CHAPTER 2. APPROACH AND COMPUTATIONAL ASPECTS

2.1. Overview of the Recursive Least-Squares Algorithm

The least-squares method is based on the available data to obtain a solution to the linear filtering problem. It does not assume that any statistical characterization of the data is available. The Recursive Least Squares (RLS) algorithm carries out the least-squares approach by way of minimizing the weighted cost function, which is expressed as [32]:

$$F_n = \sum_{i=1}^{n} l^{n-i} \left| e_i \right|^2 ; \quad 0 << l \le 1 \tag{2.1.1}$$

where l is the weighting factor, also called the forgetting factor.

The optimum linear inverse filter uses the input signal y_n to find the optimum value of the tap-weight vector $\hat{\boldsymbol{a}}$ of size N +1 by solving the following normal equation:

$$\mathbf{R}\hat{\hat{\boldsymbol{a}}} = \boldsymbol{p} \tag{2.1.2a}$$

where $\hat{\hat{\boldsymbol{a}}}$ is the coefficient vector representing the inverse filter, expressed as:

$$\hat{\boldsymbol{a}} = \left[1 \quad -\hat{\hat{\boldsymbol{a}}}\right] = \left[1 \quad -\hat{a}_1 \quad -\hat{a}_2 \quad \cdots \quad -\hat{a}_N\right] \tag{2.1.2b}$$

$\mathbf{R}$ is the *NxN* input signal auto-correlation matrix defined as:

$$\mathbf{R} = \sum_{i=1}^{n} l^{n-i} \boldsymbol{y}_i \boldsymbol{y}_i^T \tag{2.1.2c}$$

$$\boldsymbol{y_n} = \left[y_{n-1} \; y_{n-2} \cdots y_{n-N}\right]^T \tag{2.1.2d}$$

$\boldsymbol{p}$ is the cross-correlation between the input signal and the desired signal d_n defined as:

$$\boldsymbol{p} = \sum_{i=1}^{n} l^{n-i} \boldsymbol{y}_i d_i^* \tag{2.1.2e}$$

Using the matrix inversion lemma, the inverse matrix computation to solve (2.1.2a) for $\hat{\hat{\boldsymbol{a}}}$ can be avoided [32], and the solution to (2.1.2a) can be computed recursively. The complete RLS algorithm can be found in the literature [32]. The computational complexity of the RLS algorithm is O(N^2), with N being the order of the filter.

2.2. Linear Prediction in Cascade Structure

Linear prediction (LP) is used to identify the unknown system by way of "inverting" the processing that happens in that unknown system. Suppose an unknown system with the system function $1/[1-A(z)]$ is excited with an input signal x_n and it then responds with the output y_n. The linear prediction process will find an estimate of *A(z)*, let's say $\hat{A}(z)$, by way of filtering y_n with the filter $[1-\hat{A}(z)]$ and minimizing the error according to some criterion. Linear prediction of an autoregressive (AR) process is widely used in low rate speech coding.

The linear prediction, here the forward linear prediction, of an AR process is shown in Figure 2.2.1. The estimate of the AR polynomial, $\hat{A}(z)$, is obtained by minimizing F_n, the weighted sum of mean-squared forward prediction errors e_n:

$$F_n = \sum_{i=1}^{n} l^{n-i} |e_i|^2 . \tag{2.2.1}$$

Minimizing F_n results in a Yule-Walker equation or normal equation [43]:

$$\mathbf{R_n}\, \hat{\boldsymbol{a}}_n = \boldsymbol{p}_n \tag{2.2.2}$$

where $\mathbf{R}_n$ is the *NxN* auto-correlation matrix of the input signal y_n, as defined in (2.1.2d), and expressed as

$$\mathbf{R_n} = \begin{bmatrix} r_0 & r_1 & & r_{N-1} \\ r_1 & r_0 & \cdots & r_{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{N-1} & r_{N-2} & \cdots & r_0 \end{bmatrix} \tag{2.2.3}$$

$$r_k = \sum_{i=0}^{n} y_i y_{i-k} \qquad k=0,\ldots, N-1 \tag{2.2.4}$$

The right hand side of (2.2.2), $\boldsymbol{p}_n$, is the cross-correlation between the input and the desired signal, y_n and

$$\begin{aligned} \hat{\boldsymbol{a}} &= \begin{bmatrix} 1 & -\hat{a}_1 & -\hat{a}_2 \cdots -\hat{a}_N \end{bmatrix}^T \\ &= \begin{bmatrix} 1 & -\hat{\boldsymbol{a}}_{\boldsymbol{DF}}^{\boldsymbol{T}} \end{bmatrix}^T \end{aligned} \tag{2.2.5a}$$

$$\hat{\boldsymbol{a}}_{\boldsymbol{DF}} = \begin{bmatrix} \hat{a}_1 & \hat{a}_2 \cdots \hat{a}_N \end{bmatrix}^T \tag{2.2.5b}$$

are the coefficients defining $\hat{A}(z)$:

$$\hat{A}(z) = \sum_{k=1}^{N} \hat{a}_k z^{-k} \tag{2.2.5c}$$

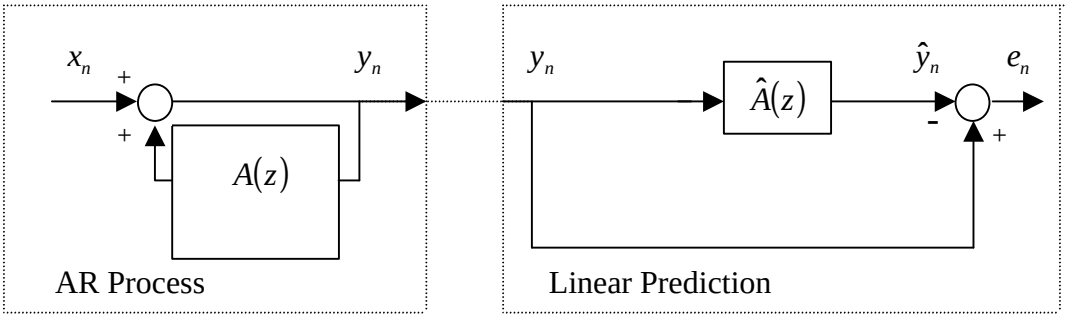

Figure 2.2.1 AR Process and Linear Prediction.

Equation (2.2.2) can be solved efficiently using the Levinson algorithm [43], or using adaptive filtering algorithms [32].

The forward linear prediction process in cascade structure is shown in Figure 2.2.2. The idea of using the cascade form in linear prediction was first proposed by Jackson and Wood [40]. By using a cascade structure, it is easier to find the root locations than when using the direct form. Also, using a cascade structure, it is easy to put constraints on the pole locations, for example to guarantee that a stable inverse exists. In the constrained case, the cascade structure provides easier and simpler ways to constrain the poles than for the direct form structure.

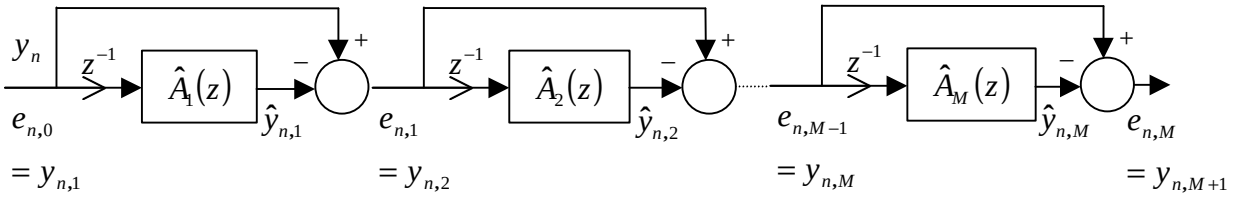

Figure 2.2.2 Linear Prediction in Cascade Structure.

Based on Figure 2.2.2, we can write the output from each section, $\hat{y}_{n,k}$, as

$$\begin{aligned}\hat{y}_{n,k} &= \hat{\boldsymbol{a}}_{\boldsymbol{k}}^{\boldsymbol{T}}\, \boldsymbol{y}_{\boldsymbol{n\text{-}1,k}} \\ &= \sum_{l=1}^{2} \hat{a}_{k,l}\, y_{n-l,k}\end{aligned} \tag{2.2.6a}$$

where $y_{n,k}$ is the input to the *k*-th section, and also the desired response for the *k*-th section, and

$$\hat{\boldsymbol{a}}_{\boldsymbol{k}} = \begin{bmatrix}\hat{a}_{k,1} & \hat{a}_{k,2}\end{bmatrix}^T \tag{2.2.6b}$$

are the coefficients defining $\hat{A}_k(z)$:

$$\hat{A}_k(z) = \sum_{l=1}^{2} \hat{a}_{k,l}\, z^{-l} \tag{2.2.6c}$$

The prediction error of the *k*-th section is

$$e_{n,k} = y_{n,k} - \hat{y}_{n,k} \tag{2.2.7}$$

The prediction error of the *k*-th section is the desired signal and input for the *k*+1-st section, etc.

The prediction error $e_{n,1}$ of the first section is:

$$\begin{aligned} e_{n,1} &= y_{n,1} - \hat{y}_{n,1} \\ &= y_n - \hat{y}_{n,1} \end{aligned} \tag{2.2.8a}$$

The prediction error $e_{n,2}$ of the second section is then

$$\begin{aligned} e_{n,2} &= y_{n,2} - \hat{y}_{n,2} \\ &= e_{n,1} - \hat{y}_{n,2} \\ &= y_n - \hat{y}_{n,1} - \hat{y}_{n,2} \end{aligned} \tag{2.2.8b}$$

Continuing this process we find that the prediction error $e_{n,M}$ of the final section, the *M*-th section, satisfies

$$e_{n,M} = y_n - \sum_{i=1}^{M} \hat{y}_{n,i} \tag{2.2.8c}$$

Using (2.2.6a), we can express the final prediction error as a function of the input signal and the coefficients of each section.

$$\begin{aligned} e_{n,M} &= y_n - \sum_{i=1}^{M} \hat{\boldsymbol{a}}_i^T \boldsymbol{y}_{\boldsymbol{n\text{-}1,i}} \\ &= y_n - \hat{\boldsymbol{a}}_{\boldsymbol{c}}^{\boldsymbol{T}} \hat{\boldsymbol{y}}_{\boldsymbol{n,c}} \end{aligned} \tag{2.2.8d}$$

where

$$\hat{\boldsymbol{a}}_{\boldsymbol{c}} = \left[\hat{\boldsymbol{a}}_{\boldsymbol{1}}^{\boldsymbol{T}} \ \hat{\boldsymbol{a}}_{\boldsymbol{2}}^{\boldsymbol{T}} \ \ldots \hat{\boldsymbol{a}}_{\boldsymbol{M}}^{\boldsymbol{T}}\right]^T \tag{2.2.8e}$$

with $\hat{\boldsymbol{a}}_k$ as given in (2.2.6b)

$$\hat{\boldsymbol{y}}_{n,c} = \left[\boldsymbol{y}^T_{n-1,1}\ \boldsymbol{y}^T_{n-1,2}\ \cdots \boldsymbol{y}^T_{n-1,M}\right]^T \tag{2.2.8f}$$

and

$$\boldsymbol{y}_{n-1,k} = \left[y_{n-1,k}\ y_{n-2,k}\right]^T \tag{2.2.8g}$$

The cost function to be minimized, rewritten from (2.2.1), is :

$$F_{n,M} = \sum_{l=1}^{n} l^{\,n-l}\left|e_{l,M}\right|^2 \tag{2.2.9}$$

Substituting (2.2.8d) into (2.2.9), we obtain

$$\begin{aligned} F_{n,M} &= \sum_{l=1}^{n} l^{\,n-l}\left|y_l - \hat{\boldsymbol{a}}_c^T \hat{\boldsymbol{y}}_{l,c}\right|^2 \\ &= \sum_{l=1}^{n} l^{\,n-l} y_l^2 - 2\sum_{l=1}^{n} l^{\,n-l} y_l\, \hat{\boldsymbol{y}}_{l,c}^T \hat{\boldsymbol{a}}_c + \sum_{l=1}^{n} l^{\,n-l} \hat{\boldsymbol{a}}_c^T \hat{\boldsymbol{y}}_{l,c} \hat{\boldsymbol{y}}_{l,c}^T \hat{\boldsymbol{a}}_c \end{aligned} \tag{2.2.10}$$

Minimizing (2.2.10) with respect to the coefficients of each section, we obtain:

$$\frac{\P F_{n,M}}{\P \hat{\boldsymbol{a}}_c} = \left[\frac{\P F_{n,M}}{\P \hat{a}_{1,1}}\ \frac{\P F_{n,M}}{\P \hat{a}_{1,2}}\ \frac{\P F_{n,M}}{\P \hat{a}_{2,1}}\ \frac{\P F_{n,M}}{\P \hat{a}_{2,2}} \cdots \frac{\P F_{n,M}}{\P \hat{a}_{M,1}}\ \frac{\P F_{n,M}}{\P \hat{a}_{M,2}}\right]^T \tag{2.2.11a}$$

From (2.2.10) and (2.2.6a)

$$\frac{\P F_{n,M}}{\P \hat{a}_{k,1}} = -2\sum_{l=1}^{n} l^{\,n-l} y_l\, y_{l-1,k} + 2\sum_{l=1}^{n} l^{\,n-l} y_{l-1,k} \hat{\boldsymbol{y}}_{l,c}^T \hat{\boldsymbol{a}}_c \tag{2.2.11b}$$

$$\frac{\P F_{n,M}}{\P \hat{a}_{k,2}} = -2\sum_{l=1}^{n} l^{\,n-l} y_l\, y_{l-2,k} + 2\sum_{l=1}^{n} l^{\,n-l} y_{l-2,k} \hat{\boldsymbol{y}}_{l,c}^T \hat{\boldsymbol{a}}_c \tag{2.2.11c}$$

and so on. Hence we get:

$$\frac{\P F_{n,M}}{\P \hat{\boldsymbol{a}}_{\boldsymbol{c}}} = \begin{bmatrix} -2\sum_{l=1}^{n} l^{\,n-l} y_l\, y_{\boldsymbol{l\text{-}1,1}} + 2\sum_{l=1}^{n} l^{\,n-l} y_{\boldsymbol{l\text{-}1,1}} \hat{\boldsymbol{y}}_{\boldsymbol{l,c}}^{\boldsymbol{T}} \hat{\boldsymbol{a}}_{\boldsymbol{c}} \\ -2\sum_{l=1}^{n} l^{\,n-l} y_l\, y_{\boldsymbol{l\text{-}2,1}} + 2\sum_{l=1}^{n} l^{\,n-l} y_{\boldsymbol{l\text{-}2,1}} \hat{\boldsymbol{y}}_{\boldsymbol{l,c}}^{\boldsymbol{T}} \hat{\boldsymbol{a}}_{\boldsymbol{c}} \\ \vdots \\ -2\sum_{l=1}^{n} l^{\,n-l} y_l\, y_{\boldsymbol{l\text{-}1,M}} + 2\sum_{l=1}^{n} l^{\,n-l} y_{\boldsymbol{l\text{-}1,M}} \hat{\boldsymbol{y}}_{\boldsymbol{l,c}}^{\boldsymbol{T}} \hat{\boldsymbol{a}}_{\boldsymbol{c}} \\ -2\sum_{l=1}^{n} l^{\,n-l} y_l\, y_{\boldsymbol{l\text{-}2,M}} + 2\sum_{l=1}^{n} l^{\,n-l} y_{\boldsymbol{l\text{-}2,M}} \hat{\boldsymbol{y}}_{\boldsymbol{l,c}}^{\boldsymbol{T}} \hat{\boldsymbol{a}}_{\boldsymbol{c}} \end{bmatrix} \tag{2.2.11d}$$

$$\begin{aligned} &= -2\sum_{l=1}^{n} l^{\,n-l} y_l \begin{bmatrix} y_{\boldsymbol{l\text{-}1,1}} \\ y_{\boldsymbol{l\text{-}2,1}} \\ \vdots \\ y_{\boldsymbol{l\text{-}1,M}} \\ y_{\boldsymbol{l\text{-}2,M}} \end{bmatrix} + 2\sum_{l=1}^{n} l^{\,n-l} \begin{bmatrix} y_{\boldsymbol{l\text{-}1,1}} \\ y_{\boldsymbol{l\text{-}2,1}} \\ \vdots \\ y_{\boldsymbol{l\text{-}1,M}} \\ y_{\boldsymbol{l\text{-}2,M}} \end{bmatrix} \hat{\boldsymbol{y}}_{\boldsymbol{l,c}}^{\boldsymbol{T}} \hat{\boldsymbol{a}}_{\boldsymbol{c}} \\ &= -2\sum_{l=1}^{n} l^{\,n-l} y_l\, \hat{\boldsymbol{y}}_{\boldsymbol{l,c}} + 2\sum_{l=1}^{n} l^{\,n-l} \hat{\boldsymbol{y}}_{\boldsymbol{l,c}} \hat{\boldsymbol{y}}_{\boldsymbol{l,c}}^{\boldsymbol{T}} \hat{\boldsymbol{a}}_{\boldsymbol{c}} \end{aligned} \tag{2.2.11e}$$

where $\hat{\boldsymbol{y}}_{l,c}$ is as defined in (2.2.8f).

Define the gradient of each section as the negative of the derivative of the final output error, as given in (2.2.8d), with respect to the coefficients for that section. The individual coefficient gradients for the *k*-th section, refer also to (2.2.6a), are:

$$\begin{aligned} y_{\,n,k,l} &= -\frac{\partial e_{n,M}}{\partial \hat{a}_{k,l}} \quad ; \quad l=1,2 \\ &= y_{n-l,k} \end{aligned} \tag{2.2.12a}$$

where $y_{\,n,k,l}$ constitute the gradient of the *k*-th section:

$$\begin{aligned} y_{\,n,k} &= \begin{bmatrix} y_{\,n,k,1} & y_{\,n,k,2} \end{bmatrix}^T \\ &= \begin{bmatrix} y_{n-1,k} & y_{n-2,k} \end{bmatrix}^T \end{aligned} \tag{2.2.12b}$$

The gradients of all sections form a column vector:

$$\begin{aligned} \boldsymbol{y}_n &= \begin{bmatrix} y_{n,1,1} & y_{n,1,2} & y_{n,2,1} & y_{n,2,2} \cdots y_{n,M,1} & y_{n,M,2} \end{bmatrix}^T \\ &= \begin{bmatrix} y_{n-1,1} & y_{n-2,1} & y_{n-1,2} & y_{n-2,2} \cdots y_{n-1,M} & y_{n-2,M} \end{bmatrix}^T \\ &= \hat{\boldsymbol{y}}_{\boldsymbol{n,c}} \end{aligned} \tag{2.2.12c}$$

where $\hat{\boldsymbol{y}}_{\boldsymbol{n,c}}$ is as defined in (2.2.8f).

Rewriting (2.2.11e) and replacing $\hat{\boldsymbol{y}}_{\boldsymbol{n,c}}$ with $\boldsymbol{y}_n$ we obtain:

$$\frac{\P F_{n,M}}{\P \hat{\boldsymbol{a}}_{\boldsymbol{c}}} = -2\sum_{l=1}^{n} l^{n-l} y_l \boldsymbol{y}_l + 2\sum_{l=1}^{n} l^{n-l} \boldsymbol{y}_l \boldsymbol{y}_l^{\boldsymbol{T}} \hat{\boldsymbol{a}}_{\boldsymbol{c}} \tag{2.2.13a}$$

Equating (2.2.13a) to zero and rearranging the expression, we get:

$$\sum_{l=1}^{n} l^{n-l} \boldsymbol{y}_l \boldsymbol{y}_l^{\boldsymbol{T}} \hat{\boldsymbol{a}}_{\boldsymbol{c}} = \sum_{l=1}^{n} l^{n-l} y_l \boldsymbol{y}_l \tag{2.2.13b}$$

The coefficients of each section of the cascade structure can be obtained from the normal equation (2.2.13b):

$$\hat{\boldsymbol{a}}_{\boldsymbol{c}} = \hat{\mathbf{R}}_y^{-1} \hat{\boldsymbol{p}}_y \tag{2.2.13c}$$

where

$$\hat{\mathbf{R}}_y = \sum_{l=1}^{n} l^{n-l} \boldsymbol{y}_l \boldsymbol{y}_l^{\boldsymbol{T}} \tag{2.2.13d}$$

and

$$\hat{\boldsymbol{p}}_y = \sum_{l=1}^{n} l^{n-l} y_l \boldsymbol{y}_l \tag{2.2.13e}$$

Now we take a look at an alternate expression for the final prediction error filter. The output error of each section, as shown in Figure 2.2.2, is:

$$e_{n,k} = \left[1 - \hat{A}_k(z)\right] \; e_{n,k-1} \tag{2.2.14a}$$

The output error of the final section can be written as:

$$\begin{aligned} e_{n,M} &= \left[\prod_{i=1}^{M}\left[1 - \hat{A}_i(z)\right]\right] e_{n,0} \\ &= \left[\prod_{i=1}^{M}\left[1 - \hat{A}_i(z)\right]\right] y_n \end{aligned} \tag{2.2.14b}$$

where $e_{n,0}$ is equal to y_n, the input to the cascade structure. Using (2.2.14b), the gradient of each section is then derived by taking the negative derivative of the output error of the final section, as stated in (2.2.12a), so that:

$$\begin{aligned} y_{n,k,l} &= -\frac{\partial e_{n,M}}{\partial \hat{a}_{k,l}} = z^{-l}\left[\prod_{\substack{i=1 \\ i\neq k}}^{M}\left[1 - \hat{A}_i(z)\right]\right] y_n \\ &= \frac{z^{-l}}{1 - \hat{A}_k(z)}\left[\prod_{i=1}^{M}\left[1 - \hat{A}_i(z)\right]\right] y_n \\ &= \frac{z^{-l}}{1 - \hat{A}_k(z)} e_{n,M} \end{aligned} \tag{2.2.15}$$

where l=1,2. From (2.2.15), we see that the gradient of the k-th section can be computed efficiently by passing the final output error $e_{n,M}$ to $\left[1 - \hat{A}_k(z)\right]^{-1}$ as shown in Figure 2.2.3. Hence the computation of the gradient does not increase the computational complexity of the cascade structure algorithm significantly relative to its total computational complexity.

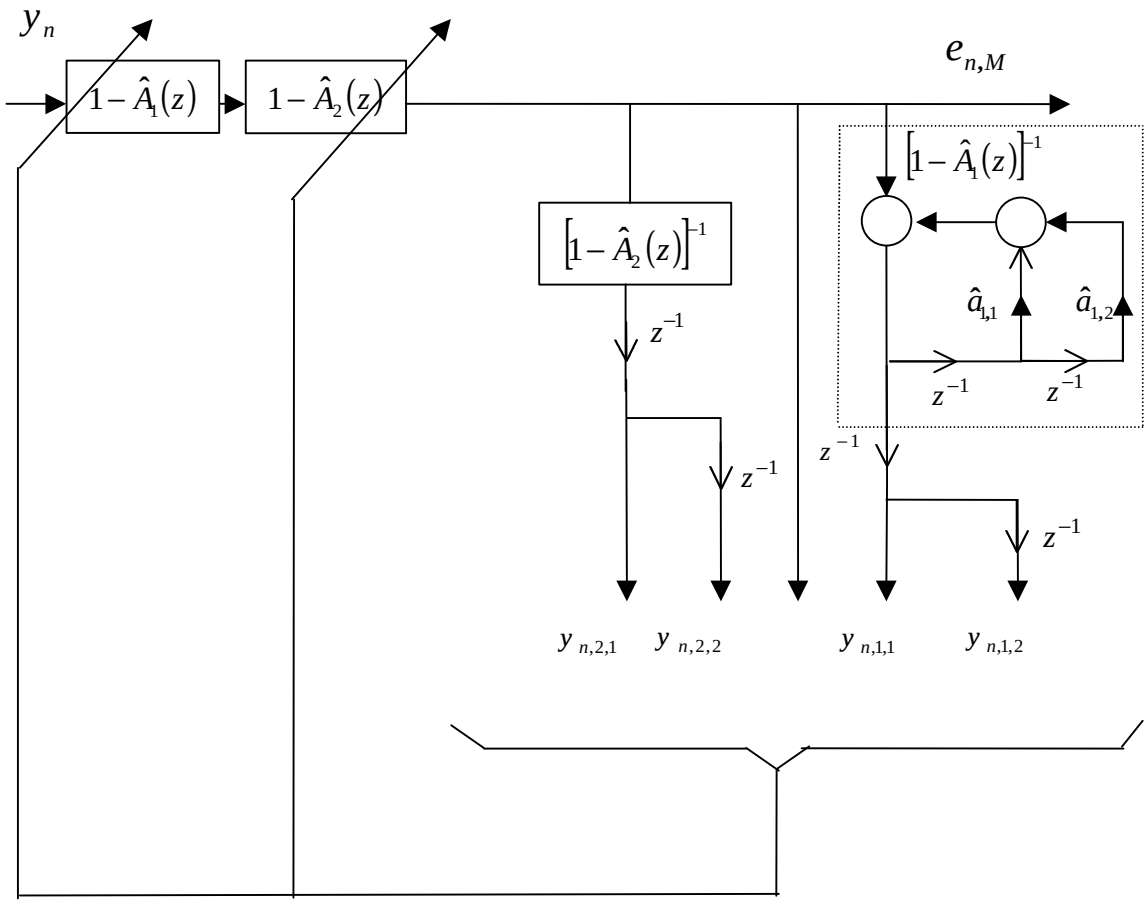

Figure 2.2.3 Cascade Linear Prediction and Gradient Computation.

2.3. CRLS-SA

From (2.2.15) we see that computing the gradient is the same as performing a whitening process except that the poles in the *k*-th section are not whitened, as only the poles of the *k*-th section are retained while the other poles have been removed. Note that this happens at or near convergence. In other words, the gradient of the *k*-th section depends mainly on the poles of the *k-th* section. As a result, the gradients for the different sections correspond to different poles, or, in the frequency domain, they (generally) occupy mostly different frequency bands. Because the poles dominate in different regions of the frequency domain at convergence, the correlation

between the gradients for different sections will be relatively small. Hence, $\hat{\mathbf{R}}_y$, the auto-correlation matrix of the gradient of all sections, as seen in (2.2.13d), tends to the form of a block-diagonal matrix. The latter is called the *direct sum matrix* [25], where each diagonal component is a *2x2* matrix that corresponds to the auto-correlation matrix of the gradient for one section of the cascade structure.

To verify the tendency towards a block-diagonal gradient auto-correlation matrix, a signal with poles at .99±0.1j and -0.891±0.09j, as shown in Figure 2.3.1, is generated. The spectra of the gradients of the first and second sections are shown in Figures 2.3.2 and 2.3.3, respectively. We see from Figures 2.3.2 and 2.3.3 that the gradients of the first and second sections occupy mostly different frequency bands.

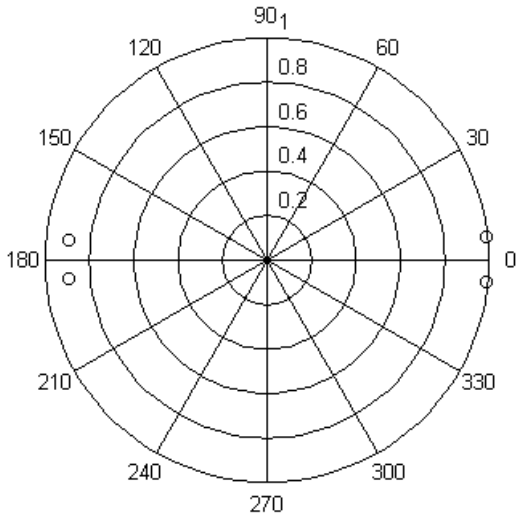

Figure 2.3.1 Example 1 Pole Locations.

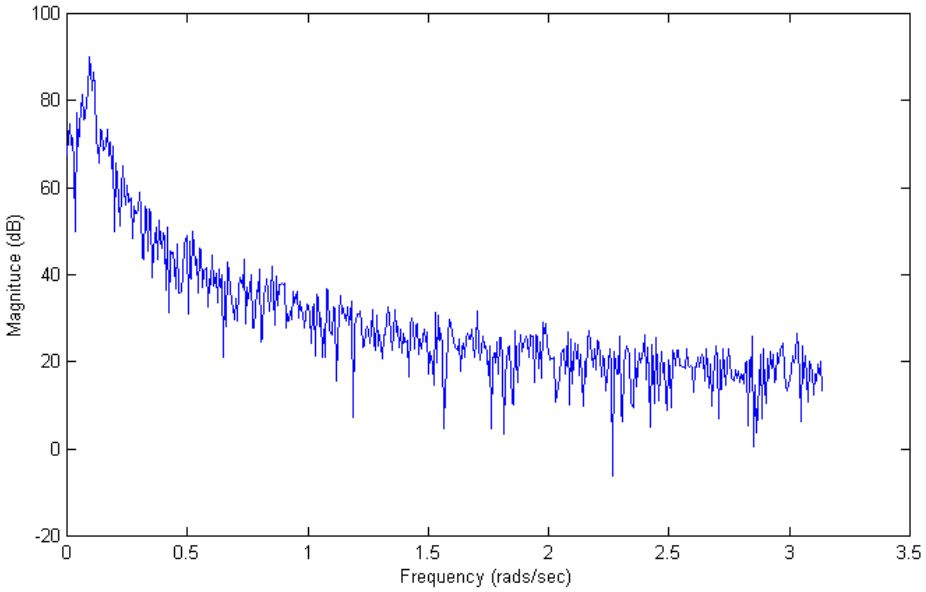

Figure 2.3.2 Spectrum of the 1st Section Gradient.

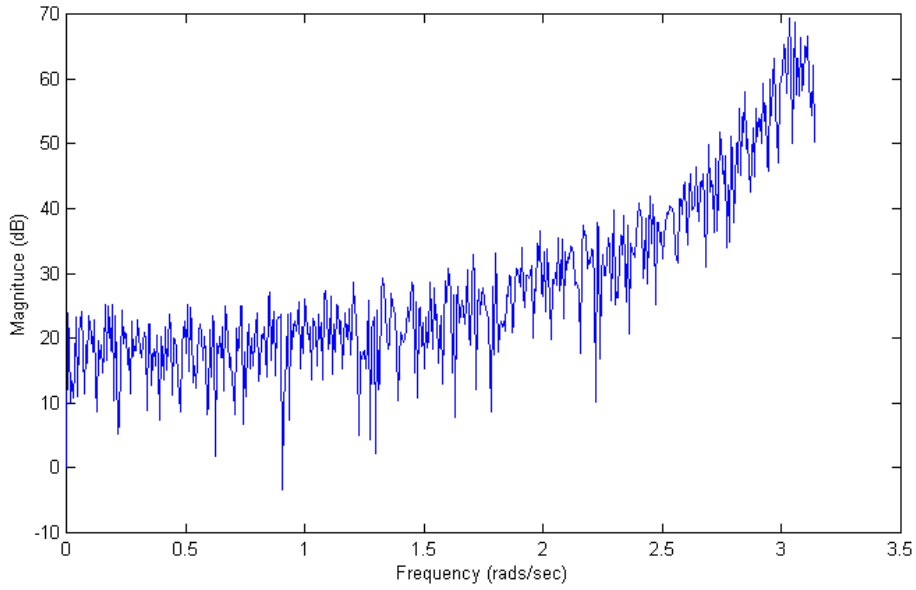

Figure 2.3.3 Spectrum of the 2nd Section Gradient.

The normalized cross-correlation between the first and second gradients is shown in Figure 2.3.4, where we see that they are not very correlated, since the value of this cross-correlation is much smaller than the auto-correlation shown in Figure 2.3.5.

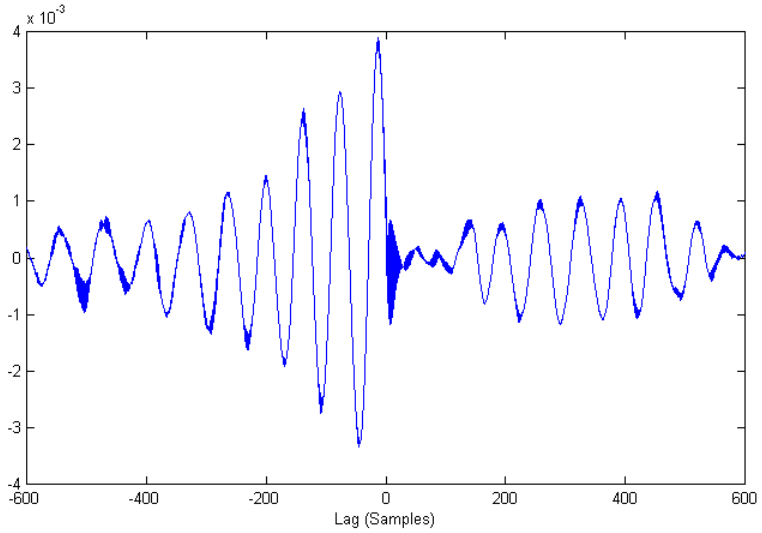

Figure 2.3.4 Normalized Cross-correlation of the First and Second Section Gradients.

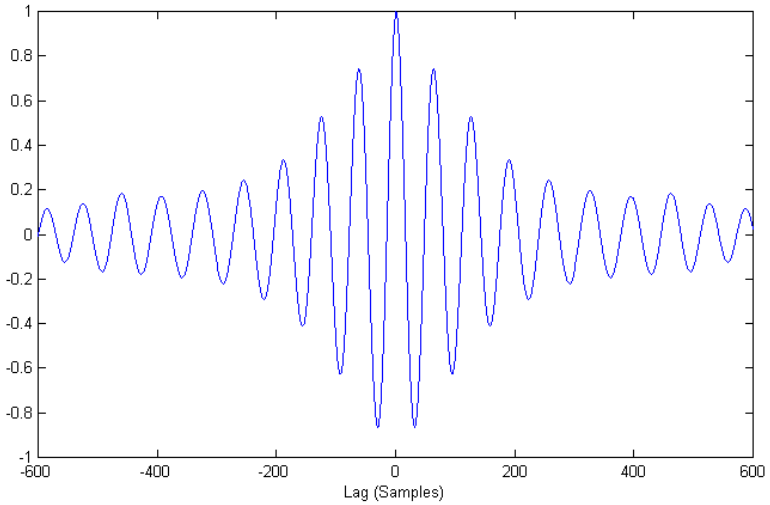

Figure 2.3.5 Normalized Auto-correlation of the First Section Gradient.

The cross-correlation of the *i*-th and *j*-th section gradients is computed as follows:

$$r_{ij,k} = \sum_{n=0}^{L-1} y_{\,n-k,i}\, y_{\,n,j} \tag{2.3.1}$$

where $y_{\,n,i}$ is the gradient of the *i*-th section, $y_{\,n,j}$ is the gradient of the *j*-th section, and *L* is the length of the data record used.

The estimated gradient auto-correlation matrix that is used in the RLS algorithm, for order 2, is:

$$\hat{\mathbf{R}}_y = 10^7 \begin{bmatrix} 1.5327 & 1.5249 & 0.0000 & 0.0001 \\ 1.5249 & 1.5327 & -0.0001 & 0.0000 \\ 0.0000 & -0.0001 & 0.0491 & -0.0486 \\ 0.0001 & 0.0000 & -0.0486 & 0.0491 \end{bmatrix}$$

where $\hat{r}_{1,1}$, $\hat{r}_{1,2}$, $\hat{r}_{2,1}$, and $\hat{r}_{2,2}$ comprise the 2*x*2 matrix which corresponds to the auto-correlation of the first gradient, while $\hat{r}_{3,3}$, $\hat{r}_{3,4}$, $\hat{r}_{4,3}$, and $\hat{r}_{4,4}$ comprise the 2*x*2 matrix which corresponds to the auto-correlation of the components of the second section gradient. The auto-correlation matrix $\hat{\mathbf{R}}_y$ is estimated from $\hat{\mathbf{R}}_y = \sum_{n=400}^{4000} \left[y^{\,T}_{\,n,1}\; y^{\,T}_{\,n,2} \right]^T \left[y^{\,T}_{\,n,1}\; y^{\,T}_{\,n,2} \right]$. Note that the gradient auto-correlation is estimated from iterations 400 through 4000, so that the transient, which persists for about 50 iterations, is not included. We see that the cross-correlation estimates are much smaller than the auto-correlation estimates, indicating that the gradients of the different sections are nearly uncorrelated. The top-left 2*x*2 matrix of $\hat{\mathbf{R}}_y$ represents the gradient associated with the first set of poles, which are at 0.99±0.1j.

The true auto-correlation for a second order filter can be computed easily from the filter coefficients as follows. Given a 2nd order AR process:

$$y_n = e_n - a_1 y_{n-1} - a_2 y_{n-1} \tag{2.3.2}$$

where e_n is a zero mean white Gaussian noise (WGN) with a variance of s_e^2. Then the auto-correlation of y_n can be derived as follows [32]:

$$s_y^2 = \left(\frac{1+a_2}{1-a_2}\right)\frac{s_e^2}{(1+a_2)^2 - a_1^2} \tag{2.3.3a}$$

$$r_0 = s_y^2 \tag{2.3.3b}$$

$$r_1 = \frac{a_1}{1+a_2} s_y^2 \tag{2.3.3c}$$

where s_y^2 is the variance of y_n, r_0 is the auto-correlation of y_n at lag 0, and r_1 is auto-correlation of y_n at lag 1. Using equations (2.3.3a), (2.3.3b), and (2.3.3c), the true auto-correlation, i.e. the analytical values, for the first set of poles located at .99±0.1j, at lags 0 and 1 are 5013 and 4987 respectively. The estimated auto-correlation at lag 0 is $1.5327 \times 10^7/3601=4255.3$ and at lag 1 it is $1.5249 \times 10^7/3601=4233$; these estimates are close to the true values. The bottom-right 2*x*2 matrix represents the gradient auto-correlation for the second set of poles, located at –0.891±0.09j. The true auto-correlations at lag 0 and 1 are 127.0751 and -125.6660 respectively. The estimated auto-correlations at lag 0 and 1 are 4908.9/3601=136.2838 and -4858.8/3601= -134.8918 respectively, which are also close to their true values. The auto-correlation components are also close to the true values, so that even when more data is used these estimates remain almost the same.

We also ran a simulation for an AR process of order 10 with the signal spectrum as shown in Figure 2.3.6. This spectrum resembles the spectrum of voiced speech.

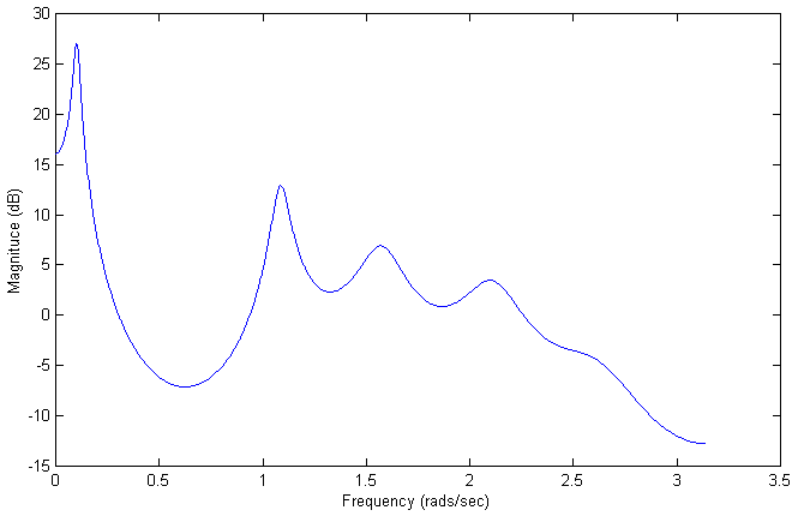

Figure 2.3.6 Spectrum for 10th Order Case.

The gradient auto-correlation matrix for the 10-th order case also shows that it tends to a 2*x*2 block-diagonal matrix. The auto-correlation is again computed from iterations 400 through 4000, so that the transient, which occurs for approximately 350 iterations, is not included. The section with the highest gradient auto-correlation corresponds to pole pair responsible for the highest peak of the spectrum, as seen in Figure 2.3.6. This can be explained using equations (2.3.3a-b). We see in equations (2.3.3a-b) that the closer a_2 is to one, a_2 is the square of the radius of the pole, the larger the auto-correlation of y_n at lag 0, r_0. We also know that the closer the radius of a pole is to 1, the higher the PSD of that pole. The rest of the auto-correlation estimates are smaller. Some of its values, in the top-left 6x6 partition $\hat{\mathbf{R}}_{y,1-6} \equiv \{\hat{r}_{i,j}\}_{i,j=1,\cdots,6}$, are

$$\hat{\mathbf{R}}_{y,1-6} = 10^6 \begin{bmatrix} 6.0865 & 6.0519 & 0.0002 & 0.0041 & 0.0001 & 0.0024 \\ 6.0519 & 6.0821 & -0.0035 & 0.0004 & -0.0021 & 0.0001 \\ 0.0002 & -0.0035 & 0.0330 & 0.0154 & 0.0017 & 0.0055 \\ 0.0041 & 0.0004 & 0.0154 & 0.0330 & -0.0042 & 0.0017 \\ 0.0001 & -0.0021 & 0.0017 & -0.0042 & 0.0109 & 0.0008 \\ 0.0024 & 0.0001 & 0.0055 & 0.0017 & 0.0008 & 0.0109 \end{bmatrix}$$

while those in the bottom-right 6*x*6 auto-correlation matrix, $\hat{\mathbf{R}}_{y,5-10} \equiv \{\hat{r}_{i,j}\}_{i,j=5,\cdots,10}$, are

$$\hat{\mathbf{R}}_{y,5-10} = 10^6 \begin{bmatrix} 0.0109 & 0.0008 & 0.0021 & 0.0034 & 0.0011 & 0.0022 \\ 0.0008 & 0.0109 & -0.0039 & 0.0021 & -0.0028 & 0.0011 \\ 0.0021 & -0.0039 & 0.0087 & -0.0036 & 0.0041 & 0.0015 \\ 0.0034 & 0.0021 & -0.0036 & 0.0087 & -0.0063 & 0.0041 \\ 0.0011 & -0.0028 & 0.0041 & -0.0063 & 0.0160 & -0.0126 \\ 0.0022 & 0.0011 & 0.0015 & 0.0041 & -0.0126 & 0.0160 \end{bmatrix}$$

We see that the bottom-right 6x6 sub-matrix values of the gradient auto-correlation matrix are quite small relative to the values of the top-left 6x6 sub-matrix. The cross-correlations between the gradients with respect to the coefficients of sections 1 and 2 and between the gradients with respect to the coefficients of sections 1 and 5 are shown in Figures 2.3.7 and 2.3.8 respectively. Here we see that the normalized cross-correlation between gradients is also small.

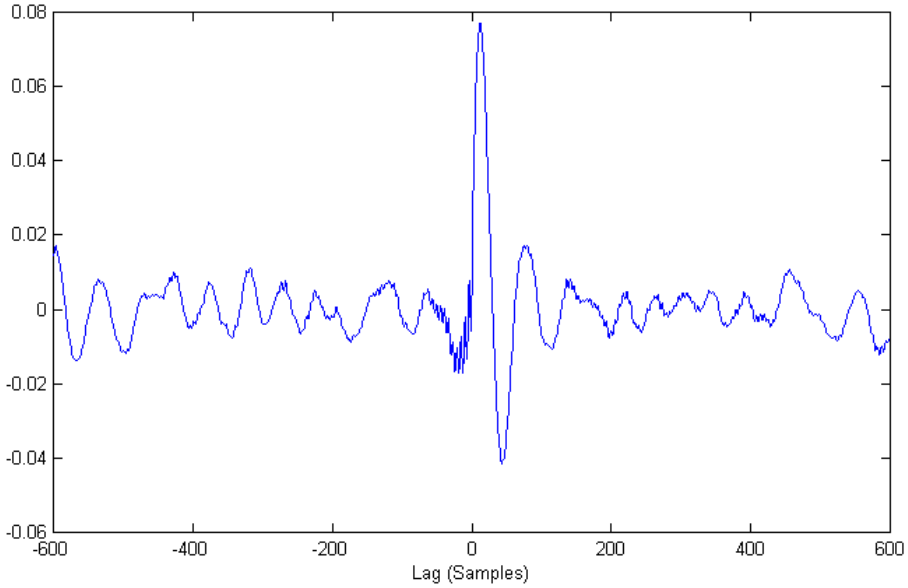

Figure 2.3.7 Normalized Cross-correlation of the 1st and 2th Section Gradients.

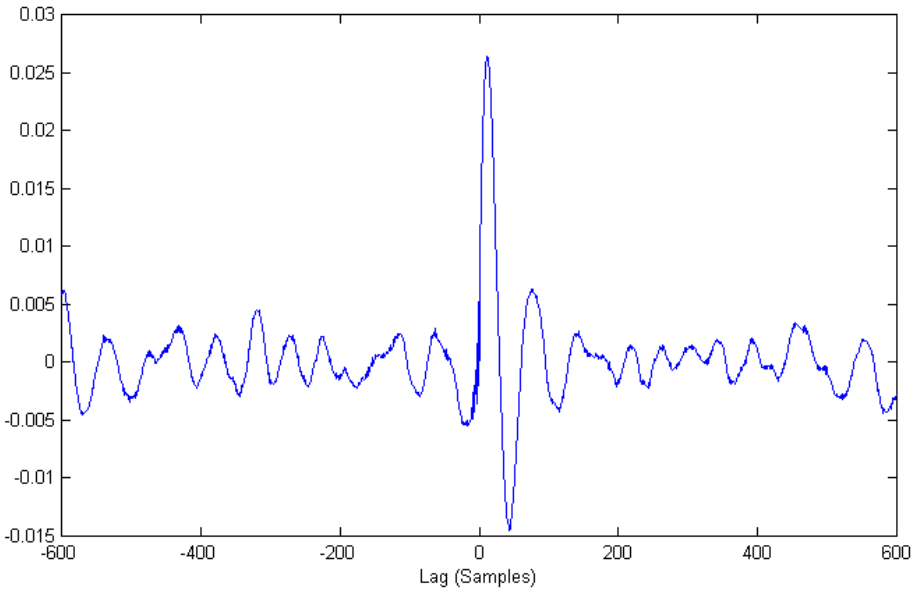

Figure 2.3.8 Normalized Cross-correlation of the 1st and 5th Section Gradients.

The nearly block-diagonal structure of $\hat{\mathbf{R}}_y$ suggests that updating each section of the cascade structure independently, while still being based on the same global minimization, is perhaps not very different from solving the original LS problem. The assumption that the gradient auto-correlation matrix is perfectly block-diagonal leads to the Cascade RLS with Subsection Adaptation (CRLS-SA) algorithm. In CRLS-SA, we ignore the off-diagonal components of the gradient auto-correlation matrix, assuming they are vanishingly small, so that the auto-correlation matrix of the gradient has the following form

$$\hat{\mathbf{R}}_{y_n} = diag\left\{\hat{\mathbf{R}}_{y\ n,1}, \hat{\mathbf{R}}_{y\ n,2}, \cdots, \hat{\mathbf{R}}_{y\ n,M}\right\} \tag{2.3.4}$$

where $\hat{\mathbf{R}}_{y\ n,k}$ is the estimate of the auto-correlation matrix of the gradient of the *k-th* section, computed as

$$\hat{\mathbf{R}}_{y\ n,k} = l\hat{\mathbf{R}}_{y\ n-1,k} + y_{\ n,k} y^{T}_{\ n,k} \tag{2.3.5}$$

and $y_{\ n,k}$ is as defined in (2.2.12b), with *M=N/2* and *N* being the order of the filter.

The block diagram for the CRLS-SA algorithm is shown in Figure 2.3.9 and the CRLS-SA algorithm is summarized in Table 2.3.1. As mentioned above, the gradient of the *k-th* section is obtained by passing the final output $e_{n,M}$ to $\left[1-\hat{A}_k(z)\right]^{-1}$ with the appropriate delay, as shown in Figure 2.3.9. The computational requirements of the CRLS-SA algorithm are about 20**L***N*/2, with *N* being the order of the filter and *L* the length of the data record.

Recall that the gradient of the final section is actually obtained as the input to that section. Thus there is no need for computation associated with the gradient for the final section, and the computational effort is reduced by 2**L*. The resulting computational complexity for CRLS-SA is then 20**L***N*/2-2**L*.

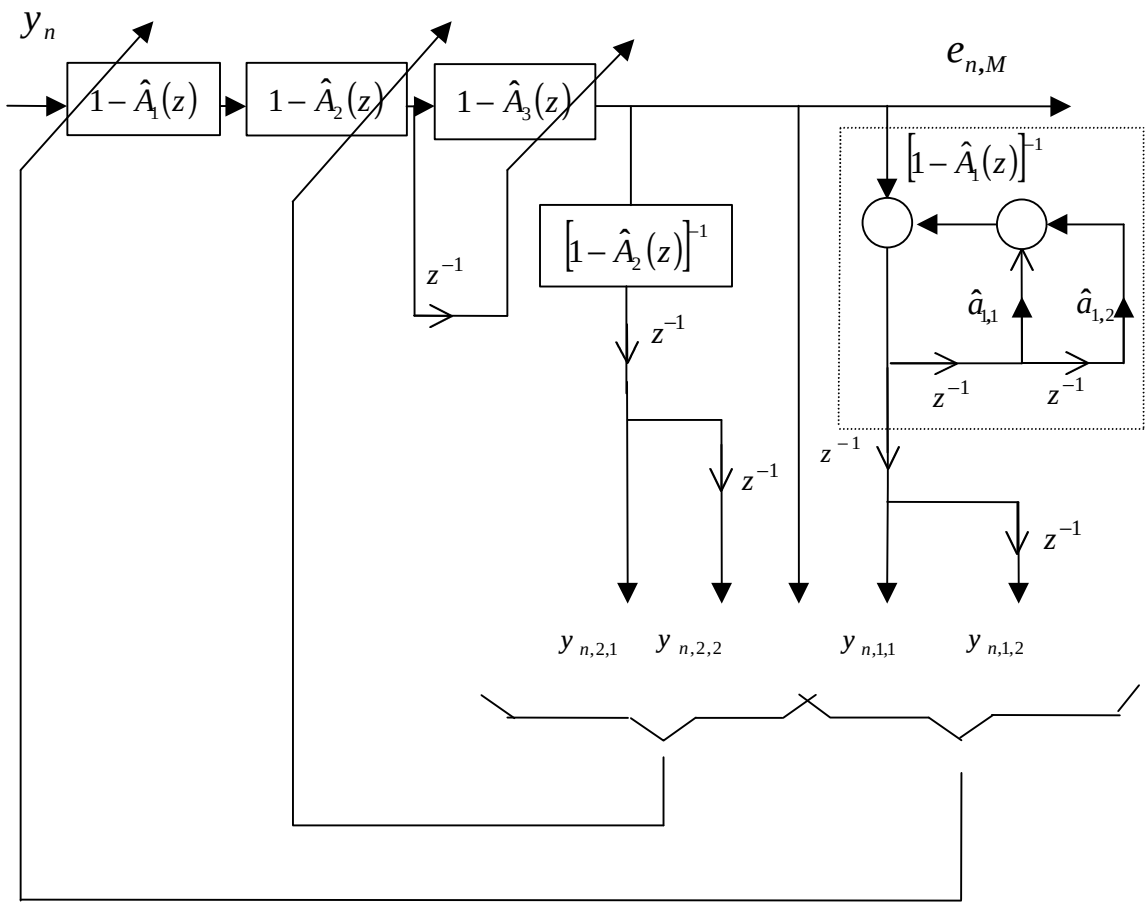

Figure 2.3.9 CRLS-SA and Its Gradient Computation.

Table 2.3.1 CRLS-SA Algorithm

For *k=1,2, …, M,* with *M* being the final section, compute the following

$$y_{n,k,1} = z^{-1}\left[1 - A_k(z)\right]^{-1}\left\{ e_{n,M} \right\} \tag{2.3.6a}$$

$$y_{n,k,2} = y_{n-1,k,1} \tag{2.3.6b}$$

$$y_{n,k} = [y_{n,k,1} \quad y_{n,k,2}]^T \tag{2.3.6c}$$

$$\tilde{\mathbf{P}}_{n-1,k} = l^{-1}\mathbf{P}_{n-1,k} \tag{2.3.6d}$$

$$\breve{\mathbf{P}}_{n-1,k} = \tilde{\mathbf{P}}_{n-1,k}\, y_{n,k} \tag{2.3.6e}$$

$$k_{n,k} = \frac{\breve{\mathbf{P}}_{n-1,k}}{1 + y_{n,k}^H \breve{\mathbf{P}}_{n-1,k}} \tag{2.3.6f}$$

$$\mathbf{P}_{n,k} = \tilde{\mathbf{P}}_{n-1,k} - k_{n,k}\, \breve{\mathbf{P}}_{n-1,k} \tag{2.3.6g}$$

$$e_{n,k} = d_{n,k} - \hat{\boldsymbol{a}}_{n-1,k}^H\, \boldsymbol{y}_{n,k} \tag{2.3.6h}$$

$$\hat{\boldsymbol{a}}_{n,k} = \hat{\boldsymbol{a}}_{n-1,k} + k_{n,k}\, e_{n,M} \tag{2.3.6i}$$