condition because at the onset and offset of speech there is big change in the vocal tract, which implies a big change in the LSF.

![Figure 5.5.4 Segment of Speech Signal.](image1)

![Figure 5.5.5 P(z) and Q(z) LSF of the First Section.](image2)
Figure 5.5.6 $P(z)$ and $Q(z)$ LSF$_2$ of the Second Section.

Figure 5.5.7 $P(z)$ and $Q(z)$ LSF$_2$ of the Third Section.

Figure 5.5.8 $P(z)$ and $Q(z)$ LSF$_2$ of the Fourth Section.
Figure 5.5.9 P(z) and Q(z) LSF$_2$ of the Fifth Section.

Figure 5.5.10 shows the histograms of the estimated formant frequencies for each section. Here we see that the first section mostly represents the lowest formant and the next section picks up the next higher formant. This is mostly a result of the initialization where the initial poles are distributed evenly on the unit circle, with the first section being initialized at the lowest frequency pole and the fifth section at the highest frequency pole. We know this is because of the initialization, since even if we rank order the formants so that the smallest formant is forced to be in the first section and the largest in the fifth section, the histogram distribution (see Figure 5.5.11) does not change appreciably. Note that Figure 5.5.10 and 5.5.11 look very similar; they are actually different only in minor details.
Figure 5.5.10 Formant Frequency Distribution of Each Section Before Ordering.

Figure 5.5.11 Formant Frequency Distribution of Each Section After Ordering.
5.6. Vector Quantization of LSF

Quantization of LSF is needed to reduce the transmission rate of the LSF. Since vector quantization (VQ) can obtain a lower distortion compared to scalar quantization with the same number of bits, VQ is usually used to quantize LSF. Here we present the method of vector quantizing LSF. Based on the properties of LSF, we will choose split VQ. Objective and subjective evaluation shows that the quantized LSF meets the usual requirements for speech coding purposes. Here we design and evaluate the vector quantization of the LSF.

5.6.1. Fundamentals of Vector Quantization

In vector quantization (VQ), an input vector or a block of input data are jointly quantized, instead of individually quantized as in scalar quantization. As a result, to meet a distortion criterion, a VQ requires fewer bits compared to a scalar quantization [66]. During the quantization process, an input data vector of length \( N \) is compared to the content of the codebook, called code-vectors. A code-vector that is closest to the input vector according to a specified criterion is chosen as the quantized version of the input vector.

Let \( y_n \) be an input vector of size \( N \). The quantized version of the input vector, \( \hat{y}_n \), is then

\[
\hat{y}_n = Q(y_n) = c_i
\]  

(5.6.1)

where \( Q( ) \) denotes quantization, \( c_i \) is the code-vector in the \( L \times N \) codebook \( C \) at index \( i \), \( N \) is the length of each code-vector, \( L = 2^B \) is the number of code-vectors in \( C \), and \( B \) is the number of bits used to represent the indices of \( C \). The vectors \( y_n \) and \( c_i \) have the following relation:

\[
d(y_n, c_i) \leq d(y_n, c_k); \quad k = 0, 1, 2, \ldots, L - 1, k \neq i
\]  

(5.6.2)

where \( d(x, y) \) denotes a distance function.
In most applications of VQ, such as in VQ of LSF, the probability density of the input to be quantized is unknown. Hence, the codebook is usually derived from training with a large number of input vectors. The training data is partitioned into \( L \) regions in \( \mathbb{R}^N \), where the mean or centroid of each region forms the entries of the codebook. The region associated with a code-vector \( c_i \) is called the Voronoi region [33] [66], denoted as \( V_i \) and defined as:

\[
V_i = \{ y \in \mathbb{R}^N : \| y - c_i \| < \| y - c_k \| ; k = 0,1,2,\ldots,L-1; i \neq k \}
\]  

(5.6.3)

Any \( y_n \in V_i \) will be quantized to \( c_i \). An example of 2-dimensional 4-point VQ (\( N=2, L=4, \) and \( B=2 \)) is illustrated in Figure 5.6.1.

![Figure 5.6.1 Illustration of 2-dimensional, 4-point VQ.](image)

The process to find \( c_i \) is usually done iteratively using a “clustering” algorithm. A widely used clustering algorithm is the Linde-Buzo-Gray (LBG) [19] algorithm. The LBG algorithm can be described briefly as follows.

**Initialization:**

Choose \( L \) arbitrary code-vectors denoted \( c_k, k=0,\ldots,L-1 \), and assign to each a cell.
1. For each code-vector \( y_n \) in the training data set, assign cluster \( y_n \) to region \( k^* \), one of the \( L \) regions based on

\[
k^* = \arg \min_k d(y_n, c_k), \quad k = 0, 1, \ldots, L - 1
\]  

(5.6.4a)

2. Compute the total distortion

\[
D = \sum_{i=0}^{P-1} d(y_i, Q(y_i))
\]  

(5.6.4b)

where \( P \) is the number of training data vectors.

if \( D \leq \varepsilon \), then STOP

3. For each region \( k \), compute the new centroid. The new centroids are the new entries of the codebook. Return to Step 1.

A VQ can be decomposed into encoding and decoding processes. In encoding, a code-vector that best represents the input vector is chosen. The index of the chosen code-vector is transmitted. In the decoding process, the received index is used to get the code-vector representing the quantized input vector. In the encoding process, if the processing to obtain a code-vector is done by comparing the input vector with all the code-vectors in the codebook, this process is called a full-search. A full-search becomes very complex if \( L \), the codebook size, is very large.

5.6.2. Vector Quantization for the Cascade LSF

VQ of the LSF is considered to have achieved good quality (transparency) if it satisfies the following requirements [47] [66]:

1. The average spectral distortion (SD) is less than 1 dB.
2. SD outliers greater than 2 dB comprise fewer than 2 %.

3. There is no SD outlier greater than 4 dB.

A full-search process can achieve the above transparency requirements for a large codebook size \( L \). However, a large codebook makes the search process computationally very complex. By making use of the structure of the LSF, there is a sub-optimum search that can result in good performance without full-search [47] [62] [61] [66], namely split VQ and multi-stage VQ (MSVQ).

In MSVQ, the quantization is done sequentially, stage by stage. At the first stage, a code-vector \( c_{0,k_0} \) is chosen to represent the input vector \( y_n \) if it satisfies:

\[
d(y_n, c_{0,k_0}) \leq d(y_n, c_{0,k}), \quad k = 0,1,\ldots, L_0, \quad k \neq k_0
\]  \hspace{1cm} (5.6.1a)

At the next stage, a code-vector \( c_{1,k_1} \) is chosen if it satisfies:

\[
d(y_n - c_{0,k_0}, c_{1,k_1}) \leq d(y_n - c_{0,k_0}, c_{1,k}), \quad k = 0,1,\ldots, L_1, \quad k \neq k_1
\]  \hspace{1cm} (5.6.5b)

At the final stage \( M \), a code-vector \( c_{M,k_{M-1}} \) is chosen if it satisfies:

\[
d(y_n - c_{M-2,k_{M-2}}, c_{M-2,k_{M-1}}) \leq d(y_n - c_{M-2,k_{M-2}}, c_{M-1,k_{M-1}}) \quad k = 0,1,\ldots, L_{M-1}, \quad k \neq k_{M-1}
\]  \hspace{1cm} (5.6.5c)

where \( L_s \) is the size of the sub-codebook \( C_s \), \( S=0,1,M-1 \), \( M \) is the number of stages, \( k_s \) is the chosen code-vector of the sub-codebook \( C_s \). The quantized version of the input vector is the sum of all the code-vectors chosen at every stage:

\[
\hat{y}_n = c_{0,k_0} + c_{1,k_1} + \cdots + c_{M,k_{M-1}}
\]  \hspace{1cm} (5.6.2)

In split VQ, the input vector \( y_{n,0} = [y_{n,0} \ y_{n,1} \cdots y_{n,N-1}] \) is split into two or more sub-code vectors. Let’s say \( N=10 \), and the input vector is split into (4,4,2), then the sub-code vectors are:
The codebook design for the LSF\(_2\) using split VQ is described as follows. The LSF\(_2\) are ordered such that \(f_1 < f_2 < f_3 < f_4 < f_5\), where \(f_k\) is the formant frequency of the LSF\(_2\) associated with the \(P\)-polynomial and \(Q\)-polynomial frequencies of the \(k\)-th section. The lower frequencies, \(P_1\) and \(Q_1\) associated with \(f_1\), are assigned to the first section, and the highest frequencies, \(P_5\) and \(Q_5\) associated with \(f_5\), are assigned to the fifth section. A mean-squared error (MSE) is used as the criterion for the design of the codebook as well as for selecting the best code-vector. The distance or distortion function with MSE can be the Euclidean distance. The distortion criterion is defined as:

\[
d_k, \quad k = 0, \ldots, M - 1
\]
where \( M \) denotes the number of splits, \( d_k \) is the distortion of the \( k \)-th sub-codebook. The minimum distortion for each codebook is:

\[
\text{mind}_k
\]  

(5.6.8b)

because each sub-codebook is optimized independently. The distortion \( d_k \) is defined:

\[
d_k (y_{n,k}, \hat{y}_{n,k}) = (y_{n,k} - \hat{y}_{n,k})^T I_k (y_{n,k} - \hat{y}_{n,k})
\]

(5.6.8c)

where \( \hat{y}_{n,k} \) is the quantized version of \( y_{n,k} \), and \( I_k \) is a identity matrix.

The codebook is evaluated using two kinds of spectral distortion SD. \( SD_1 \) defined as [47]:

\[
SD_1 = \frac{1}{n_1 - n_0} \sum_{n=n_0}^{n_1} 10 \log_{10} \left( \frac{|A_q(e^{j2\pi n/N})|^2}{|A(e^{j2\pi n/N})|^2} \right)^2
\]

(5.6.5a)

where \( A(e^{j2\pi n/N}) \) and \( A_q(e^{j2\pi n/N}) \) are un-quantized and quantized LPC filter frequency responses respectively, and \( SD_2 \) evaluated on the Bark Scale [15]

\[
SD_2 = \sqrt{\frac{1}{W_o} \sum_{j=0}^{W_j=4000} W_B(f)^2} 10 \log_{10} \left( \frac{|A_q(e^{j2\pi f})|^2}{|A(e^{j2\pi f})|^2} \right)^2
\]

(5.6.9b)

where

\[
W_B(f) = \frac{1}{25 + 75 \left( 1 + 1.4 \left( \frac{f}{1000} \right)^2 \right)^{6.9}}
\]

(5.6.9c)

\[
W_o = \sum_{j=0}^{W_j=4000} W_B(f)
\]

(5.6.9d)

Practically, a 256-point FFT is used to compute the spectrum of \( A(z) \) and \( A_q(z) \).
5.6.3. Codebook Design

Based on the properties of LSF\(_2\) in Section 5.6.2, we see that a split VQ seems to be a natural choice to quantize LSF\(_2\). We split the LSF\(_2\) into three sub-codebooks, namely VQ1, VQ2, and VQ3. VQ1 is for LSF\(_2\) section 1, VQ 2 is for LSF\(_2\) sections 2 and 3, and VQ3 is for LSF\(_2\) sections 4 and 5. We denote this codebook as split VQ (2,4,4). For the current design, 9-bits are allocated for each sub-codebook, so that a total of 27 bits is used. The training algorithm is shown in Table 5.6.1.

Table 5.6.1. Split VQ (2,4,4) Training Algorithm

<table>
<thead>
<tr>
<th>Initialization</th>
<th>1. Set iteration number (j=1).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Set (d_{vQ1} = 0), (d_{vQ2} = 0), (d_{vQ3} = 0), (d_r = 0), where (d_{vQk}) is the weighted error for the (k)-th VQ, and (d_r) is the total weighted error.</td>
</tr>
<tr>
<td></td>
<td>3. Create random codebook for each sub-codebook, arrange the random values as the entries for each index of the code such that (f_1 &lt; f_2 &lt; \cdots &lt; f_{10}). Put (f_1) and (f_2) in VQ1, (f_3), (f_4), (f_5), (f_6) in VQ2, and (f_7), (f_8), (f_9), (f_{10}) in VQ3.</td>
</tr>
<tr>
<td>Training</td>
<td>4. Partition the training data such that for each set of LSF, the lowest LSF is assigned to section 1 and the highest LSF is assigned to section 5.</td>
</tr>
<tr>
<td></td>
<td>5. Cluster the training data that belong to section 1 into VQ1, belonging to sections 2 and 3 into VQ2, and belonging to sections 4 and 5 into VQ3.</td>
</tr>
<tr>
<td></td>
<td>6. Compute new sub-codebook VQ1, VQ2, and VQ3.</td>
</tr>
<tr>
<td></td>
<td>7. Compute (d_{vQ1}), (d_{vQ2}), (d_{vQ3}).</td>
</tr>
<tr>
<td></td>
<td>8. Perform convergence test. If (d_{j-1,vQ1} - d_{j,vQ1} &gt; \epsilon), (d_{j-1,vQ2} - d_{j,vQ2} &gt; \epsilon), (d_{j-1,vQ3} - d_{j,vQ3} &gt; \epsilon), (j=j+1), go to 4.</td>
</tr>
<tr>
<td></td>
<td>9. End</td>
</tr>
</tbody>
</table>

Recall that we have 10 LSF grouped into 5 pairs of \(P\) and \(Q\) frequencies. We then arrange such that the lowest \(P\) and \(Q\) pair are put in the first section, the next lowest in the second section, and so on until the highest \(P\) and \(Q\) pair are located in the fifth section. Since codebook...
VQ1 is intended to represent the first section, codebook VQ2 is intended to represent the 2nd and 3rd sections, and codebook VQ 5 is to represent the 4th and 5th sections, we initialize to do just that. VQ1 is initialized with the lowest random P and Q pair, VQ2 is initialized with the next 2nd and 3rd highest, and VQ 3 is initialized with the 4th and 5th highest LSF’s. This initialization turns out to yield much better results, comparatively, than randomly initializing VQ1, VQ2, and VQ3.

5.6.4. Performance Evaluation

The first evaluation is to verify the transparency of the VQ. A test data base, which was not used for training, is quantized using the obtained codebooks, VQ1, VQ2, and VQ3. Then the SD using SD1 in (5.6.9a) and SD2 in (5.6.9b) are computed. The results are shown in Table 5.6.2.

<table>
<thead>
<tr>
<th>Spectral Distortion</th>
<th>Mean dB</th>
<th>2 dB &lt;Outlier &lt; 4 dB %</th>
<th>Outlier &gt; 4 dB %</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD1</td>
<td>1.0</td>
<td>2.01</td>
<td>0</td>
</tr>
<tr>
<td>SD2</td>
<td>0.169</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

We see in Table 5.6.2 that split VQ(2,4,4) barely meets the transparency requirements using SD1. This is because for SD1 all frequency components in the PSD are treated equally during the distortion measurement, so that distortion anywhere in the measurement range will equally contribute to the total distortion. Using SD2, on the other hand, higher frequencies are given less emphasis during distortion measurement. It is assumed in SD2 that higher frequencies are not perceptually (as) important to the human ear.

The next evaluation is by using real speech. From a block of speech, called a frame, LSF2 are estimated. The speech, then, is inverse filtered using these LSF2 to yield the residual. Next, the LSF2 are quantized. The re-synthesized speech is obtained by filtering the residual with the
quantized LSF$_2$ using direct synthesis, as explained in Section 5.4. This evaluation process is shown in Figure 5.6.2.

Every 20 ms, a 160 sample speech record $s_{n,M}$ is obtained, where $n=1,…,160$, and $M$ represents the $M$-th speech record, for example $M=2$ means that a second 160 sample speech record is collected. Then the LSF$_2$ are estimated from the 240 sample record (the previous 160 sample speech record combined with the first half of the current 160 sample speech record). This is shown in Figure 5.6.3.

**Figure 5.6.2 Analysis and Synthesis of Speech Signal Using Cascade LSF$_2$.**

Using the obtained LSF$_2$ parameters, the residual speech signal $\tilde{s}_{n,M}$ is generated by inverse filtering of the speech signal with the now fixed cascade LSF. Then the processed speech signal $\hat{s}_{n,M}$ is generated by filtering the residual speech signal with the quantized LSF$_2$ synthesis filter. This process is shown in Figure 5.6.2 and Figure 5.6.3. Note that for generating the LSF we use 240 samples by combining the previous 160 samples and the current 80 samples, so that there is some correspondence between the current LSF and the next LSF.
The results are evaluated by listening and computing the segmental signal-to-noise ratio (SSNR). The SSNR is computed as follows. At any time $n$, for which $n$ is a multiple of $N$, with $N$ being the size of a frame of the speech signal, $N$ samples of the original speech $s_{n,M}$ and the processed speech $\hat{s}_{n,M}$ are collected. Then for each frame, i.e. each $n$, the SSNR$_n$ is:

$$SSNR_n = 10 \log_{10} \left( \frac{\sum_{i=0}^{N-1} s_{n,i}^2}{\sum_{i=0}^{N-1} (s_{n,i} - \hat{s}_{n,i})^2} \right) \quad (5.6.1a)$$
where \( s_{n,i} \) are the components of \( s_{n,M} \) defined as:

\[
s_{n,M} = \begin{bmatrix} s_{n,0} & s_{n,2} & \cdots & s_{n,N-1} \end{bmatrix}^T
\]  

(5.6.10b)

and \( \hat{s}_{n,i} \) are the components of \( \hat{s}_{n,M} \) defined as:

\[
\hat{s}_{n,M} = \begin{bmatrix} \hat{s}_{n,0} & \hat{s}_{n,2} & \cdots & \hat{s}_{n,N-1} \end{bmatrix}^T
\]  

(5.6.10c)

\( N=160 \). Then the SSNR is the average of the SSNR\(_n\). A high value of SSNR shows that the processed speech is almost the same as the original speech.

The obtained SSNR is 7 dB. The result shows that the obtained SSNR is not very high. It turned out that this result arised because the obtained code book is not anywhere near optimal yet. That is, the split VQ is designed such that VQ1 will be used to represent the lowest LSF\(_2\), where this lowest LSF\(_2\) is assigned to the 1\(^{st}\) section, and VQ3 is for the next to highest and highest LSF\(_2\) that will be assigned to the 4\(^{th}\) and 5\(^{th}\) sections respectively. However, during training, all VQ were initialized randomly. It seems that the purely random initialization makes it difficult for all VQ to exhibit the desired order. After this observation the training was modified.

We retrain the VQ where the VQ now are initialized using random values but with the random values arranged such that the smallest value is assigned to VQ1, and the highest value assigned to VQ5. It was mentioned already that VQ 1 represents the codebook for section 1, VQ 2 represents sections 2 and 3, and VQ 3 represents sections 4 and 5. Also, the LSF\(_2\) are ordered such that section 1 represents the lowest poles, and section 5 represents the highest poles. So during initialization, VQ1 is initialized with the lowest LSF\(_2\) representing the lowest poles, VQ 2 is initialized by the next two higher poles, and VQ 3 is initialized with the 4\(^{th}\) and 5\(^{th}\) highest poles. This turns out to produce better results than with the previous initialization, where all codebooks are initialized randomly, regardless of the ordering. The reason is that by ordering the
initial values, it is easier for the VQ1, VQ2, and VQ3 codebooks to develop appropriate clusters, as before clustering we order the training data. The results from modified training are shown in Table 5.6.3. The obtained SSNR is 21.5 dB. This result is much better than the previous result.

Table 5.6.3 Spectral Distortion from Ordered Random Initialization.

<table>
<thead>
<tr>
<th>Spectral Distortion</th>
<th>Mean dB</th>
<th>2 dB &lt;Outlier &lt; 4 dB</th>
<th>Outlier &gt; 4 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD₁</td>
<td>0.58</td>
<td>1.2</td>
<td>0</td>
</tr>
</tbody>
</table>

To see how good an SSNR of 21.5 dB is, we compare with the SSNR for the Modulated Noise Reference Unit (MNRU) ITU Recommendation P.81 [10]. Because a subjective test of a speech coder can be different from other tests, there is a reference system that can be used to relate one test to another test. The widely used reference for this purpose is Modulated Noise Reference Unit (MNRU), which is defined in ITU Recommendation P.81. The MNRU produces noise similar to the noise produced by a PCM system, in which the original signal is contaminated by noise of which the amplitude is modulated by the instantaneous amplitude of the source signal (multiplicative noise), as shown in Figure 5.6.4 [10]. The attenuation in dB is chosen to create the desired ratio of source and noise powers at the output of the band-pass filter. This ratio is usually called Q dB. For example, Q 20 means that the ratio of source and noise powers is 20 dB. The higher the Q value, the less noisy the output, meaning that the quality of the output is good. The SSNR of the speech database used in this experiment for different Q is shown in Table 5.6.4.
We see in this table that the SSNR becomes almost saturated at about Q 30 with an SSNR of about 20 dB. Consequently, we conclude that the SSNR of 21.5 dB that we achieved is relatively much better than Q 40. This also means that the processed speech using the split VQ(2,4,4) is almost transparent. For comparison, good quality ADPCM ITU G.721 is almost equivalent to Q 28 [12].
5.7. Conclusion

The CRLS-SA algorithm was proposed for directly finding the line spectral frequencies (LSF) for the cascaded second-order-sections corresponding to a higher order AR process. This turns out to be a good alternative to using the popular auto-correlation method, which entails additional computation for conversion to LSF. Two different approaches can be used. The first is to use CRLS-SA as a cascade of second order AR inverse filters and to obtain the LSF by simply converting the coefficients of each section using a look-up table. The second approach directly adapts the LSF coefficients by constructing a cascade of second order section structures based on the LSF polynomials $P(z)$ and $Q(z)$. Simulation with, spectrally, speech-like signals shows that our direct adaptation method produces results that are statistically competitive with those from the popular auto-correlation approach.