Chapter 2
Introduction to Adaptive Filters

José A. Apolinário Jr. and Sergio L. Netto

Abstract This chapter introduces the general concepts of adaptive filtering and its families of algorithms, and settles the basic notation used in the remaining of the book. Section 2.1 presents the fundamentals concepts, highlighting several configurations, such as system identification, interference cancelation, channel equalization, and signal prediction, in which adaptive filters have been successfully applied. The main objective functions associated to optimal filtering are then introduced in Section 2.2, followed, in Section 2.3, by the corresponding classical algorithms, with emphasis given to the least-mean square, data-reusing, and recursive least-squares (RLS) families of algorithms. It is observed that RLS algorithms based on the so-called QR decomposition combines excellent convergence speed with good numerical properties in finite-precision implementations. Finally, computer simulations are presented in Section 2.4, illustrating some convergence properties of the most important adaptation algorithms. For simplicity, all theoretical developments are performed using real variables, whereas the algorithm pseudo-codes are presented in their complex versions, for generality purposes.

2.1 Basic Concepts

In the last decades, the field of digital signal processing, and particularly adaptive signal processing, has developed enormously due to the increasingly availability of technology for the implementation of the emerging algorithms. These algorithms have been applied to an extensive number of problems including noise and

José A. Apolinário Jr.
Military Institute of Engineering (IME), Rio de Janeiro – Brazil
e-mail: apolin@ieee.org

Sergio L. Netto
Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro – Brazil
e-mail: sergioln@lps.ufrj.br

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_2,

echo canceling, channel equalization, signal prediction, adaptive arrays as well as many others.

An adaptive filter may be understood as a self-modifying digital filter that adjusts its coefficients in order to minimize an error function. This error function, also referred to as the cost function, is a distance measurement between the *reference* or *desired* signal and the output of the adaptive filter.

Adaptive filtering algorithms, which constitute the adjusting mechanism for the filter coefficients, are in fact closely related to classical optimization techniques although, in the latter, all calculations are carried out in an *off-line* manner. Moreover, an adaptive filter, due to its real-time self-adjusting characteristic, is sometimes expected to track the optimum behavior of a slowly varying environment.

In order to compare the wide variety of algorithms available in the literature of adaptive filtering, the following aspects must be taken into account [1–3]:

- Filter structure: The input–output relationship of the adaptive filter depends on its transfer function implementation. Due to its simplicity and efficacy, the most widely employed adaptive structure is by far the transversal filter (or tapped-delay line) associated to standard finite-duration impulse response (FIR) filters. Other structures comprise FIR lattice and infinite-duration impulse response (IIR) filters. This aspect greatly influences the computational complexity of a given adaptive algorithm and the overall speed of the adaptation process.
- Rate of convergence, misadjustment, and tracking: In a noiseless (no measurement or modeling noise) situation, the coefficients of an adaptive filter can be made to converge fast or slowly to the optimum solution. In practice, the adaptive coefficients do not reach the optimum values but stay close to the optimum. Misadjustment is a measure of excess error associated to how close these coefficients (the estimated and the optimum) are to each other in steady-state. It can be taken as a general rule that, for a given algorithm, a faster convergence yields a higher misadjustment. In non-stationary environments, the algorithm convergence speed is also associated to the tracking ability of the adaptive filter.
- Computational aspects: Due to the desired real-time characteristic, the adaptive filter performance must take into account practical levels of computational complexity and limited-precision representation of associated signals and coefficients. The effort in obtaining fast versions of more complex algorithms results from the desire of reducing the computational requirements to a minimal number of operations, as well as reducing the size of memory necessary to run these algorithms in practical applications. On the other hand, a limited-precision environment generates quantization errors which drive the attention of designers to numerical stability, numerical accuracy, and convergence robustness of the algorithm.

The basic configuration of an adaptive filter, operating in the discrete-time domain k, is illustrated in Figure 2.1. In such a scheme, the input signal is denoted

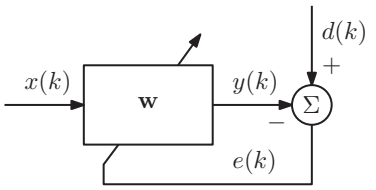


Fig. 2.1 Basic block diagram of an adaptive filter.

by $x(k)$, the reference signal $d(k)$ represents the desired output signal (that usually includes some noise component), $y(k)$ is the output of the adaptive filter, and the error signal is defined as $e(k) = d(k) - y(k)$.

The error signal is used by the adaptation algorithm to update the adaptive filter coefficient vector $\mathbf{w}(k)$ according to some performance criterion. In general, the whole adaptation process aims at minimizing some metric of the error signal, forcing the adaptive filter output signal to approximate the reference signal in a statistical sense.

It is interesting to notice how this basic configuration fits perfectly in several practical applications such as system identification, interference canceling, channel equalization, and signal prediction [1–3], which are detailed as follows.

For instance, Figure 2.2 depicts a typical system identification configuration, where $\mathbf{w}_o$ is an ideal coefficient vector of an unknown plant, whose output is represented by $y_o(k)$, and $n(k)$ denotes the observation or measurement noise. In this setup, the plant and the adaptive filter receive the same input signal. After convergence, the output signals of both systems become similar, and consequently the adaptive transfer function becomes a good model for the input–output relationship of the plant.

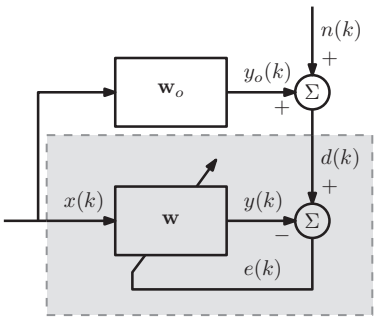


Fig. 2.2 System identification configuration of an adaptive filter: The adaptive coefficient $\mathbf{w}$ vector estimates the unknown system coefficient vector $\mathbf{w}_o$.

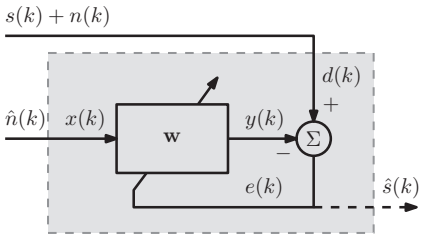


Fig. 2.3 Interference cancelation configuration of an adaptive filter: The error signal $e(k)$ approximates the desired signal component $s(k)$ if $n(k)$ and $\hat{n}(k)$ are correlated.

Another application of an adaptive filter is interference canceling or signal enhancement represented in Figure 2.3. In this problem, a signal of interest $s(k)$ is corrupted by a noise component $n(k)$. A cleaner version of $s(k)$ is desired but cannot be obtained directly in practice. The noisy signal, $s(k)+n(k)$, is then employed as the reference signal for the adaptive filter, whose input must be another version, $\hat{n}(k)$, of the noise signal, strongly correlated to $n(k)$. The adaptive mechanism adjusts the filter coefficients in such a manner that the filter output $y(k)$ approximates $n(k)$, thus forcing the error signal $e(k)$ to resemble signal $s(k)$.

In practical communications systems, a transmitted signal can be heavily distorted by the transmission channel. One may attempt to recover the original signal by employing an adaptive filter in the channel equalization configuration, as depicted in Figure 2.4. In such a framework, a training sequence $s(k)$ known by the receiver is sent via a given channel generating a distorted signal. The same sequence $s(k)$, after a proper time shift to compensate for transmission delays, is used as a reference signal in the receiver for the adaptive filter, whose input is the distorted signal. When the error function approximates zero, the output signal $y(k)$ resembles the transmitted signal $s(k)$, indicating that the adaptive filter is compensating for the channel distortions. After this training process, the desired information can be sent through the channel, which is properly equalized by the adaptive filter.

The adaptive predictor configuration is depicted in Figure 2.5. In this case, the adaptive filter input signal $x(k)$ is a delayed version of the reference signal $d(k)$.

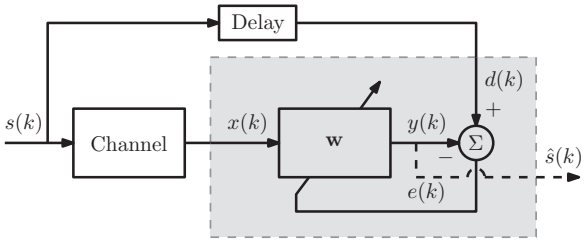


Fig. 2.4 Channel equalization configuration of an adaptive filter: The output signal $y(k)$ estimates the transmitted signal $s(k)$.

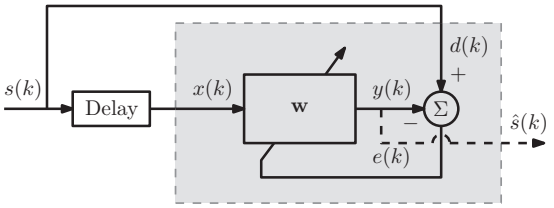


Fig. 2.5 Predictor configuration of an adaptive filter: The output signal $y(k)$ estimates the present input sample $s(k)$ based on past values of this same signal.

Therefore, when the adaptive filter output $y(k)$ approximates the reference, the adaptive filter operates as a predictor system.

From the discussion so far, one observes that the reference signal, through the definition of the error signal, acts as a general guide for the entire adaptation process. The four configurations illustrated above indicate how one can determine the desired output signal in several practical situations. In all cases, one can clearly identify the adaptive filter block given in Figure 2.1. To completely characterize this common basic cell, three main aspects must be defined:

1. Adaptive filter structure: This book will focus on the adaptive transversal FIR structure, whose input–output relationship is described by

$$\begin{aligned} y(k) &= w_0 x(k) + w_1 x(k-1) + \cdots + w_N x(k-N) \\ &= \sum_{i=0}^{N} w_i x(k-i) \\ &= \mathbf{w}^{\mathrm{T}} \mathbf{x}(k), \end{aligned} \tag{2.1}$$

   where N is the filter order and $\mathbf{x}(k)$ and $\mathbf{w}$ are vectors composed by the input-signal samples and the filter coefficients, respectively; that is

$$\mathbf{x}(k) = [x(k)\ x(k-1)\ \ldots\ x(k-N)]^{\mathrm{T}}, \tag{2.2}$$

$$\mathbf{w} = [w_0\ w_1\ \ldots\ w_N]^{\mathrm{T}}. \tag{2.3}$$

   In cases of complex implementations, the output signal is represented as $\mathbf{w}^{\mathrm{H}}\mathbf{x}(k)$, where the superscript H denotes the Hermitian operator (transpose and complex conjugate).
2. Error metric: As mentioned before, the adaptation algorithms adjust the adaptive filter coefficients in an attempt to minimize a given error norm. Different metrics yield adaptation processes with quite distinct characteristics. The commonly employed processes are discussed in detail in Section 2.2.
3. Adaptation algorithm: Several optimization procedures can be employed to adjust the filter coefficients, including, for instance, the least mean-square (LMS) and its normalized version, the data-reusing (DR) including the affine projection

(AP), and the recursive least-squares (RLS) algorithms. All these schemes are discussed in Section 2.3, emphasizing their main convergence and implementation characteristics. The remaining of the book focuses on the RLS algorithms, particularly, those employing QR decomposition, which achieve excellent overall convergence performance.

2.2 Error Measurements

Adaptation of the filter coefficients follows a minimization procedure of a particular objective or cost function. This function is commonly defined as a norm of the error signal $e(k)$. The three most commonly employed norms are the mean-square error (MSE), the instantaneous square error (ISE), and the weighted least-squares (WLS), which are introduced below.

2.2.1 The mean-square error

The MSE is defined as

$$\xi(k) = E[e^2(k)] = E[|d(k) - y(k)|^2]. \tag{2.4}$$

Writing the output signal $y(k)$ as given in Equation (2.1), one obtains

$$\begin{aligned}\xi(k) &= E[|d(k) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(k)|^2] \\ &= E[d^2(k)] - 2\mathbf{w}^{\mathrm{T}}E[d(k)\mathbf{x}(k)] + \mathbf{w}^{\mathrm{T}}E[\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)]\mathbf{w} \\ &= E[d^2(k)] - 2\mathbf{w}^{\mathrm{T}}\mathbf{p} + \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w},\end{aligned} \tag{2.5}$$

where $\mathbf{R}$ and $\mathbf{p}$ are the input-signal correlation matrix and the cross-correlation vector between the reference signal and the input signal, respectively, and are defined as

$$\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)], \tag{2.6}$$

$$\mathbf{p} = E[d(k)\mathbf{x}^{\mathrm{T}}(k)]. \tag{2.7}$$

Note, from the above equations, that $\mathbf{R}$ and $\mathbf{p}$ are not represented as a function of the iteration k or not time-varying, due to the assumed stationarity of the input and reference signals.

From Equation (2.5), the gradient vector of the MSE function with respect to the adaptive filter coefficient vector is given by

$$\nabla_{\mathbf{w}}\xi(k) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}. \tag{2.8}$$

The so-called Wiener solution $\mathbf{w}_o$, that minimizes the MSE cost function, is obtained by equating the gradient vector in Equation (2.8) to zero. Assuming that $\mathbf{R}$ is non-singular, one gets that

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}. \tag{2.9}$$

2.2.2 The instantaneous square error

The MSE is a cost function that requires knowledge of the error function $e(k)$ at all time k. For that purpose, the MSE cannot be determined precisely in practice and is commonly approximated by other cost functions. The simpler form to estimate the MSE function is to work with the ISE given by

$$\hat{\xi}(k) = e^2(k). \tag{2.10}$$

In this case, the associated gradient vector with respect to the coefficient vector is determined as

$$\begin{aligned} \nabla_{\mathbf{w}}\hat{\xi}(k) &= 2e(k)\nabla_{\mathbf{w}}e(k) \\ &= 2e(k)\nabla_{\mathbf{w}}\left[d(k) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(k)\right] \\ &= -2e(k)\mathbf{x}(k). \end{aligned} \tag{2.11}$$

This vector can be seen as a noisy estimate of the MSE gradient vector defined in Equation (2.8) or as a precise gradient of the ISE function, which, in its own turn, is a noisy estimate of the MSE cost function seen in Section 2.2.1.

2.2.3 The weighted least-squares

Another objective function is the WLS function given by

$$\xi_D(k) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(i)]^2 \tag{2.12}$$

where $0 \ll \lambda < 1$ is the so-called *forgetting factor*. The parameter λ^{k-i} emphasizes the most recent error samples (where $i \approx k$) in the composition of the deterministic cost function $\xi_D(k)$, giving to this function the ability of modeling non-stationary processes. In addition, since the WLS function is based on several error samples, its stochastic nature reduces in time, being significantly smaller than the noisy ISE nature as k increases.

Defining the auxiliary variables

$$\mathbf{d}(k) = [d(k)\ \lambda^{1/2}d(k-1)\ \ldots\ \lambda^{k/2}d(0)]^{\mathrm{T}}, \tag{2.13}$$

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{x}^{\mathrm{T}}(k-1) \\ \vdots \\ \lambda^{k/2}\mathbf{x}^{\mathrm{T}}(0) \end{bmatrix}, \tag{2.14}$$

with $\mathbf{x}(k)$ as defined in Equation (2.2), one may rewrite Equation (2.12) as

$$\xi_D(k) = \mathbf{e}^{\mathrm{T}}(k)\mathbf{e}(k), \tag{2.15}$$

where

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}. \tag{2.16}$$

The corresponding WLS gradient vector is given by

$$\begin{aligned} \nabla_{\mathbf{w}}\xi_D(k) &= -2\sum_{i=0}^{k}\lambda^{k-i}\mathbf{x}(i)\left[d(i) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(i)\right] \\ &= -2\mathbf{X}^{\mathrm{T}}(k)\mathbf{d}(k) + 2\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\mathbf{w} \\ &= -2\mathbf{p}(k) + 2\mathbf{R}(k)\mathbf{w}, \end{aligned} \tag{2.17}$$

where

$$\mathbf{R}(k) = \sum_{i=0}^{k}\lambda^{k-i}\mathbf{x}(i)\mathbf{x}^{\mathrm{T}}(i) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k), \tag{2.18}$$

$$\mathbf{p}(k) = \sum_{i=0}^{k}\lambda^{k-i}d(i)\mathbf{x}(i) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{d}(k), \tag{2.19}$$

are the deterministic counterparts of $\mathbf{R}$ and $\mathbf{p}$ defined in Equations (2.6) and (2.7), respectively. The optimum solution $\mathbf{w}(k)$ in the WLS sense is determined by equating the gradient vector $\nabla_{\mathbf{w}}\xi_D(k)$ to zero, yielding

$$\mathbf{w}(k) = \mathbf{R}^{-1}(k)\mathbf{p}(k). \tag{2.20}$$

2.3 Adaptation Algorithms

In this section, a number of schemes are presented to find the optimal filter solution for the error functions seen in Section 2.2. Each scheme constitutes an adaptation algorithm that adjusts the adaptive filter coefficients in order to minimize the associated error norm.

The algorithms seen here can be grouped into three families, namely the LMS, the DR, and the RLS classes of algorithms. Each group presents particular characteristics of computational complexity and speed of convergence, which tend to determine the best possible solution to an application at hand.

2.3.1 LMS and normalized-LMS algorithms

Determining the Wiener solution for the MSE problem requires inversion of matrix $\mathbf{R}$, which makes Equation (2.9) hard to implement in real time. One can then estimate the Wiener solution, in a computationally efficient manner, iteratively adjusting the coefficient vector $\mathbf{w}$ at each time instant k, in such a manner that the resulting sequence $\mathbf{w}(k)$ converges to the desired $\mathbf{w}_o$ solution, possibly in a sufficiently small number of iterations.

The so-called steepest-descent scheme searches for the minimum of a given function following the opposite direction of the associated gradient vector. A factor $\mu/2$, where μ is the so-called convergence factor, adjusts the step size between consecutive coefficient vector estimates, yielding the following updating procedure:

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{\mu}{2}\nabla_{\mathbf{w}}\xi(k). \tag{2.21}$$

This iterative procedure is illustrated in Figure 2.6 for the two-dimensional coefficient vector $\mathbf{w}(k) = [w_0(k)\ w_1(k)]^{\mathrm{T}}$ case.

The Wiener solution requires knowledge of the autocorrelation matrix $\mathbf{R}$ and the cross-correlation vector $\mathbf{p}$. To do that, one must have access to the complete second-order statistics of signals $x(k)$ and $d(k)$, what makes Equation (2.9) unsuitable for most practical applications. Naturally, the Wiener solution can be approximated by a proper estimation of $\mathbf{R}$ and $\mathbf{p}$ based on sufficiently long time intervals, also assuming ergodicity. A rather simpler approach is to approximate the MSE by the ISE function, using the gradient vector of the latter, given in Equation (2.11), to adjust the coefficient vector in Equation (2.21). The resulting algorithm is the LMS algorithm characterized by

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{\mu}{2}\nabla_{\mathbf{w}}\hat{\xi}(k) = \mathbf{w}(k-1) + \mu e(k)\mathbf{x}(k), \tag{2.22}$$

where, in this iterative case,

$$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k). \tag{2.23}$$

The LMS algorithm is summarized in Table 2.1, where the superscripts $*$ and H denote the complex-conjugate and the Hermitian operations, respectively. Although behavior analysis of the LMS algorithm is beyond the scope of this work, it is important to mention that the step-size parameter μ plays an important role in the convergence characteristics of the algorithm as well as in its stability condition.

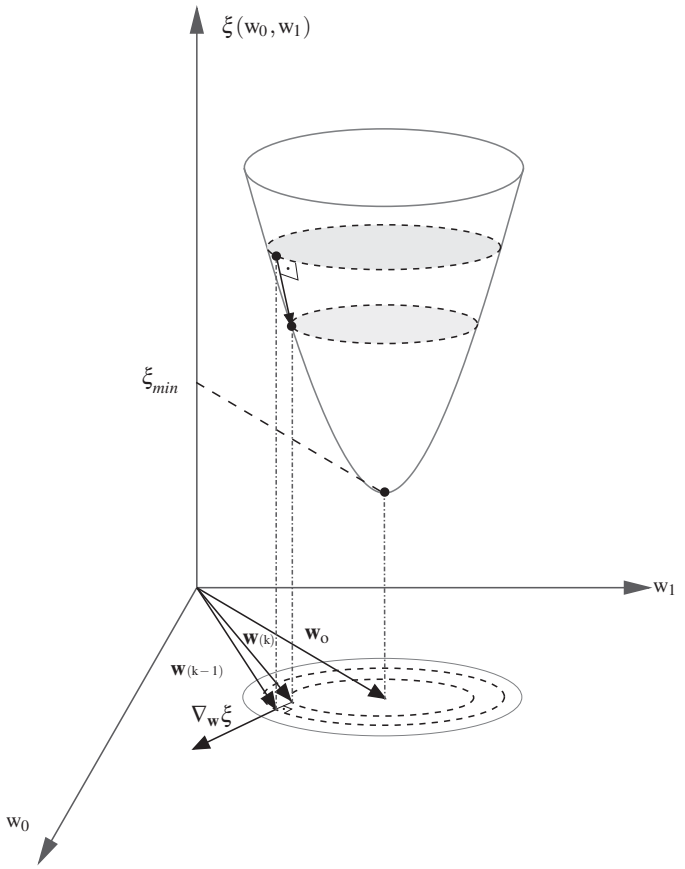


Fig. 2.6 Coefficient updating in a steepest-descent-based algorithm.

Table 2.1 The LMS algorithm.

LMS
Initialize μ for each k $\{\ e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k);$ $\mathbf{w}(k) = \mathbf{w}(k-1) + \mu e^{*}(k)\mathbf{x}(k);$ $\}$

An approximation for the upperbound of this parameter is given in the technical literature and may be stated as [1–3]

$$0 < \mu < \frac{2}{\mathrm{tr}[\mathbf{R}]}, \tag{2.24}$$

where tr[.] denotes the trace operator of a matrix.

The LMS algorithm is very popular and has been widely used due to its extreme simplicity. Its convergence speed, however, is highly dependent on the condition number ρ of the input-signal autocorrelation matrix [1–3], defined as the ratio between the maximum and minimum eigenvalues of this matrix.

Alternative schemes which attempt to improve performance at the cost of minimum additional computational complexity have been proposed and are extensively discussed in [3, 4]. One approach that has been successfully employed in situations where signal statistics are unknown is the on-line calculation of the convergence factor which takes part in updating the filter coefficients [5, 6]. The normalized LMS (NLMS) algorithm can be included in this category [5, 7]. The NLMS algorithm normalizes the convergence factor such that the relation

$$\mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k) = d(k) = \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) + \mu e(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k) \tag{2.25}$$

is always satisfied. This results in a variable step-size parameter given by

$$\mu(k) = \frac{1}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k)}. \tag{2.26}$$

In practice, this parameter is modified to

$$\mu(k) = \frac{\bar{\mu}}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k) + \varepsilon}, \tag{2.27}$$

where another fixed step-size, usually within the range $0 < \bar{\mu} \leq 1$, is used to control misadjustment[1] and convergence speed, and the parameter ε is a very small positive number that avoids possible divisions by zero.[2] Therefore, the NLMS updating equation is given by

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mu(k)e(k)\mathbf{x}(k), \tag{2.28}$$

which is summarized in Table 2.2. In the NLMS algorithm, when $\bar{\mu} = 0$, one has $\mathbf{w}(k) = \mathbf{w}(k-1)$ and the updating halts. When $\bar{\mu} = 1$, the fastest convergence is attained at the price of a higher misadjustment then the one obtained for $0 < \bar{\mu} < 1$. Using $\bar{\mu} > 1$ is not a practical choice (although, theoretically, $\bar{\mu}$ can vary within the range $0 < \bar{\mu} < 2$), since it yields slower convergence rate and an even higher misadjustment than when $\bar{\mu} = 1$. Figure 2.7 depicts the theoretical misadjustment of the NLMS algorithm as a function of $\bar{\mu}$ in two cases of exact-order system identification: with small ($N = 5$) and large ($N = 50$) values of filter order. In both cases, the

[1] The misadjustment is defined as $M = (\xi(\infty) - \xi_{min})/\xi_{min}$ [3], where $\xi(\infty)$ and ξ_{min} denote the steady-state MSE yielded by the adaptation algorithm and the theoretical minimum value for the MSE, respectively.

[2] Some authors name this algorithm the ε-NLMS algorithm when this constant is employed [8].

Table 2.2 The NLMS algorithm.

NLMS
Initialize $\varepsilon \approx 0_+$ and $0 < \bar{\mu} \leq 1$
for each k
$\{\ e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k);$
$\quad \mathbf{w}(k) = \mathbf{w}(k-1) + \frac{\bar{\mu}}{\mathbf{x}^{\mathrm{H}}(k)\mathbf{x}(k)+\varepsilon} e^*(k)\mathbf{x}(k);$
$\}$

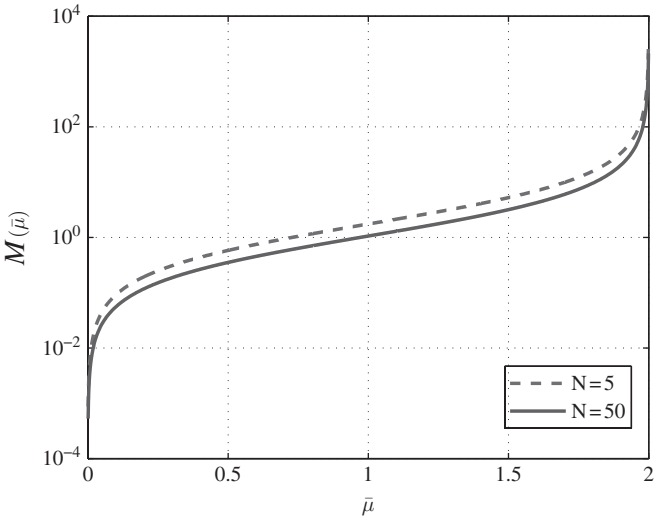


Fig. 2.7 Misadjustment value, as a function of the NLMS convergence factor, using white noise as input signal.

input signal consisted of a zero-mean Gaussian white noise with variance σ_x^2 and the misadjustment was approximated by [5]

$$M(\bar{\mu}) \approx \frac{\bar{\mu}}{2-\bar{\mu}} \frac{N+2}{N-1}. \tag{2.29}$$

2.3.2 Data-reusing LMS algorithms

As remarked before, the LMS algorithm estimates the MSE function with the current ISE value, yielding a noisy adaptation process. In this algorithm, information from each time sample k is disregarded in future coefficient updates. DR algorithms [9–11] employ present and past samples of the reference and input signals to improve convergence characteristics of the overall adaptation process.

For the DR-LMS algorithm, with L data reuses, the coefficients are updated as

$$\mathbf{w}_{i+1}(k) = \mathbf{w}_i(k) + \mu e_i(k)\mathbf{x}(k), \tag{2.30}$$

for $i = 0, 1, \ldots, L$, where

$$e_i(k) = d(k) - \mathbf{w}_i^{\mathrm{T}}(k)\mathbf{x}(k), \tag{2.31}$$

and

$$\mathbf{w}_0(k) = \mathbf{w}(k-1), \tag{2.32}$$
$$\mathbf{w}_{L+1}(k) = \mathbf{w}(k). \tag{2.33}$$

Note that, if $L = 0$, these equations correspond to the LMS algorithm.

As for the LMS algorithm, the DR-LMS also has a normalized-DR (NDR) version, whose updating equation, for L data reuses, is given by

$$\mathbf{w}_{i+1}(k) = \mathbf{w}_i(k) + \frac{e_i(k)}{\mathbf{x}^{\mathrm{T}}(k-i)\mathbf{x}(k-i) + \varepsilon}\mathbf{x}(k-i) \tag{2.34}$$

for $i = 0, \ldots, L$, where

$$e_i(k) = d(k-i) - \mathbf{w}_i^{\mathrm{T}}(k)\mathbf{x}(k-i), \tag{2.35}$$

with $\mathbf{w}_0(k) = \mathbf{w}(k-1)$ and $\mathbf{w}_{L+1}(k) = \mathbf{w}(k)$ as before.

Figure 2.8 provides a geometric interpretation for the coefficient vector upgrade for the LMS, NLMS, DR-LMS, and NDR-LMS algorithms in the two-dimensional case ($N = 1$). In this figure, $\mathscr{S}(k)$ denotes the hyperplane which contains all vectors

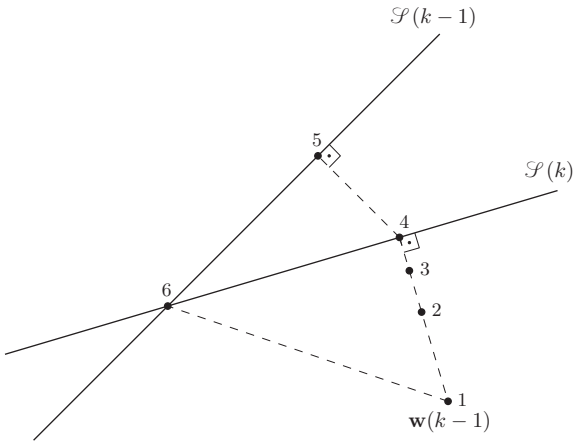


Fig. 2.8 Coefficient vector update for several LMS-type algorithms: 1. Initial vector $\mathbf{w}(k-1)$; 2. LMS and intermediary DR-LMS; 3. DR-LMS ($L = 1$); 4. NLMS and DR-LMS (for $L \to \infty$); 5. NDR-LMS; 6. BNDR-LMS.

$\mathbf{w}$ such that $\mathbf{w}^{\mathrm{T}}\mathbf{x}(k) = d(k)$ and the initial coefficient vector $\mathbf{w}(k-1)$ is indicated by position 1. In a noise-free exact-order modeling situation, $\mathscr{S}(k)$ would contain the optimal coefficient vector $\mathbf{w}_o$. In this scenario, it can be verified that $\mathbf{x}(k)$ and, consequently, the ISE gradient vector, is orthogonal to the hyperplane $\mathscr{S}(k)$. The LMS algorithm takes a single step towards $\mathscr{S}(k)$ yielding a new position represented by point 2 in Figure 2.8. The NLMS algorithm performs a line search in the direction of $\mathbf{x}(k)$ to reach the position represented by point 4, which belongs to $\mathscr{S}(k)$, in a unique iteration. The DR-LMS algorithm iteratively approaches $\mathscr{S}(k)$ by taking successive steps (within a single iteration) in the direction of $\mathbf{x}(k)$, as indicated in Equations (2.30) and (2.31). In Figure 2.8, with $L = 1$, positions 2 and 3 indicate the intermediary and final DR-LMS positions. It can be verified that the DR-LMS algorithm reaches $\mathscr{S}(k)$ in the limit, as the number of data reuses L approaches infinity [10, 11], turning this algorithm similar to the NLMS algorithm. From Equations (2.34) and (2.35), the NDR algorithm [11] employs more than one hyperplane, that is, uses more data pairs $\{\mathbf{x}(k-i), d(k-i)\}$, with $i > 0$, to adjust the coefficient vector $\mathbf{w}(k)$, closer to $\mathbf{w}_o$ than the adjustment obtained with only the current data pair $\{\mathbf{x}(k), d(k)\}$. In Figure 2.8, the NDR update is represented by point 5. Position 6 corresponds to the binormalized data-reusing (BNDR) LMS [4] algorithm addressed below.

For a noise-free exact-order modeling situation, the MSE optimal solution $\mathbf{w}_o$ is at the intersection of $(N+1)$ hyperplanes determined by $(N+1)$ linearly independent input-signal vectors. In this case, an orthogonal-projections algorithm [12] would reach $\mathbf{w}_o$ in exact $(N+1)$ iterations. This algorithm may be viewed as an orthogonal-NDR algorithm that performs exact line searches in $(N+1)$ orthogonal directions determined by the data pairs $\{\mathbf{x}(k-i), d(k-i)\}$, for $i = 0, 1, \ldots, N$. The BNDR algorithm [4] employs normalization on two orthogonal directions obtained from consecutive data pairs within each iteration. In simulations carried out with colored input signals, this algorithm presents faster convergence than all other data-reusing algorithms for the case of two data pairs, or, equivalently, $L = 1$ data reuse.

In order to state the BNDR problem, one may note that the solution which belongs to $\mathscr{S}(k)$ and $\mathscr{S}(k-1)$ at a minimum distance from $\mathbf{w}(k-1)$ is the one characterized by

$$\mathbf{w}(k) = \min_{\mathbf{w}} \|\mathbf{w} - \mathbf{w}(k-1)\|^2, \tag{2.36}$$

subject to

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}(k) = d(k), \tag{2.37}$$

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}(k-1) = d(k-1). \tag{2.38}$$

Using Lagrange multipliers, the constraints above can be incorporated into Equation (2.36), resulting in the modified objective function

$$\begin{aligned} f[\mathbf{w}] = \|\mathbf{w} - \mathbf{w}(k-1)\|^2 &+ \lambda_1 \left[d(k) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(k)\right] \\ &+ \lambda_2 \left[d(k-1) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(k-1)\right], \end{aligned} \tag{2.39}$$

which, for linearly independent input-signal vectors $\mathbf{x}(k)$ and $\mathbf{x}(k-1)$, has the unique solution

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \frac{\lambda_1}{2}\mathbf{x}(k) + \frac{\lambda_2}{2}\mathbf{x}(k-1), \tag{2.40}$$

where

$$\frac{\lambda_1}{2} = \frac{e(k)\|\mathbf{x}(k-1)\|^2 - \varepsilon(k-1)\mathbf{x}^{\mathrm{T}}(k-1)\mathbf{x}(k)}{\|\mathbf{x}(k)\|^2\|\mathbf{x}(k-1)\|^2 - [\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k-1)]^2}, \tag{2.41}$$

$$\frac{\lambda_2}{2} = \frac{\varepsilon(k-1)\|\mathbf{x}(k)\|^2 - e(k)\mathbf{x}^{\mathrm{T}}(k-1)\mathbf{x}(k)}{\|\mathbf{x}(k)\|^2\|\mathbf{x}(k-1)\|^2 - [\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k-1)]^2}, \tag{2.42}$$

with $e(k)$ as in Equation (2.23) and

$$\varepsilon(k-1) = d(k-1) - \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k-1). \tag{2.43}$$

The BNDR derivation presented above is valid for any $\mathbf{w}(k-1)$, which may or may not belong to $\mathscr{S}(k-1)$. However, if successive optimized steps are taken for $\mathbf{w}(k)$ for all k, then

$$\mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k-1) = d(k-1), \tag{2.44}$$

corresponding to $\varepsilon(k-1) = 0$, and a simplified version of the BNDR algorithm results:

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \frac{\lambda_1'}{2}\mathbf{x}(k) + \frac{\lambda_2'}{2}\mathbf{x}(k-1), \tag{2.45}$$

where

$$\frac{\lambda_1'}{2} = \frac{e(k)\|\mathbf{x}(k-1)\|^2}{\|\mathbf{x}(k)\|^2\|\mathbf{x}(k-1)\|^2 - [\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k-1)]^2}, \tag{2.46}$$

$$\frac{\lambda_2'}{2} = \frac{-e(k)\mathbf{x}^{\mathrm{T}}(k-1)\mathbf{x}(k)}{\|\mathbf{x}(k)\|^2\|\mathbf{x}(k-1)\|^2 - [\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k-1)]^2}. \tag{2.47}$$

It is worth mentioning that the excess MSE for either implementation of the BNDR-LMS algorithm, as in (2.40, 2.41, 2.42) or in (2.45, 2.46, 2.47), is close to the observation noise power when there is no modeling error, as expected from normalized algorithms. In order to control this excess MSE, a convergence parameter μ may be introduced into the algorithm, forcing a trade-off between convergence rate, maximized with $\mu = 1$, and lower steady-state MSE, associated to smaller values of μ, which may be required in applications with significant measurement error. With the introduction of μ, the coefficient vector $\mathbf{w}(k)$ at each iteration is not at the exact intersection of hyperplanes $\mathscr{S}(k-1)$ and $\mathscr{S}(k)$ and, therefore, the simplified version of the algorithm given by (2.45, 2.46, 2.47) should not be used.

If $\mathbf{x}(k)$ and $\mathbf{x}(k-1)$ are linearly dependent, then $\mathscr{S}(k)$ is parallel to $\mathscr{S}(k-1)$, and $\mathbf{w}(k) = \mathbf{w}_1(k)$, which corresponds to the NLMS algorithm for any step-size value μ. Particularly when $\mu = 1$, it is also correct to say that $\mathbf{w}(k-1)$ is already on the hyperplane $\mathscr{S}(k-1)$.

Table 2.3 The BNDR-LMS algorithm.

BNDR-LMS
Initialize $\varepsilon \approx 0_+$
for each k
$\{\ e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k);$
$\alpha = \mathbf{x}^{\mathrm{H}}(k)\mathbf{x}(k-1);$
$\rho(k) = \mathbf{x}^{\mathrm{H}}(k)\mathbf{x}(k);$
$D = \rho(k)\rho(k-1) - \lvert\alpha\rvert^2;$
if $D < \varepsilon$
$\{\ \mathbf{w}(k) = \mathbf{w}(k-1) + \mu e^*(k)\mathbf{x}(k)/\rho(k);$
$\}$
else
$\{\ \varepsilon(k-1) = d(k-1) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k-1);$
$\frac{\lambda_1}{2} = [e^*(k)\rho(k-1) - \varepsilon^*(k-1)\alpha]/D;$
$\frac{\lambda_2}{2} = [\varepsilon^*(k-1)\rho(k) - e^*(k)\alpha^*]/D;$
$\mathbf{w}(k) = \mathbf{w}(k-1) + \mu\left[\frac{\lambda_1}{2}\mathbf{x}(k) + \frac{\lambda_2}{2}\mathbf{x}(k-1)\right];$
$\}$
$\}$

The BNDR-LMS algorithm is summarized in Table 2.3, where the denominator D in Equations (2.46) and (2.47) is determined recursively in time.

In Figure 2.8, the updating process was represented for a two-dimensional ($N = 1$) weight vector. In such a case, only one solution for $\mathbf{w}(k)$ was available as the intersection of two lines. In a more practical situation, the BNDR-LMS algorithm decreases in two the degree of freedom of $\mathbf{w}(k)$ by choosing the closest solution to $\mathbf{w}(k-1)$, following the least perturbation property [8] or the principle of minimal disturbance [2]. The three-dimensional case is depicted in Figure 2.9, where the updated coefficient vector $\mathbf{w}(k)$ is chosen closest to $\mathbf{w}(k-1)$

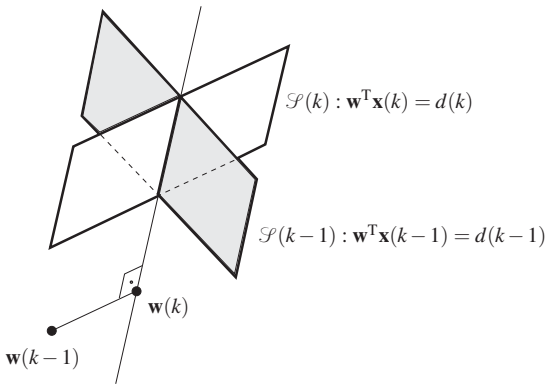


Fig. 2.9 Choice of the BNDR-LMS updated coefficient vector as the intersection of two hyperplanes, according to the minimal disturbance principle, assuming that $\mathbf{w}(k-1)$ does not belong to $\mathscr{S}(k-1)$.

and belonging to the intersection of the hyperplanes $\mathscr{S}(k) \cap \mathscr{S}(k-1)$. If one considers a third hyperplane, $\mathscr{S}(k-2)$ for instance, performing normalization in three directions, this would determine one single possibility for $\mathbf{w}(k)$, given by $\mathscr{S}(k) \cap \mathscr{S}(k-1) \cap \mathscr{S}(k-2)$.

As a generalization of the previous idea, the AP algorithm [13–15] is among the prominent adaptation algorithms that allow trade-off between fast convergence and low computational complexity. By adjusting the number of projections, or alternatively, the number of data reuses, one obtains adaptation processes ranging from that of the NLMS algorithm to that of the sliding-window RLS algorithm [16, 17].

The AP algorithm updates its coefficient vector such that the new solution belongs to the intersection of L hyperplanes defined by the present and the $(L-1)$ previous data pairs. The optimization criterion used for the derivation of the AP algorithm is given by

$$\mathbf{w}(k) = \min_{\mathbf{w}} \|\mathbf{w} - \mathbf{w}(k-1)\|^2, \tag{2.48}$$

subject to

$$\mathbf{d}_L(k) = \mathbf{X}_L^{\mathrm{T}}(k)\mathbf{w}, \tag{2.49}$$

where

$$\mathbf{d}_L(k) = [d(k)\ d(k-1)\ \ldots\ d(k-L+1)]^{\mathrm{T}}, \tag{2.50}$$

$$\mathbf{X}_L(k) = [\mathbf{x}(k)\,\mathbf{x}(k-1)\,\ldots\,\mathbf{x}(k-L+1)]. \tag{2.51}$$

The updating equations for the AP algorithm obtained as the solution to the minimization problem in (2.48) are presented in Table 2.4 [13, 14]. To control stability, convergence speed, and misadjustment, a convergence factor, usually constrained to $0 < \mu < 1$, is introduced. In order to improve robustness, a diagonal matrix $\varepsilon\mathbf{I}$, with $\varepsilon > 0$, is employed to regularize the inverse operation required by the AP algorithm.

Table 2.4 The AP algorithm.

APA
Initialize $\varepsilon \approx 0_+$ for each k { $\mathbf{e}_L(k) = \mathbf{d}_L(k) - \mathbf{X}_L^{\mathrm{T}}(k)\mathbf{w}^*(k-1)$; $\mathbf{t}_k = \left[\mathbf{X}_L^{\mathrm{H}}(k)\mathbf{X}_L(k) + \varepsilon\mathbf{I}\right]^{-1}\mathbf{e}_L^*(k)$; $\mathbf{w}(k) = \mathbf{w}(k-1) + \mu\mathbf{X}_L(k)\mathbf{t}_k$; }

2.3.3 RLS-type algorithms

This subsection presents the basic versions of the RLS family of adaptive algorithms. Importance of the expressions presented here cannot be overstated for they allow an easy and smooth reading of the forthcoming chapters.

The RLS-type algorithms have a high convergence speed which is independent of the eigenvalue spread of the input correlation matrix. These algorithms are also very useful in applications where the environment is slowly varying. The price of all these benefits is a considerable increase in the computational complexity of the algorithms belonging to the RLS family.

In order to obtain the equations of the conventional RLS algorithm, the deterministic correlation matrix and cross-correlation vector defined in Equations (2.18) and (2.19), respectively, are rewritten as

$$\mathbf{R}(k) = \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) + \lambda\mathbf{R}(k-1), \tag{2.52}$$

$$\mathbf{p}(k) = \mathbf{x}(k)d(k) + \lambda\mathbf{p}(k-1). \tag{2.53}$$

Using these recursive expressions into Equation (2.20), the following development can be made:

$$\begin{aligned}
\mathbf{w}(k) &= \mathbf{R}^{-1}(k)\left[\mathbf{x}(k)d(k) + \lambda\mathbf{p}(k-1)\right] \\
&= \mathbf{R}^{-1}(k)\left[\mathbf{x}(k)d(k) + \lambda\mathbf{R}(k-1)\mathbf{w}(k-1)\right] \\
&= \mathbf{R}^{-1}(k)\left[\mathbf{x}(k)d(k) + \lambda\mathbf{R}(k-1)\mathbf{w}(k-1)\right. \\
&\quad \left. + \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1) - \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)\right] \\
&= \mathbf{R}^{-1}(k)\left\{\mathbf{x}(k)\left[d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)\right] + \mathbf{R}(k)\mathbf{w}(k-1)\right\} \\
&= \mathbf{R}^{-1}(k)\left[\mathbf{x}(k)e(k) + \mathbf{R}(k)\mathbf{w}(k-1)\right],
\end{aligned} \tag{2.54}$$

and then

$$\mathbf{w}(k) = \mathbf{w}(k-1) + e(k)\mathbf{R}^{-1}(k)\mathbf{x}(k). \tag{2.55}$$

In this updating expression, the computational burden for determining the inverse matrix $\mathbf{R}^{-1}(k)$ can be reduced significantly by employing the *matrix inversion lemma*[3] [3], which, in this case, yields that

$$\mathbf{R}^{-1}(k) = \frac{1}{\lambda}\left[\mathbf{R}^{-1}(k-1) - \frac{\mathbf{R}^{-1}(k-1)\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k-1)}{\lambda + \mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}\right]. \tag{2.56}$$

[3] For suitable matrix orders, $[\mathbf{A}+\mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1}]^{-1}\mathbf{DA}^{-1}$.

For convenience of notation, consider the following auxiliary vectors:

$$\boldsymbol{\kappa}(k) = \mathbf{R}^{-1}(k)\mathbf{x}(k), \tag{2.57}$$
$$\mathbf{k}(k) = \mathbf{R}^{-1}(k-1)\mathbf{x}(k), \tag{2.58}$$

which in conjunction to Equation (2.56) yield that

$$\boldsymbol{\kappa}(k) = \frac{\mathbf{k}(k)}{\lambda + \mathbf{x}^{\mathrm{T}}(k)\mathbf{k}(k)}. \tag{2.59}$$

Hence, by substituting Equation (2.59) into Equation (2.56), one gets that

$$\begin{aligned}\mathbf{R}^{-1}(k) &= \frac{1}{\lambda}\left[\mathbf{R}^{-1}(k-1) - \mathbf{R}^{-1}(k)\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k-1)\right] \\ &= \frac{1}{\lambda}\left[\mathbf{R}^{-1}(k-1) - \boldsymbol{\kappa}(k)\mathbf{k}^{\mathrm{T}}(k)\right],\end{aligned} \tag{2.60}$$

and that the conventional RLS algorithm, requiring $\mathcal{O}[N^2]$ multiplications, can be implemented as indicated in Table 2.5. A few RLS-type algorithms requiring only $\mathcal{O}[N]$ multiplications, such as the fast transversal (FT) [18] and the lattice (L) [19] RLS algorithms, can be found in the technical literature.

An alternative RLS implementation with good numerical properties employs the so-called QR decomposition in the triangularization of the input data matrix combined with a back-substitution procedure [3] to obtain the coefficient vector.

It is worth mentioning that matrix $\mathbf{X}(k)$ is $(k+1)\times(N+1)$, which means that its order increases as the iterations progress. The QR-decomposition process applies an orthogonal matrix $\mathbf{Q}(k)$ of order $(k+1)\times(k+1)$ to transform $\mathbf{X}(k)$ into a triangular matrix $\mathbf{U}(k)$ of order $(N+1)\times(N+1)$ such that

$$\mathbf{Q}(k)\mathbf{X}(k) = \begin{bmatrix}\mathbf{O} \\ \mathbf{U}(k)\end{bmatrix} \tag{2.61}$$

where $\mathbf{O}$ is a null matrix of order $(k-N)\times(N+1)$. Matrix $\mathbf{Q}(k)$ represents the overall triangularization process and may be implemented in different ways as, for

Table 2.5 The RLS algorithm.

RLS
Initialize $0 \ll \lambda < 1$, $\varepsilon \approx 1/\sigma_x^2$, and $\mathbf{R}^{-1}(0) = \varepsilon\mathbf{I}$
for each k
{ $e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$;
$\mathbf{k}(k) = \mathbf{R}^{-1}(k-1)\mathbf{x}(k)$;
$\boldsymbol{\kappa}(k) = \frac{\mathbf{k}(k)}{\lambda + \mathbf{x}^{\mathrm{H}}(k)\mathbf{k}(k)}$;
$\mathbf{R}^{-1}(k) = \frac{1}{\lambda}\left[\mathbf{R}^{-1}(k-1) - \frac{\boldsymbol{\kappa}(k)\boldsymbol{\kappa}^{\mathrm{H}}(k)}{\lambda + \mathbf{x}^{\mathrm{H}}(k)\mathbf{k}(k)}\right]$;
$\mathbf{w}(k) = \mathbf{w}(k-1) + e^*(k)\boldsymbol{\kappa}(k)$;
}

instance, the numerically well-conditioned Givens rotations [2, 3, 8] or the Householder transformation [20, 21].

The main advantages associated to the QR-decomposition RLS (QRD-RLS) algorithms, as opposed to their conventional RLS counterpart, are the possibility of implementation in systolic arrays and the improved numerical behavior in limited precision environment.

The basic QRD [2, 3] and the so-called inverse QR (IQR) [2, 3, 22] RLS algorithms have a computational requirement of $\mathcal{O}[N^2]$ multiplications. A number of QR-based RLS versions requiring only $\mathcal{O}[N]$ multiplications, such as the fast QR (FQR) [23–28], the fast QR-Lattice (FQR-L) [29–31], are also available in the technical literature.

Since the scope of the next chapter encompasses both the basic (or conventional) and the inverse QRD-RLS algorithms, tables with their equations were not included at this point but left to be presented with their derivations.

2.4 Computer Simulations

This section presents simulations of system identification using some of the adaptation algorithms previously presented. In this framework, a few issues, such as misadjustment, convergence speed, tracking performance, and algorithm stability, are addressed in a practical point of view.

In the system identification setup implemented here, the plant is described by the transfer function $H(z) = -1 + 3z^{-1}$, such that $N = 1$ and $\mathbf{w}_o = [-1 \;\; 3]^{\mathrm{T}}$. An adaptive filter of correct order (exact modeling) is employed to identify $H(z)$. Therefore, $\mathbf{w}(k) = [w_0(k) \;\; w_1(k)]^{\mathrm{T}}$. In practical situations, under-modeling or over-modeling is usually a problem that requires special attention of the system designer [3]. In all cases, measurement noise $n(k)$, consisting of a zero-mean white Gaussian noise with variance $\sigma_n^2 = 10^{-6}$ and uncorrelated to the input signal $x(k)$, is added to the plant output signal.

2.4.1 Example 1: Misadjustment of the LMS algorithm

In a first example, consider that a zero-mean unit-variance white Gaussian noise is used as the input signal $x(k)$. The average ISE over an ensemble of 400 experiments is shown in Figure 2.10 for the LMS algorithm with $\mu = 0.22$ and $\mu = 0.013$. From this figure, one clearly notices how the step-size μ controls the trade-off between convergence speed and steady-state excess of average ISE (which approximates the MSE if the ensemble is large enough) which characterizes the misadjustment.

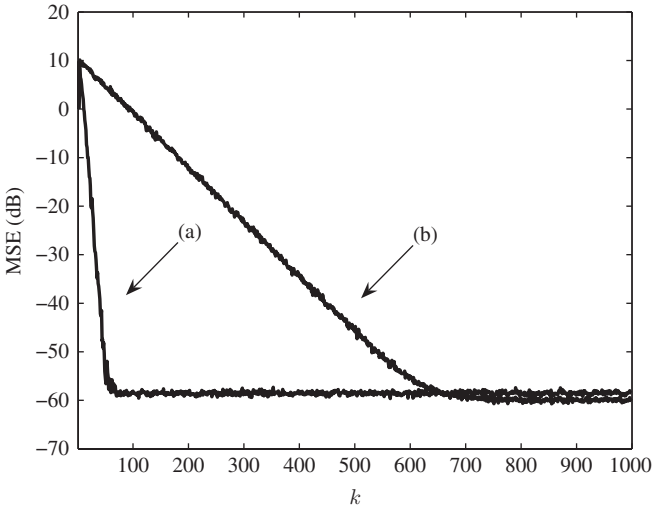


Fig. 2.10 Example 1: MSE convergence (*learning curve*) for the LMS algorithm with: (a) $\mu = 0.22$; (b) $\mu = 0.013$. The larger step-size parameter yields faster convergence and higher misadjustment.

2.4.2 Example 2: Convergence trajectories

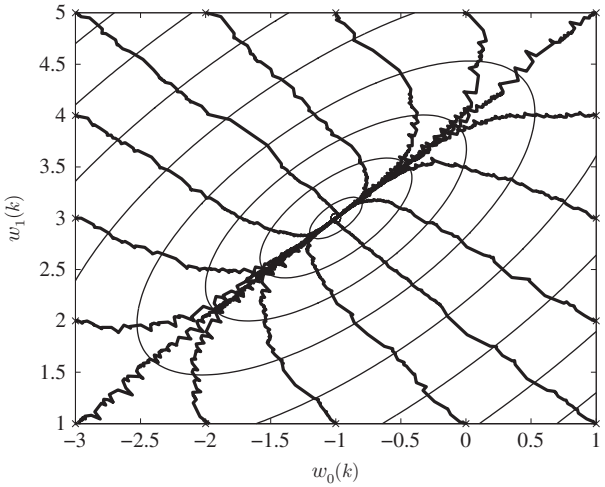
In the previous case, since $x(k)$ consisted of a white noise, the LMS algorithm could achieve a very fast convergence regardless the initial position of the adaptive filter coefficient vector. If, however, the input signal is colored, the LMS convergence process becomes highly dependent on the initial value of $\mathbf{w}(k)$. Let then $x(k)$ be obtained by processing a zero-mean unit-variance white Gaussian noise with the filter $H(z) = 1/(1+1.2z^{-1}+0.81z^{-2})$. The resulting input signal autocorrelation matrix $\mathbf{R}$ presents a condition number around $\rho = 5$.

The LMS ($\mu = 0.005$) and RLS ($\lambda = 0.95$, $\varepsilon = 1$) coefficient trajectories for this setup are seen in Figure 2.11, where the crosses indicate distinct initial conditions for the adaptive coefficient vector and the circle indicates the optimum value $\mathbf{w}_o = [-1\ \ 3]^{\mathrm{T}}$. From this figure, one observes how the LMS trajectories converge first to the main axis of the elliptic MSE contour lines, whereas the RLS trajectories follow a straight path to $\mathbf{w}_o$ regardless the initial value of $\mathbf{w}(k)$.

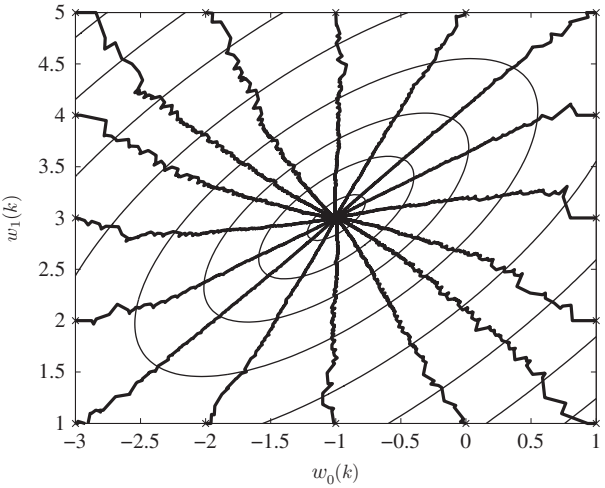
2.4.3 Example 3: Tracking performance

In this example, the adaptive filter performance is evaluated in a non-stationary environment. The idea is to perform an adaptive equalization of a channel, whose impulse response is initially given by

$$h_1(k) = e^{-k/5}[u(k) - u(k-5)], \tag{2.62}$$



(a)



(b)

Fig. 2.11 Example 2: Convergence of adaptive coefficient vector – crosses indicate the initial position and circle indicates the MSE optimal solution: (a) LMS algorithm; (b) RLS algorithm.

where $u(k)$ is the unitary step sequence. It is then assumed that, for iteration 3000 on, the channel impulse response suddenly changes to

$$h_2(k) = h_1(k)e^{j\pi k}. \tag{2.63}$$

The adaptive filter order was set to $N = 49$, and the delay block, as in Figure 2.4, was set to $\Delta = 5$. A very small amount of additive noise (zero-mean white Gaussian

noise with variance $\sigma_n^2 = 10^{-8}$) was included in the reference signal $d(k)$ to highlight the performance of the equalizer. The resulting MSE (average ISE over an ensemble of 1000 independent runs) is presented in Figure 2.12 for the NLMS, the BNDR-LMS, the AP (with $L = 5$), and the RLS (with $\lambda = 0.98$) algorithms, where in all cases the step-size was set to 1.

From Figure 2.12, one can conclude that all algorithms converge properly despite the sudden change in the environment. As expected, the algorithms considered converge with different speeds (the RLS algorithm being the fastest one) and misadjustment levels (the AP algorithm presenting the highest misadjustment amongst the LMS-type algorithms, since it uses $L = 5$, whereas the BNDR-LMS has $L = 2$ and the NLMS corresponds to having $L = 1$).

To verify the algorithm tracking ability, the channel was made to vary continuously. In this case, the impulse response was set to correspond to a five-tap Rayleigh fading channel with sampling frequency 1.25 MHz and Doppler frequency equal to 50 Hz. The 500 transmitted symbols are equalized by the same adaptive algorithms used in Figure 2.12, the only difference being that the RLS forgetting factor was set to $\lambda = 0.995$ in the present case. The resulting time-varying channel taps and the algorithm learning curves are shown in Figure 2.13. From this figure, one may observe that the RLS algorithm, although presenting a fast convergence, is not so well tuned for this application, since its MSE increases as time goes by. The reason for this is the forgetting factor λ, which should be decreased in order for the algorithm to remember only the most recent samples. However, this reduction cannot be implemented for the conventional RLS algorithm, as will be verified in Section 2.4.4.

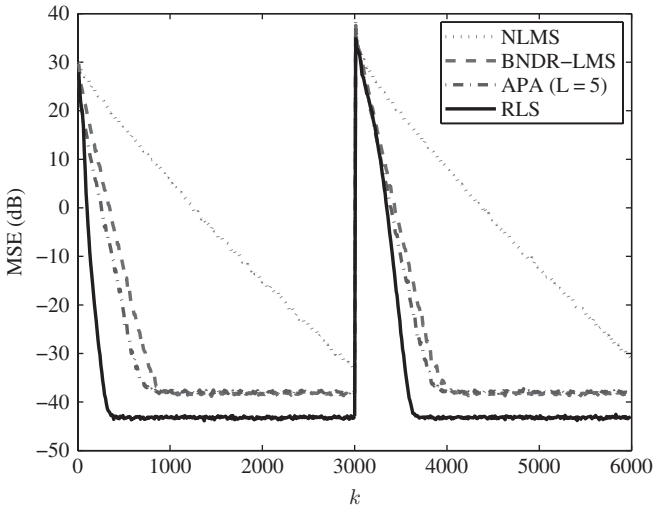


Fig. 2.12 Example 3: Resulting MSE in non-stationary channel equalization for the NLMS, BNDR-LMS, AP, and RLS algorithms.

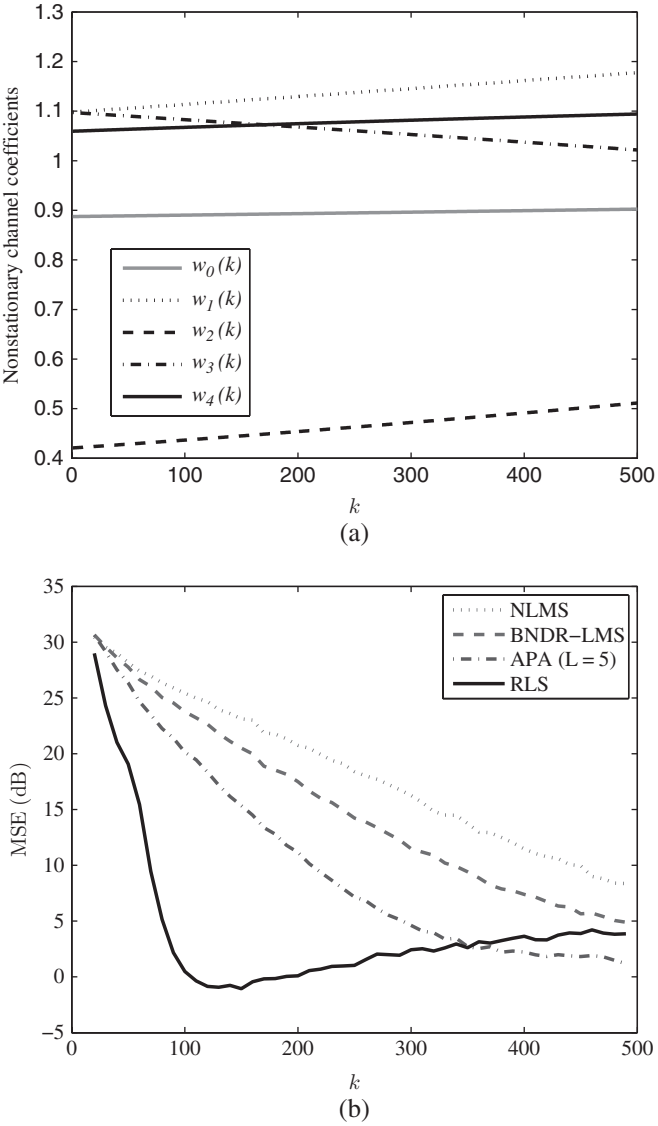


Fig. 2.13 Example 3: (a) Time-varying channel coefficients; (b) Resulting MSE in non-stationary channel equalization for the NLMS, BND-LMS, AP, and RLS algorithms.

2.4.4 Example 4: Algorithm stability

Although, as previously seen, the conventional RLS algorithm is fast converging, presents low misadjustment, and is suitable for time-varying environments, a last experiment shows its vulnerability: the lack of numerical stability. In an

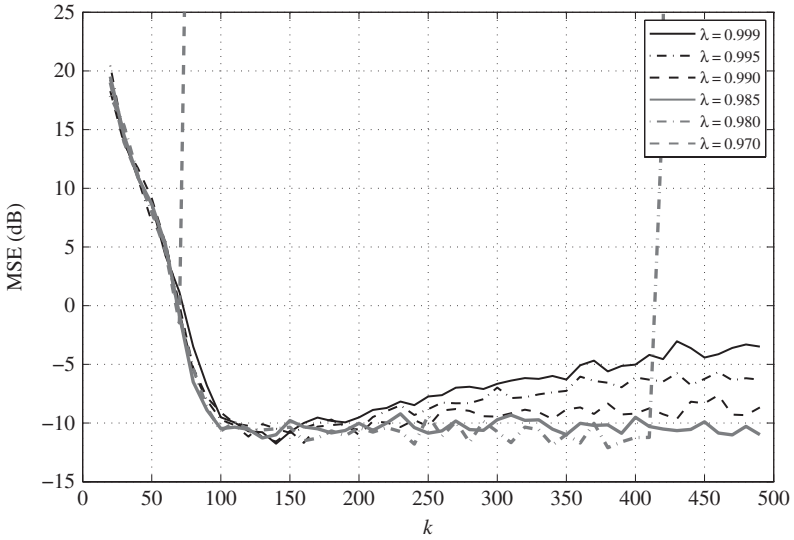


Fig. 2.14 Example 4: Resulting MSE for the RLS algorithm, with different values of the forgetting factor, illustrating how small values of λ may lead to algorithm divergence.

attempt to improve its performance in the non-stationary channel equalization of the last experiment, the RLS algorithm was employed with different values of forgetting factor. The associated learning curves for each value of λ within the set $\{0.999, 0.995, 0.99, 0.985, 0.98, 0.97\}$ are presented in Figure 2.14. From this figure, one can conclude that decreasing the value of λ improves the RLS performance (meaning that the algorithm can track better the channel variation) in this case. However, if one greatly reduces the value of λ, the RLS algorithm may diverge due to the time-varying nature of this example. For instance, when $\lambda = 0.98$, the adaptation process diverges a bit after 400 iterations, whereas the divergence occurs even sooner when $\lambda = 0.97$.

This example brings to light an important issue of an adaptive algorithm: its numerical stability. As seen above, the conventional RLS algorithm, besides its high computational complexity, also presents numerical stabilities problems even in double precision floating point arithmetic. This fact calls for one's attention to the need of a stable RLS version, as, for instance, the family of RLS algorithms based on the QR decomposition, which constitutes an elegant and efficient answer to the algorithm stability problem.

2.5 Conclusion

This chapter introduced the most basic concepts associated to adaptive filtering establishing the notation used throughout the book. It was verified how adaptive algorithms are employed to adjust the coefficients of a digital filter to achieve

a desired time-varying performance in several practical situations. Emphasis was given on the description of several adaptation algorithms. In particular, the LMS and the NLMS algorithms were seen as iterative schemes for optimizing the ISE, an instantaneous approximation of the MSE objective function. Data-reuse algorithms introduced the concept of utilizing data from past time samples, resulting in a faster convergence of the adaptive process. Finally, the RLS family of algorithms, based on the WLS function, was seen as the epitome of fast adaptation algorithms, which use all available signal samples to perform the adaptation process. In general, RLS algorithms are used whenever fast convergence is necessary, for input signals with a high eigenvalue spread, and when the increase in the computational load is tolerable. A detailed discussion on the RLS family of algorithms based on the QR decomposition, which also guarantees good numerical properties in finite-precision implementations, constitutes the main goals of this book. Practical examples of adaptive system identification and channel equalization were presented, allowing one to visualize convergence properties, such as misadjustment, speed, and stability, of several distinct algorithms discussed previously.

References

1. B. Widrow and S. D. Stearns, Adaptive Signal Processing. Prentice-Hall, Englewood-Cliffs, NJ, USA (1985)
2. S. Haykin, Adaptive Filter Theory. 2nd edition Prentice-Hall, Englewood Cliffs, NJ, USA (1991)
3. P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 3rd edition Springer, New York, NY, USA (2008)
4. J. A. Apolinário Jr., M. L. R. de Campos, and P. S. R. Diniz, Convergence analysis of the binormalized data-reusing LMS algorithm. IEEE Transactions on Signal Processing, vol. 48, no. 11, pp. 3235–3242 (November 2000)
5. D. T. Slock, On the convergence behavior of the LMS and the normalized LMS algorithms. IEEE Transactions on Signal Processing, vol. 41, no. 9, pp. 2811–2825 (September 1993)
6. P. S. R. Diniz, M. L. R. de Campos, and A. Antoniou, Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor. IEEE Transactions on Signal Processing, vol. 43, no. 3, pp. 617–627 (March 1995)
7. F. F. Yassa, Optimality in the choice of the convergence factor for gradient-based adaptive algorithms. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-35, no. 1, pp. 48–59 (January 1987)
8. A. H. Sayed, Fundamentals of Adaptive Filtering. John Wiley & Sons, Hoboken, NJ, USA (2003)
9. S. Roy and J. J. Shynk, Analysis of the data-reusing LMS algorithm, 32nd Midwest Symposium on Circuits and Systems, MWSCAS'89, Urbana-Champaign, USA, pp. 1127–1130 (1989)
10. W. K. Jenkins, A. W. Hull, J. C. Strait, B. A. Schnaufer, and X. Li, Advanced Concepts in Adaptive Signal Processing. Kluwer Academic Publishers, Norwell, MA, USA (1996)
11. B. A. Schnaufer, *Practical Techniques for Rapid and Reliable Real-Time Adaptive Filtering*. Ph.D. thesis, Adviser: W. K. Jenkins, Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, IL, USA (1995)
12. G. C. Goodwin and S. K. Sin, Adaptive Filtering Prediction and Control. Prentice-Hall, Englewood-Cliffs, NJ, USA (1984)

13. K. Ozeki and T. Umeda, An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties. Transactions IECE Japan, vol. J67-A, no. 5, pp. 126–132 (1984)
14. S. L. Gay and S. Tavathia, The fast affine projection algorithm. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'95, Detroit, USA, pp. 3023–3036 (May 1995)
15. S. G. Sankaran and A. A. Beex, Convergence behavior of affine projection algorithms. IEEE Transactions on Signal Processing, vol. 48, no. 4, pp. 1086–1096 (April 2000)
16. M. L. R. de Campos, P. S. R. Diniz, and J. A. Apolinário Jr., On normalized data-reusing and affine-projections algorithms. IEEE International Conference on Electronics, Circuits, and Systems, ICECS'99, Pafos, Cyprus, pp. 843–846 (1999)
17. M. Montazeri and P. Duhamel, A set of algorithms linking NLMS and block RLS algorithms. IEEE Transactions on Signal Processing, vol. 43, no. 2, pp. 444–453 (February 1995)
18. J. M. Cioffi and T. Kailath, Fast, recursive-least-squares transversal filters for adaptive filtering. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-32, no. 2, pp. 302–337 (April 1984)
19. D. L. Lee, M. Morf, and B. Friedlander, Recursive least squares ladder estimation algorithms. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-29, no. 3, pp. 627–641 (June 1981)
20. J. M. Cioffi, The fast Householder filters RLS adaptive filter. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'90, Albuquerque, USA, pp. 1619–1622 (1990)
21. K. J. R. Liu, S.-F. Hsieh, and K. Yao, Systolic block Householder transformation for RLS algorithm with two-level pipelined implementation. IEEE Transactions on Signal Processing, vol. 40, no. 4, pp. 946–957 (April 1992)
22. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares adaptive filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. SP-41, no. 1, pp. 20–30 (January 1993)
23. J. M. Cioffi, The fast adaptive ROTOR's RLS algorithm. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-38, no. 4, pp. 631–653 (April 1990)
24. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. SP-39, no. 4, pp. 879–891 (April 1991)
25. M. D. Miranda and M. Gerken, A hybrid QR-lattice least squares algorithm using a priori errors. 38th Midwest Symposium on Circuits and Systems, MWSCAS'95, Rio de Janeiro, Brazil, pp. 983–986 (August 1995)
26. A. A. Rontogiannis and S. Theodoridis, New fast inverse QR least squares adaptive algorithms. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'95, Detroit, USA, pp. 1412–1415 (May 1995)
27. J. A. Apolinário Jr. and P. S. R. Diniz, A new fast QR algorithm based on a priori errors. IEEE Signal Processing Letters, vol. 4, no. 11, pp. 307–309 (November 1997)
28. J. A. Apolinário Jr., M. G. Siqueira, and P. S. R. Diniz, Fast QR algorithms based on backward prediction errors: a new implementation and its finite precision performance. Birkhäuser Circuits Systems and Signal Processing, vol. 22, no. 4, pp. 335–349 (July/August 2003)
29. F. Ling, Givens rotation based least squares lattice and related algorithms. IEEE Transactions on Signal Processing, vol. SP-39, no. 7, pp. 1541–1551 (July 1991)
30. I. K. Proudler, J. G. McWhirter, and T. J. Shepard, Computationally efficient QR decomposition approach to least squares adaptive filtering. IEE Proceedings-F, vol. 138, no. 4, pp. 341–353 (August 1991)
31. F. Desbouvries and P. A. Regalia, A minimal, rotation-based FRLS lattice algorithm. IEEE Transactions on Signal Processing, vol. 45, no. 5, pp. 1371–1374 (May 1997)