**Development and Analysis of Fast and Robust Newton-Type Adaptation Algorithms**

by

**Marcello Luiz Rodrigues de Campos**
Elec.Eng., Federal University of Rio de Janeiro, 1990
M.Sc., COPPE/Federal University of Rio de Janeiro, 1991

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

We accept this dissertation as conforming
to the required standard

---

Dr. A. Antoniou, Supervisor (Dept. of Elect. and Comp. Eng.)

---

Dr. P. Agathoklis, Departmental Member (Dept. of Elect. and Comp. Eng.)

---

Dr. R. L. Kirlin, Departmental Member (Dept. of Elect. and Comp. Eng.)

---

Dr. G. McLean, Outside Member (Dept. of Mech. Eng.)

---

Dr. A. Spanias, External Examiner (Arizona State University)

Supervisor: Dr. Andreas Antoniou

# ABSTRACT

Fast and robust Newton-type adaptation algorithms are first developed and are then analyzed.

Chapter 1 is an introduction to the field of FIR digital adaptive filters. It gives an overview of some applications and how adaptive filters are used. Wiener filtering is briefly introduced as a first step towards the understanding of several adaptation algorithms, which are presented in the order of increasing complexity. The first algorithm discussed is the least-mean-squares (LMS) algorithm, followed by an overview of transform-domain algorithms, and the LMS-Newton (LMSN) algorithm. The recursive-least-squares (RLS) algorithm is then presented as a special case of the more general LMSN algorithm, and the finite-precision effects on the performance of the algorithms are examined. The chapter concludes with a QR-decomposition implementation of the LMSN algorithm and a brief discussion on fast algorithms.

The original contributions of the thesis start in chapter 2 where two LMSN algorithms with variable convergence factor are analyzed. Convergence in the mean and variance are investigated, and a comprehensive study of the implications of a fixed-point-arithmetic implementation is carried out. In order to provide a more robust alternative to these algorithms, a QR-LMSN algorithm with variable convergence factor is proposed. The performance of the algorithm is compared to that of the RLS algorithm in different conditions in stationary and nonstationary environments. Simulations using fixed-point arithmetic were performed and the results are compared to those expected from theory.

Chapter 3 covers the quasi-Newton (QN) algorithm in detail. Although developed in entirely different ways, the QN and the LMSN algorithms are very similar and much of what was discussed in chapter 2 with regards to mean-squared error can be applied to the QN algorithm as well. A thorough discussion of the algorithm and a series of lemmas and theorems are provided that demonstrate the stability of the algorithm and the positive definiteness of the estimated input-signal autocorrelation matrix.

The issue of positive definiteness of the estimates of the input-signal autocorrelation matrix used in the QN and RLS algorithms is also discussed. This study qualitatively shows that a greatly improved mechanism of self stabilization is inherent in the QN algorithm relative to the RLS algorithm. Simulations show stable operation of the QN algorithm in fixed and floating-point arithmetic for highly correlated and even nonpersistently exciting signals.

In chapter 4, the issue of fast implementation of Newton-type algorithms is discussed and a fast LMSN algorithm along with its initialization is proposed. Important relations between the fast LMSN and the fast RLS algorithms are first established, and the introduction of a variable convergence factor is carefully examined.

In chapter 5, the conclusions of the thesis are summarized and outstanding issues that need to be addressed in the future are discussed.

Examiners:

Dr. A. Antoniou, Supervisor (Dept. of Elect. and Comp. Eng.)

Dr. P. Agathoklis, Departmental Member (Dept. of Elect. and Comp. Eng.)

Dr. R. L. Kirlin, Departmental Member (Dept. of Elect. and Comp. Eng.)

Dr. G. McLean, Outside Member (Dept. of Mech. Eng.)

Dr. A. Spanias, External Examiner (Arizona State University)

# Table of Contents

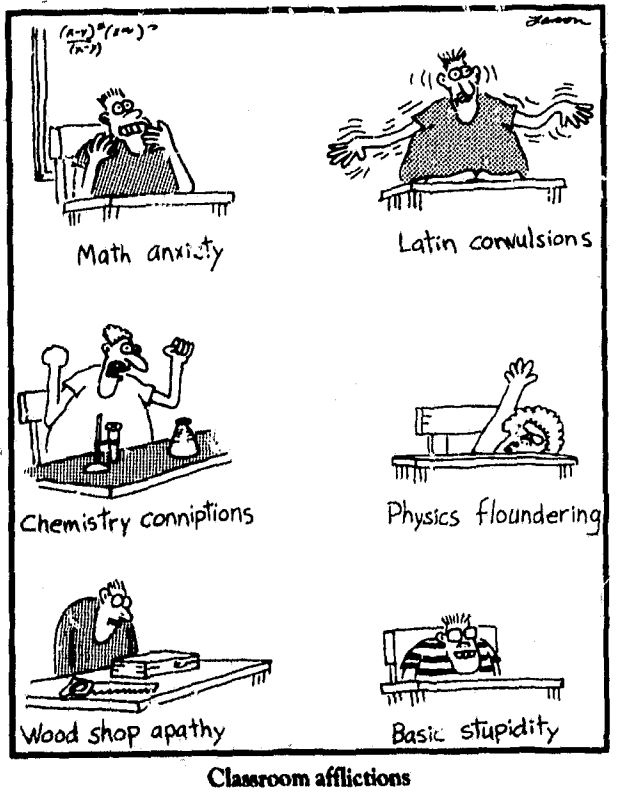
# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| ARMA | autoregressive moving average |
| BFGS | Broyden Fletcher Goldfarb Shanno |
| DFP | Davidon Fletcher Powell |
| DFT | discrete Fourier transform |
| FIR | finite-duration impulse response |
| FTF | fast transversal filter |
| IIR | infinite-duration impulse response |
| ISI | intersymbol interference |
| KLT | Karhunen-Lóeve transform |
| LEMS | loudspeaker-enclosure-microphone system |
| LMS | least mean squares |
| LMSN | LMS-Newton |
| LS | least squares |
| MSE | mean-squared error |
| NLMS | normalized least mean squares |
| QN | quasi-Newton |
| QR-LMSN | QR-decomposition LMS-Newton |
| RLS | recursive least squares |
| TDAF | transform-domain adaptive filter |
| TF | transversal filter |
| VLSI | very large scale integration |

# Acknowledgments




**Classroom afflictions**

Throughout the almost four years that I have been involved with my Ph.D. program I was very fortunate to know many incredible people and to be able to rely on old and new friends. I thank them all for their support. They helped me through my anxieties, convulsions, conniptions, and flounderings. They forgave my apathy and stupidity, and I will be forever grateful. In particular I want to thank my parents Luiz Marcello and Heloisa, my grandmother Elza, my sisters Maria, Teresa, and Ana, and the brothers, nice, and nephews they gave me, Hélio, Antonio Guaracy, José Carlos, Camila, Thiago, and João Marcelo. Their love is much treasured. I want also to thank my friends in the "comunidade," Alexandre, Tatiana, Kurth, Cristina, Sylvio, Cristiane, Ronaldo, Jacqueline, Paulo Ricardo, Mônica, Andréa, and Paulo. I will never forget where home is; it is wherever they are. A special thanks to my parents-

in-law, Sylvio and Maria José, for truly letting me feel like a son, and D. Luiza for her sweetness. I thank my friends Ronald, Rebecca, Liglio, Karla, Bagiuski, Sergio Ricardo, Marcio Guidorizzi, Marcio Magalhães, Claudia, Ronaldo, Juan, and Cristina for their support. In special, thanks to Sergio, my coauthor, my companion through endless winters. I could write another thesis with what I learned from him. A big thanks to Leonardo, he knows the obstacles I encountered and helped me to clear my way with coffee and beer. Thanks to Lisiane, Axel, Vera, Lane, Sloane, Peixoto, Sean, José Luiz, and Adriana for making me feel at home when it seemed so far away. I am also grateful to all the people in the Department of Electrical and Computer Engineering, especially Cathy and Al, for their help, support, invaluable teachings, fruitful discussions, and companionship. I thank Dr. Paulo Diniz, who guided me through my early steps in science and taught me invaluable lessons I will never forget. Last, but not least, I thank my supervisor, Dr. Andreas Antoniou, for his patience, strong guidance, and unconditional support. I am very honored to have been his coauthor.

This work was partially supported by CAPES— Ministry of Education, Brazil.

To *Luciana*, my wife, friend,
and companion. To you, all
my gratitude and love.

# Chapter 1

# Adaptive Filters

## 1.1 Introduction

An adaptive filter is a time-dependent filter that has its coefficients, and in some cases its order, automatically adjusted in real time in order to improve its performance [1]. Though adaptive filtering is a relatively young area in signal processing, recursively adjusting the parameters of a function based on observed data goes back centuries. The idea is to compare the solution obtained from the parameterized function with some reference data and to produce a new set of parameters that is closer in some sense to the optimal solution. Although intuitively sound, the formalization of the method can become very involved. For instance, carefully chosen criteria must guide the actions of the algorithm, and interference by the designer must be kept to a minimum. Furthermore, adaptive filters must be reliable for in some applications failure can be disastrous.

A good example of an adaptation method was proposed by Gauss at the beginning of the 19th century for the determination of orbits of heavenly bodies satisfying as nearly as possible any number of observations [2]. His approach was to correct the available knowledge of the orbit with new data in order to satisfy all the observations in the most accurate manner possible. The more data gathered, the more accurate the parameters describing the orbit should be.

In the area of signal processing, the need for better predictors, equalizers, inter-

ference cancellers, etc., requires systems that can adapt themselves to the characteristics of a particular environment and, therefore, can yield more satisfactory results. Moreover, if the environment changes, the adaptive filter must adjust its parameters automatically to the new situation. In these applications, as in the trajectory predictions by Gauss, the adaptive filter must rely on new observations to achieve the best solution possible.

Although adaptive filters can be analog or digital, finite-duration impulse response (FIR) digital-filter structures have become the most widely used because of their simplicity and reliability. Adaptive infinite-duration impulse response (IIR) digital filters are still being investigated as a possible alternative to FIR adaptive filters in applications where a high filter order is required. The recursive structure of IIR adaptive filters, however, poses extra problems with regards to robustness and stability [3], [4]. For these reasons, the use of IIR adaptive filters is still the subject of considerable research.

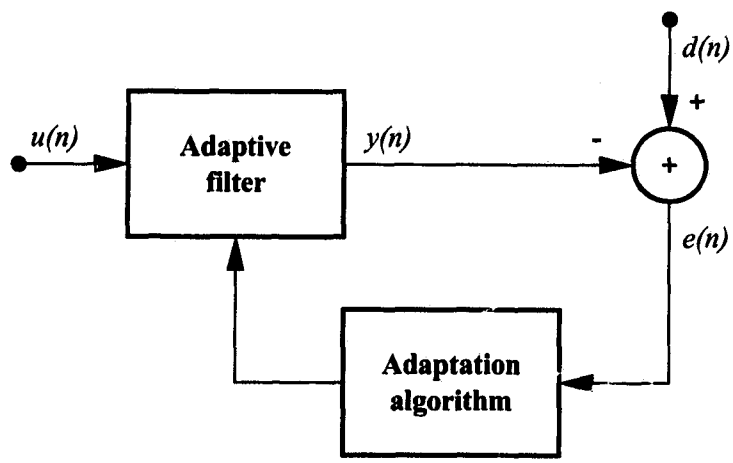



Figure 1.1: General adaptive filter structure.

A typical adaptive digital filter configuration is illustrated in Fig. 1.1. The external data provided to the system at every iteration $n$ are the input signal, $u(n)$, and the reference (or desired) signal, $d(n)$. The adaptation algorithm adjusts the coefficients of the filter such that some norm of the error signal defined as

$$e(n) = d(n) - y(n)$$

is minimized. Signal $y(n)$ is the output of the filter for the given input $u(n)$. The updating is, in general, carried out at every iteration according to the equation

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu(n)\mathbf{c}(n) \tag{1.1}$$

where $\mathbf{w}(n)$ is the coefficient vector of the filter, $\mu(n)$ is the convergence factor or step size, and $\mathbf{c}(n)$ is the correction to be applied to the coefficients. Sometimes the amount of computation is excessive and the coefficients are updated only every $N$ iterations; this is referred to as block adaptation.

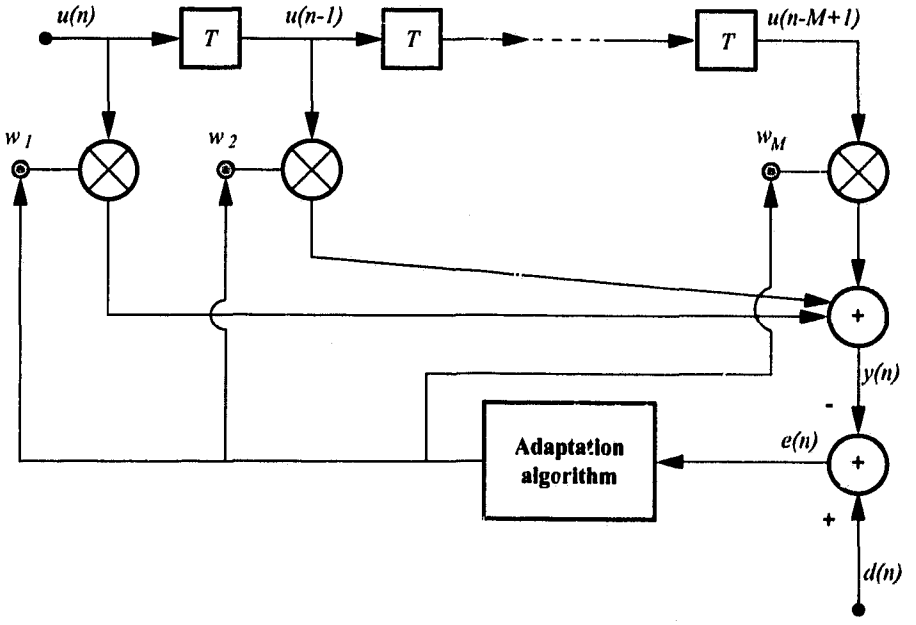



Figure 1.2: Transversal filter structure.

For readers familiar with optimization procedures, the format of (1.1) should come as no surprise. The differences among the several available adaptive-filtering schemes are in the way $\mathbf{w}$ is constructed as well as how $\mu$ and $\mathbf{c}$ are calculated. The most common structure, and the most simple, is the transversal filter (TF), sometimes called the tapped delay line or linear combiner, depicted in Fig. 1.2. In this structure, the output of the filter is the result of the inner product operation between the coefficient vector

$$\mathbf{w} = [w_1\ w_2\ \cdots\ w_M]^T$$

and an input-signal vector containing delayed versions of the input signal, i.e.,

$$\mathbf{u}(n) = [u(n)\ u(n-1)\ \cdots\ u(n-M+1)]^T$$

where $M$ is the number of coefficients of the filter. For the TF structure, the derivation of algorithms and their analysis are relatively simple as compared to those of more complex structures. These reasons are strong enough to make the TF structure popular and a perfect example for introducing the subject.

Once the structure is chosen, the next step is to choose among the available adaptation algorithms the one that best suits the application at hand. Before we go on with the study of the different algorithms and their properties, we briefly discuss some applications where adaptive filters have proved particularly useful. This will help the reader to better understand how signals $u(n)$ and $d(n)$ are obtained and will also emphasize the importance, flexibility, and usefulness of adaptive filters.

Echo cancellation — either in loudspeaker-enclosure-microphone systems (LEMS) or in satellite-communication systems — is one application where the nonstationarity of the signals involved and changes in the medium make adaptive filters especially useful.

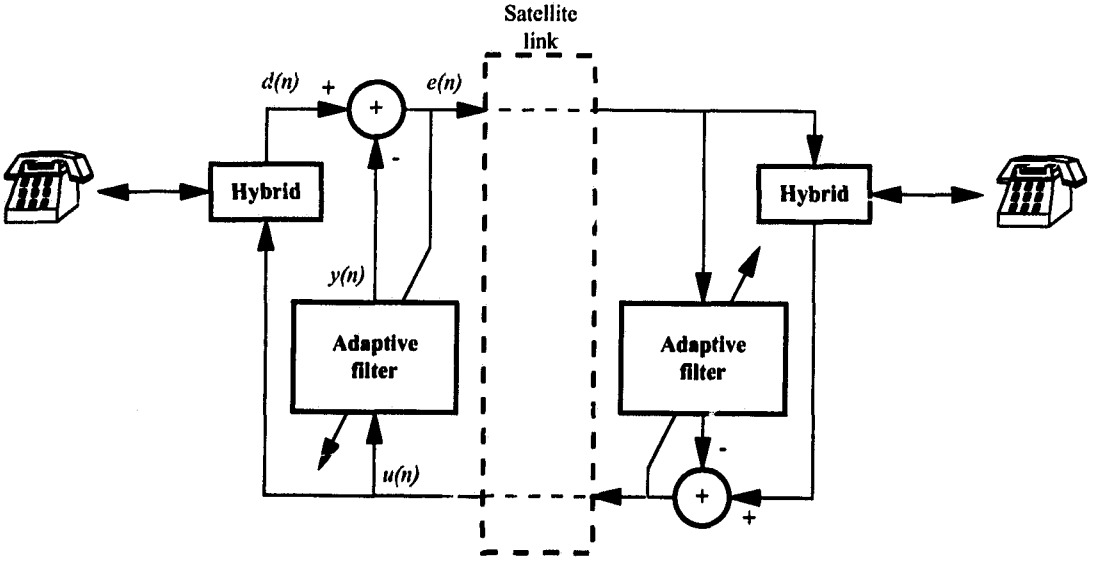



Figure 1.3: Adaptive echo canceller.

In the telephone network, although local calls are established through two-wire

circuits which serve for transmission in both directions, long-distance calls are made through four-wire circuits where separate paths are provided for each direction of transmission, as depicted in Fig. 1.3. The connection between the two-wire and the four-wire circuits is made by a hybrid circuit. Impedance mismatch at the hybrid circuit results in coupling of the incoming and outgoing branches of the four-wire circuit and, consequently, part of the signal is reflected and returned to the original talker. Traditional methods for echo control can degrade significantly the quality of the conversation, especially when both parties are talking simultaneously. High-quality communication systems call for alternative solutions which suppress the echo without compromising the quality of the transmission. The adaptive filter employed as echo canceller is supposed to mimic the transfer characteristics of the echo path and produce a copy of the echo to be subtracted from the return signal. In the case of acoustic cancellation in an LEMS, the adaptive filter compensates for any echo generated when the microphone captures the signal generated at the loudspeaker [5].

Another application where adaptive filters are very useful is channel equalization. Due to a nonideal frequency response in a telephone or radio channel, e.g., if the amplitude response is not constant and/or the phase response is not linear with respect to the channel bandwidth, time dispersion can arise which causes symbols to overlap thereby resulting in intersymbol interference (ISI). The use of an adaptive filter for equalization is illustrated in Fig. 1.4 where the equalizer must correct ISI caused by channels with different and often time-dependent characteristics. In the past, equalization has been done largely with fixed linear equalizers which work well when tuned to a specific channel whose characteristics do not change with time. The underlying adaptation process essentially adjusts the transfer function of the equalizer such that the equalized channel has a constant amplitude response and linear phase response with respect to the channel bandwidth. In this application the reference signal is, in general, a training sequence known at the receiver. After the training period, decision-directed techniques are used to continuously provide a reference signal and, therefore, to allow for tracking slow variations of the channel [6]. Better results in mitigating the effects of time dispersion can be achieved with more sophisticated

adaptive equalizers than with the simplified model shown in Fig. 1.4 [7]. However, the basics of the adaptation of the coefficients are the same.

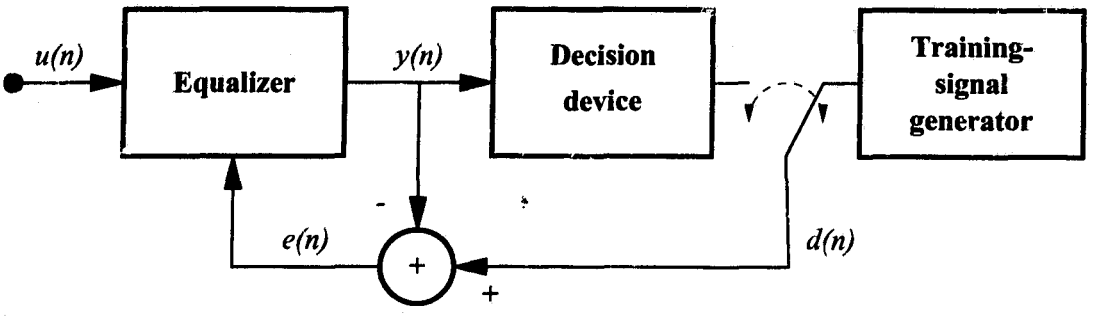



Figure 1.4: Simplified adaptive channel equalizer.

Adaptive filters can be employed in numerous other applications, such as system identification, automatic regulation, linear prediction, etc. The need for more efficient and robust algorithms is, therefore, constantly increasing. Next, we shall present an overview of some of the known algorithms and their basic properties.

## 1.2 Wiener Filtering

Some knowledge of linear least-squares estimation, for example, the Wiener-Hopf equation [8], is essential for the development and understanding of adaptation algorithms. Therefore, a brief introduction of Wiener filtering is appropriate at this point.[1]

Let the objective function $\mathcal{J}(n)$ denote the mean-squared error (MSE) between the reference signal and the output of a transversal filter with coefficient vector $\mathbf{w}$, i.e.,

$$\mathcal{J}(n) = E\left\{\left[d(n) - \mathbf{u}^T(n)\mathbf{w}\right]^2\right\} \tag{1.2}$$

The goal of optimal filtering in this case is to find the optimal coefficient vector, $\mathbf{w} = \mathbf{w}_o$, that minimizes $\mathcal{J}(n)$, i.e., $\mathcal{J}(n) = \mathcal{J}_{min}$. By differentiating (1.2) with

[1]For a more complete discussion on linear estimation, the reader is referred to [9].

respect to the elements of $\mathbf{w}$ and equating the first derivative to zero, we obtain

$$E\left\{\mathbf{u}(n)\left[d(n)-\mathbf{u}^T(n)\mathbf{w}_o\right]\right\}=0 \tag{1.3}$$

Since (1.2) is quadratic in $\mathbf{w}$, a unique solution is guaranteed. Furthermore, it is easy to verify that it corresponds to a minimum as long as $E\left[\mathbf{u}(n)\mathbf{u}^T(n)\right]$ is positive definite. Equation (1.3) is often referred to as the orthogonality principle [8], which means that for the optimal set of coefficients, the estimation error is orthogonal to the previous $M$ input-signal samples. The solution, $\mathbf{w}_o$, satisfies the Wiener-Hopf equation which can be expressed in this case in terms of the input-signal autocorrelation matrix

$$\mathbf{R}=E\left[\mathbf{u}(n)\mathbf{u}^T(n)\right]$$

and the crosscorrelation between the input signal and the reference signal

$$\mathbf{p}=E\left[d(n)\mathbf{u}(n)\right]$$

as

$$\mathbf{R}\mathbf{w}_o=\mathbf{p}$$

The minimum mean-squared error, obtained when $\mathbf{w}=\mathbf{w}_o$, is given by

$$\mathcal{J}_{min}=E\left[d^2(n)\right]-\mathbf{w}_o^T\mathbf{R}\mathbf{w}_o \tag{1.4}$$

Since $\mathbf{R}$ and $\mathbf{p}$ are often unknown, the task of the adaptation algorithm is to recursively produce estimates of $\mathbf{w}_o$, i.e., $\mathbf{w}(n)$. For signals that are stationary, at least in the wide sense [10], convergence of $\mathbf{w}(n)$ to $\mathbf{w}_o$ in the mean can be expected in most cases. The covariance, however, depends on several different factors and is, in general, one of the chief figures for comparing the performance of different algorithms. A study of the first two moments of $\mathbf{w}(n)$ are key to assure convergence and to establish a performance criterion. In general, performance is measured in terms of the excess mean-squared error $\mathcal{J}_{ex}(n)$, which is defined as

$$\mathcal{J}_{ex}(n)=\mathcal{J}(n)-\mathcal{J}_{min} \tag{1.5}$$

This figure is closely related to the covariance of $\mathbf{w}(n)$, since from (1.4) and (1.5) it can be easily shown that

$$\mathcal{J}_{ex}(n) = E\left\{[\mathbf{w}(n) - \mathbf{w}_o]^T \mathbf{R}\,[\mathbf{w}(n) - \mathbf{w}_o]\right\}$$

For convenience, sometimes performance is measured in terms of a dimensionless parameter called misadjustment defined as the excess MSE divided by the minimum MSE, i.e.,

$$\mathcal{M}(n) = \frac{\mathcal{J}_{ex}(n)}{\mathcal{J}_{min}}$$

Function $\mathcal{J}(n)$ can be represented by a performance surface which describes the MSE as a function of the $M$-dimensional coefficient vector. The shape of the performance surface in the $(M+1)$-dimensional hyperspace depends on the input signal; the more spread in the eigenvalues of $\mathbf{R}$, the more asymmetric is the performance surface. Figure 1.5 shows an example of a performance surface for $M = 2$.

When nonstationary signals are present, either $\mathbf{R}$ or $\mathbf{p}$ or both are time dependent and the algorithm is required to track the solution. In practice, nonstationarities are identified in various forms (e.g., fading channels, changes in the position of furniture and people in LEMS, etc.). Modelling the optimal coefficients as an autoregressive-moving average (ARMA) process [8], i.e.,

$$\mathbf{w}_o(n) = \sum_{i=1}^{M} a_i \mathbf{w}_o(n-i) + \sum_{i=0}^{M} b_i \boldsymbol{\nu}(n-i)$$

where $M$ is the order of the process, is a valuable tool to verify the performance of an algorithm under these conditions. Reconvergence testing can be performed by introducing abrupt changes in the coefficients. Proper operation can be observed only when the time constant of the algorithm is smaller than that of the nonstationarity.

## 1.3 LMS Algorithm

An adaptation algorithm known as the least-mean-squares (LMS) algorithm has become very popular since it was proposed by Widrow and Hoff [11], [12], and its simplicity and reliability have a great deal to do with its popularity. In this algorithm,

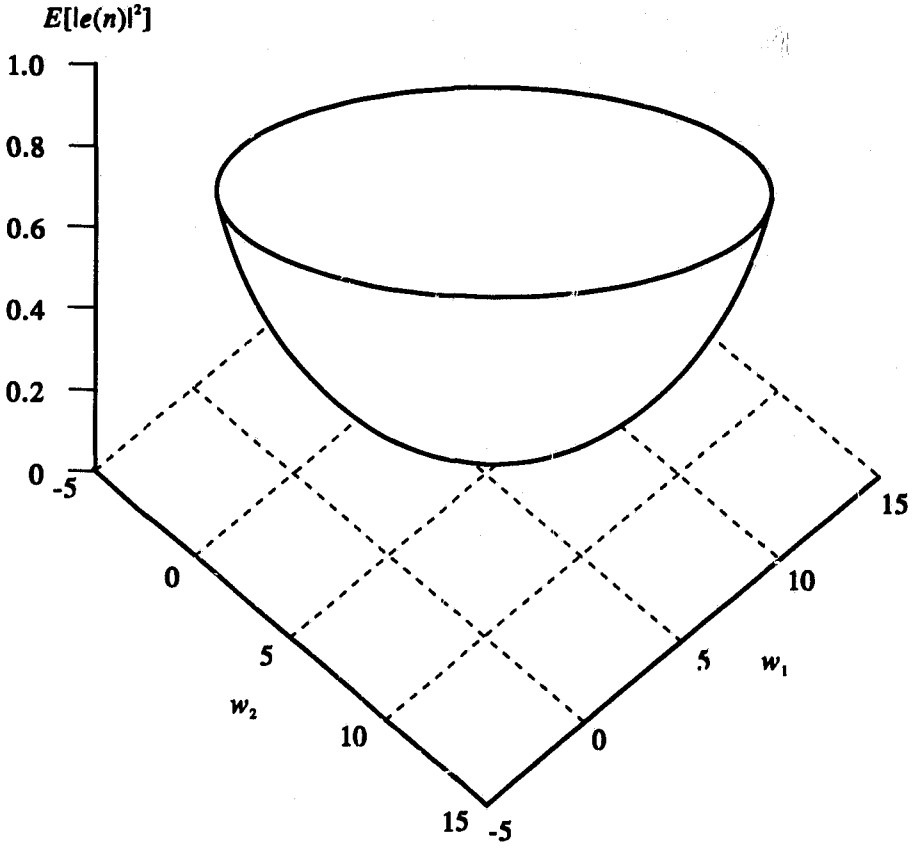



Figure 1.5: Performance surface for $M = 2$.

the coefficients are updated as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + 2\mu e(n)\mathbf{u}(n) \tag{1.6}$$

where $\mu$ is the convergence factor and $e(n)$ is referred to as the *a priori* output error and is defined as

$$e(n) = d(n) - \mathbf{u}^T(n)\mathbf{w}(n-1) \tag{1.7}$$

If the computational complexity of the algorithm can be measured by the number of multiplications required per iteration, the LMS algorithm has a complexity which is proportional to the number of coefficients of the filter $M$, or $\mathcal{O}(M)$ in mathematical notation. The algorithm can be further simplified by replacing the output error by

its sign. The resulting sign-error updating can be described as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + 2\mu\,\text{sign}[e(n)]\mathbf{u}(n) \tag{1.8}$$

where

$$\text{sign}[e(n)] = \begin{cases} -1 & \text{for } e(n) < 0 \\ 0 & \text{for } e(n) = 0 \\ +1 & \text{for } e(n) > 0 \end{cases}$$

In (1.6) and (1.8) the coefficients are corrected in the direction of the negative of the gradient of the instantaneous squared error [13], $e^2(n)$, and the magnitude of $e(n)$, $|e(n)|$, respectively. The LMS algorithm and its variants are often referred to as *gradient-based* algorithms.

The LMS algorithm is not consistent from the statistical point of view, for measurement noise and nonstationarity imply nonzero covariance of the coefficient vector [14]. For instance, $\mu$ is proportional to the portion of the misadjustment due to measurement noise and it is inversely proportional to that due to lag in tracking nonstationarities. In stationary environments, zero misadjustment can be attained if the convergence factor is gradually reduced to zero after convergence. In most of the cases, though, this procedure should be avoided, for the algorithm will have no power to track most of the changes in the environment. The convergence factor is intrinsically related to stability and a sufficient condition for convergence of the coefficient vector in the mean, assuming stationary signals, is

$$0 < \mu < \frac{1}{\text{tr}(\mathbf{R})} \tag{1.9}$$

where $\text{tr}(\cdot)$ denotes trace. This condition is both necessary and sufficient for the convergence of the variance [14].

The optimal choice for the convergence factor depends on prior knowledge of the statistics of the signals involved, which can be difficult or even impossible to obtain. For this reason, many methods to estimate $\mu$ online have been proposed in the past (see, for example, [15], [16], and [17]). Although most of these methods introduce

other parameters to which the algorithm is less sensitive, there is strong indication that a convergence factor that is inversely proportional to the instantaneous power of the input-signal vector would yield a potentially faster algorithm when limited knowledge is available [18] [19]. Such an algorithm is called normalized LMS (NLMS) algorithm and the coefficients are updated as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + 2\frac{\bar{\mu}}{\|\mathbf{u}(n)\|_2^2}e(n)\mathbf{u}(n) \tag{1.10}$$

Note that a convergence factor is kept in (1.10) but as a rule of thumb $\bar{\mu} = 0.5$ can be used with good results in most cases.

The major problem affecting both the LMS and NLMS algorithms is their slow convergence when the input signal is highly correlated. For the LMS algorithm, for instance, the coefficient-error vector converges in the mean to zero according to the relation [14]

$$E[\mathbf{v}(n)] = (\mathbf{I} - 2\mu\mathbf{R})E[\mathbf{v}(n-1)] \tag{1.11}$$

where $\mathbf{I}$ denotes the identity matrix and $\mathbf{v}(n)$ is defined as

$$\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_o$$

Applying the spectral decomposition to $\mathbf{R}$, i.e.,

$$\mathbf{R} = \mathbf{Q}\Lambda\mathbf{Q}^T$$

with $\mathbf{Q}$ formed by the orthonormal eigenvectors of $\mathbf{R}$ and $\Lambda$ is a diagonal matrix formed by its eigenvalues, (1.11) can be rewritten in terms of the modified coordinates $\bar{\mathbf{v}}$ as

$$E[\bar{\mathbf{v}}(n)] = (\mathbf{I} - 2\mu\Lambda)E[\bar{\mathbf{v}}(n-1)] \tag{1.12}$$

where $\bar{\mathbf{v}}(n) = \mathbf{Q}^T\mathbf{v}(n)$. The relaxation process in (1.12) is a geometric progression with a ratio of $(1 - 2\mu\lambda_i)$ for each mode. An exponential envelope the type $e^{-n/\tau_i}$ can then be fitted to each mode with a time constant approximately equal to [13]

$$\tau_i \approx \frac{1}{2\mu\lambda_i}$$

where $\mu\lambda_i \ll 1$. Therefore, the overall speed of convergence is dominated by the smallest eigenvalue, while from (1.9) the upper limit for the convergence factor is dominated by the largest eigenvalue. In the case where the eigenvalues of $\mathbf{R}$ are widely spread, unacceptably slow convergence can result. Several attempts to increase the speed of convergence of the LMS algorithm have been reported (see, e.g., [15] [19] [20] [21]), invariably at the cost of increasing its computational complexity. Since simple real-time methods for optimizing the convergence factor still suffer from slow convergence when $\mathbf{R}$ is ill-conditioned, more elaborate estimation schemes have to be employed in such situations. Transform-domain adaptive filters constitute the next step towards the study of algorithms which are less sensitive to input-signal correlation.

## 1.4 Transform-Domain Adaptive Filters

Transform-domain FIR adaptive filters (TDAF) are basically gradient-based algorithms operating on a transformed input signal, as depicted in Fig. 1.6 [20].

Equation (1.12) suggests that maximum convergence speed can be achieved if the input signal is transformed to have a diagonal autocorrelation matrix, and a different convergence factor is used for each mode. In this way the speed of convergence of each mode can be independently maximized. In other words, the transformation combined with the power normalization introduced by different convergence factors provide the means for changing the aspect of the performance surface associated to the transformed input signal [22]. Usually it can be said that the more symmetric the performance surface, the faster is the convergence of a gradient-based algorithm. It can also be shown that the use of an orthogonal transformation followed by normalization does not degrade the performance of the LMS algorithm [20].

A transformation that performs a perfect decoupling of the modes of the process described by (1.11) is the Karhunen-Lóeve transformation (KLT) [8]. Unfortunately, its implementation requires knowledge of the eigenvalues and eigenvectors of $\mathbf{R}$ which are not available in most applications. A more feasible approach is to use other

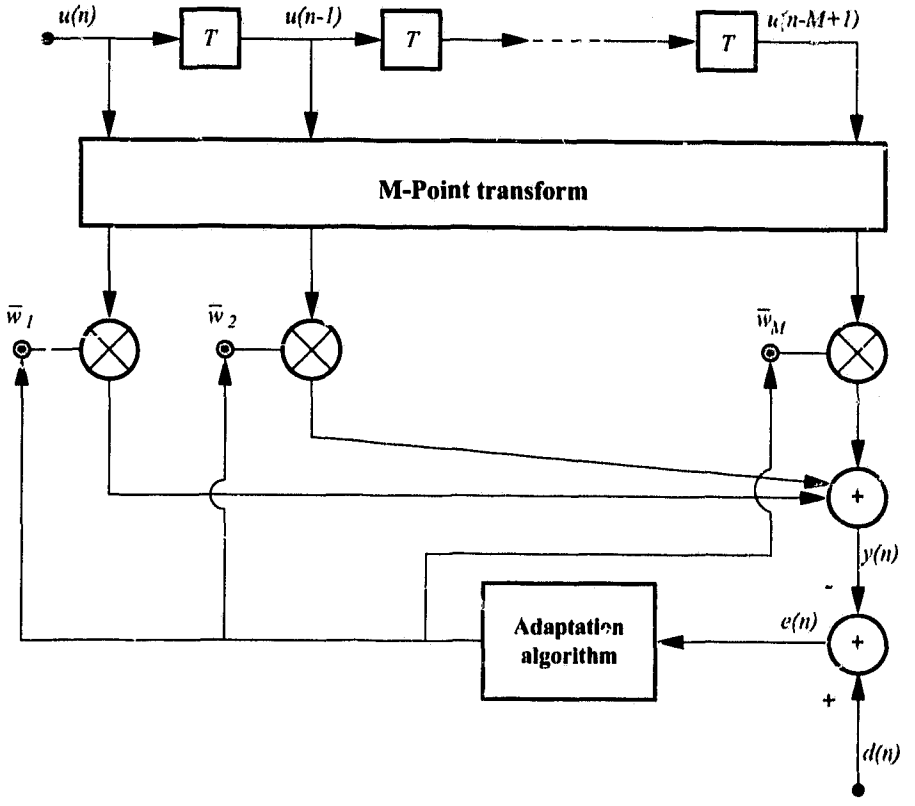



Figure 1.6: Transform-domain adaptive filter.

orthogonal transformations $\mathbf{T}$ with suboptimal results. In general, transformations that lead to a fast implementation are preferred due to real-time constraints. For the transform-domain LMS algorithm, the coefficient vector is updated according to the equation

$$\bar{\mathbf{w}}(n) = \bar{\mathbf{w}}(n-1) + 2\mu e(n)\Omega^{-1}\mathbf{T}^T\mathbf{u}(n) \tag{1.13}$$

where $\bar{\mathbf{w}}(n) = \mathbf{T}^T\mathbf{w}(n)$ and $\Omega^{-1}$ is a diagonal matrix with an individual power normalization for each coefficient. Although the coefficient vector $\bar{\mathbf{w}}(n)$ converges in the mean to a transformed version of $\mathbf{w}_o$, it is worthwhile to note that the output of the TF with coefficient vector $\bar{\mathbf{w}}(n)$ and input-signal vector $\mathbf{T}^T\mathbf{u}(n)$ is not affected by the orthogonal transformation $\mathbf{T}$.

The choice of a good transformation for a particular application is by no means an easy task. Computer simulations comparing different transforms show that knowledge

of the input-signal characteristics is key to assuring a substantial improvement [21]. For instance, as $\mathbf{R}$ is approximately circulant for large values of $M$ the discrete Fourier transform (DFT) should be a good choice in this case, for the eigenvectors of an $M \times M$ circulant matrix can be obtained from an $M$-point DFT. For small values of $M$, however, the DFT might be a bad choice as compared to other transforms. Furthermore, its complex coefficients impose restrictions to its applicability. A potentially more efficient solution is to provide means to the adaptation algorithm to estimate the KLT in real time. An algorithm that employs such a technique is called LMS-Newton algorithm [20].

## 1.5 LMS-Newton Algorithm

The LMS-Newton (LMSN) algorithm inherits its name from the analogy to Newton's method for finding the zeros of a function. The method has also been successfully used for many decades as a powerful optimization technique. The improvement in performance when compared to that achieved with gradient-based techniques is due to a more accurate description of the objective function based on second-order information. For quadratic convex functions, the method converges to the optimal solution in a single step [12]. As (1.2) is quadratic in $\mathbf{w}$, the Newton method would, in theory, give the solution to the Wiener-Hopf equation instantaneously. In practice, however, the objective function is not known and an iterative solution must be sought.

The LMSN algorithm updates the filter coefficients according to the equation

$$\mathbf{w}(n) = \mathbf{w}(n-1) + 2\mu e(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n) \tag{1.14}$$

where $\hat{\mathbf{R}}(n)$ is a positive definite estimate of the input-signal autocorrelation matrix. This estimate must also allow an efficient inversion, for it must be performed in real time. Good results can be achieved with a stochastic approximation method like the one proposed by Robbins and Monro in [23] in which

$$\hat{\mathbf{R}}(n) = \hat{\mathbf{R}}(n-1) + \alpha(n)\left[\mathbf{u}(n)\mathbf{u}^T(n) - \hat{\mathbf{R}}(n-1)\right] \tag{1.15}$$

where $\alpha(n)$ is a sequence of positive scalars. For practical purposes, some of the conditions for consistency are, in general, relaxed so that variance is traded for the ability to track nonstationarities in the input signal. The most common scheme uses a small positive constant $\alpha(n) = \alpha$, for which (1.15) can be viewed as an exponentially weighted average of the outer product of vectors $\mathbf{u}(n)$ and $\mathbf{u}^T(n)$, i.e.,

$$\hat{\mathbf{R}}(n) = \alpha \sum_{i=1}^{n} (1-\alpha)^{n-i} \mathbf{u}(i)\mathbf{u}^T(i) + (1-\alpha)^n \hat{\mathbf{R}}(0) \tag{1.16}$$

In order to facilitate the inversion of $\hat{\mathbf{R}}(n)$ when $n < M$, a positive definite $\hat{\mathbf{R}}(0)$ must be provided since it is assumed that $u(n) = 0$ for $n < 0$. The influence of $\hat{\mathbf{R}}(0)$ decays exponentially for large values of $n$ as can be seen from (1.16). This scheme combines a very good approximation for sufficiently small values of $\alpha$ with the advantage that an inverse can be trivially obtained. The matrix inversion lemma [8] applied to (1.15) yields

$$\hat{\mathbf{R}}^{-1}(n) = \frac{1}{1-\alpha} \left\{ \hat{\mathbf{R}}^{-1}(n-1) - \frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\frac{1-\alpha}{\alpha} + \mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)} \right\} \tag{1.17}$$

which is recursive in time and has computational complexity $\mathcal{O}(M^2)$. This is considerably better than the complexities of more general inversion techniques. Matrix $\hat{\mathbf{R}}(n)$ is positive definite provided that the input signal is at least persistently exciting of order $M$, i.e., its spectrum is nonzero at $M$ or more points in the frequency $-\omega_s/2$ to $\omega_s/2$, where $\omega_s$ is the sampling rate [24].

A slightly more efficient implementation of the LMSN algorithm can be obtained if the relation [25]

$$\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n) = \frac{1}{\alpha} \cdot \frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}{\frac{1-\alpha}{\alpha} + \mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}$$

is observed. The algorithm can be rewritten as

$$\mathbf{t}(n) = \hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$$

$$z(n) = \frac{1-\alpha}{\alpha} + \mathbf{u}^T(n)\mathbf{t}(n)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \frac{2\mu}{\alpha} \cdot \frac{e(n)\mathbf{t}(n)}{z(n)}$$

$$\hat{\mathbf{R}}^{-1}(n) = \frac{1}{1-\alpha}\left\{\hat{\mathbf{R}}^{-1}(n-1) - \frac{\mathbf{t}(n)\mathbf{t}^T(n)}{z(n)}\right\}$$

where $\hat{\mathbf{R}}^{-1}(n)$, being symmetric, does not need to be fully calculated.

It can be easily verified that the LMSN algorithm is a transform-domain LMS algorithm which uses an estimate of the KLT [26]. Let $\mathbf{T} = \hat{\mathbf{Q}}(n)$ be the matrix of the eigenvectors of $\hat{\mathbf{R}}(n)$ and $\Omega = \hat{\Lambda}(n)$ be the diagonal matrix with the eigenvalues of $\hat{\mathbf{R}}(n)$. From (1.13) we have

$$\begin{aligned}\mathbf{T}\bar{\mathbf{w}}(n) = \mathbf{w}(n) &= \mathbf{w}(n-1) + 2\mu e(n)\mathbf{T}\Omega^{-1}\mathbf{T}^T\mathbf{u}(n) \\ &= \mathbf{w}(n-1) + 2\mu e(n)\hat{\mathbf{Q}}(n)\hat{\Lambda}^{-1}(n)\hat{\mathbf{Q}}^T(n)\mathbf{u}(n) \\ &= \mathbf{w}(n-1) + 2\mu e(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n)\end{aligned}$$

Although it can be argued that the gradient direction is noisy, for it is based on the instantaneous squared error rather than on the MSE, the correction applied with a good estimate of $\mathbf{R}^{-1}$ greatly improves the convergence speed when the performance surface is asymmetric. Furthermore, uncorrelated input signals yield symmetric performance surfaces and $\hat{\mathbf{R}}^{-1}(n)$, being diagonal, should not modify the gradient direction.

Parameter $\alpha$ is solely responsible for the memory in estimating $\mathbf{R}$, i.e., it controls how previous input vectors are weighted to form $\hat{\mathbf{R}}(n)$. Its influence on the coefficient adaptation is, therefore, indirect. Control over misadjustment and speed of convergence can be more effectively achieved through $\mu$. It is interesting to note that for the specific case where $2\mu = \alpha$, the LMSN algorithm is equivalent to the recursive least-squares (RLS) algorithm, which will be discussed in the next section.

## 1.6 Recursive Least-Squares Algorithm

The least-squares (LS) method was probably first used by Gauss in the late 18th century [27] and since then it has been applied in a vast number of areas. In every case, the motivation is the same: to estimate the set of parameters that best fits a model for an observed phenomenon. It comes as no surprise that the method should be tried in adaptive filtering as well for, except from the real-time constraints, the problem is basically parameter fitting. What is really a surprise is that the method suits the application so well and that it can be exactly and efficiently implemented in real time. In addition, schemes for dealing with signals with time-dependent statistics can be easily incorporated through the use of different weights, as was done by Gauss in his work [2].

Let $\epsilon(i)$ be defined as the *a posteriori* output error calculated when the output signal of a transversal filter with coefficient vector $\mathbf{w}(n)$ and input-signal vector $\mathbf{u}(i)$ is compared with a reference signal $d(i)$, $i \leq n$, i.e.,

$$\epsilon(i) = d(i) - \mathbf{w}^T(n)\mathbf{u}(i) \tag{1.18}$$

A weighted sum of the a posteriori errors for $i = 1, \ldots, n$ can be constructed as

$$\mathcal{E}(n) = \sum_{i=1}^{n} \lambda^{n-i}\epsilon^2(i) \tag{1.19}$$

which represents the objective function to be minimized. A forgetting factor $\lambda \in (0, 1]$ is introduced in order to exponentially reduce the effect of older samples in the function. Usually a constant $\lambda$ close to unity gives excellent results, although time-dependent values can be used. If we differentiate $\mathcal{E}(n)$ with respect to $\mathbf{w}(n)$ and equate the result to zero, the solution

$$\hat{\mathbf{R}}(n)\mathbf{w}(n) = \hat{\mathbf{p}}(n) \tag{1.20}$$

is obtained, where

$$\hat{\mathbf{R}}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^T(i) \tag{1.21}$$

$$\hat{\mathbf{p}}(n) = \sum_{i=1}^{n} \lambda^{n-i}d(i)\mathbf{u}(i) \tag{1.22}$$

are the sample averages of the autocorrelation matrix and the crosscorrelation vector between the input signal and the reference signal, respectively. Equation (1.20) is sometimes referred to as the deterministic counterpart of the Wiener-Hopf equation [8].

As $\lambda$ controls the amount of memory of the algorithm, $\lambda = 1$ corresponds to the case where all the samples are equally weighted. Although in this case of "infinite memory" the algorithm cannot track variations in the environment, it can be shown that the excess MSE reduces to zero asymptotically as $n$ goes to infinity.

A recursive solution of (1.20) which is based on previously obtained values of $\mathbf{w}(n)$ and $\hat{\mathbf{R}}(n)$, can be derived by simple algebraic manipulation of (1.21) and (1.22) [8]. From (1.21) it can be verified that

$$\hat{\mathbf{R}}(n) = \lambda\hat{\mathbf{R}}(n-1) + \mathbf{u}(n)\mathbf{u}^T(n) \tag{1.23}$$

Similarly, from (1.22)

$$\hat{\mathbf{p}}(n) = \lambda\hat{\mathbf{p}}(n-1) + d(n)\mathbf{u}(n) \tag{1.24}$$

Combining (1.20), (1.23), and (1.24), we have

$$\begin{aligned}
\hat{\mathbf{R}}(n)\mathbf{w}(n) &= \lambda\hat{\mathbf{p}}(n-1) + d(n)\mathbf{u}(n) \\
&= \lambda\hat{\mathbf{R}}(n-1)\mathbf{w}(n-1) + d(n)\mathbf{u}(n) \\
&= \left[\hat{\mathbf{R}}(n) - \mathbf{u}(n)\mathbf{u}^T(n)\right]\mathbf{w}(n-1) + d(n)\mathbf{u}(n) \\
&= \hat{\mathbf{R}}(n)\mathbf{w}(n-1) + \mathbf{u}(n)\left[d(n) - \mathbf{u}^T(n)\mathbf{w}(n-1)\right] \\
&= \hat{\mathbf{R}}(n)\mathbf{w}(n-1) + e(n)\mathbf{u}(n)
\end{aligned} \tag{1.25}$$

where the a priori output error, $e(n)$, is defined as in (1.7). Assuming $\hat{\mathbf{R}}(n)$ is invertible, then from (1.25) we have

$$\mathbf{w}(n) = \mathbf{w}(n-1) + e(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n)$$

The matrix inversion lemma [8] can now be applied to $\hat{\mathbf{R}}(n)$ to yield

$$\hat{\mathbf{R}}^{-1}(n) = \frac{1}{\lambda}\left\{\hat{\mathbf{R}}^{-1}(n-1) - \frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\lambda + \mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}\right\} \tag{1.26}$$

where $\hat{\mathbf{R}}^{-1}(0)$ must be a positive definite matrix. As for the LMSN algorithm, for $\lambda < 1$ the introduction of $\hat{\mathbf{R}}^{-1}(0)$ has a decreasing effect on (1.19), which becomes

$$\mathcal{E}(n) = \sum_{i=1}^{n} \lambda^{n-i} \epsilon^2(i) + \lambda^n \mathbf{w}^T(n) \hat{\mathbf{R}}(0) \mathbf{w}(n) \tag{1.27}$$

The advantages of a recursive algorithm as described by the above equations are well worth the negligible effects introduced by $\hat{\mathbf{R}}^{-1}(0)$.

Using the recursive relation

$$\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n) = \frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}{\lambda + \mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}$$

the RLS algorithm can be implemented as

$$\mathbf{t}(n) = \hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$$

$$z(n) = \lambda + \mathbf{u}^T(n)\mathbf{t}(n)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \frac{e(n)\mathbf{t}(n)}{z(n)}$$

$$\hat{\mathbf{R}}^{-1}(n) = \frac{1}{\lambda}\left\{\hat{\mathbf{R}}^{-1}(n-1) - \frac{\mathbf{t}(n)\mathbf{t}^T(n)}{z(n)}\right\}$$

The influence of the forgetting factor $\lambda$ on the misadjustment of the RLS algorithm is similar to that of $\mu$ and $\alpha$ for the LMSN algorithm. As $\lambda \to 1$ the algorithm is given more memory, which translates in a better performance in stationary environments. On the other hand, better tracking of nonstationarities is observed for smaller values of $\lambda$. Similarities between the LMSN and RLS algorithms go even further, since it can be easily shown that for $2\mu = \alpha = 1 - \lambda$, the LMSN algorithm minimizes a scaled version of (1.27). Under these conditions, both algorithms should have similar performance. Another interesting parallel can be established between the RLS algorithm and the Kalman filter. The RLS algorithm can be viewed as a special case of the Kalman filter, if the correspondences given in Table 1.I are assumed [8]. As compared to the LMS algorithm, the RLS algorithm presents a much better convergence rate when the input signal is correlated. Therefore, improvement in convergence speed is seldom necessary for the RLS algorithm. It is much more important to reduce the number of operations per iteration and assure robustness.

TABLE 1.I

CORRESPONDENCE BETWEEN THE RLS ALGORITHM AND THE KALMAN FILTER

| Kalman Filter | RLS Algorithm |
|---|---|
| Observation | $d(n)$ |
| State Transition Matrix | $\mathbf{I}$ |
| Measurement Matrix | $\mathbf{u}^T(n)$ |
| Estimated State Vector | $\mathbf{w}(n)$ |
| Innovation | $e(n)$ |
| Filtered State-Error Correlation Matrix | $\lambda\hat{\mathbf{R}}^{-1}(n)$ |
| Predicted State-Error Correlation Matrix | $\hat{\mathbf{R}}^{-1}(n)$ |
| Kalman Gain | $\mathbf{t}(n)$ |
| Estimated Process-Noise Correlation Matrix | $(\lambda^{-1}-1)\lambda\hat{\mathbf{R}}^{-1}(n)$ |
| Estimated Measurement-Noise Correlation Matrix | $\lambda$ |

### 1.6.1 Finite-Precision Effects

One aspect that has been purposely ignored in the previous sections is the effect of finite precision in the implementation of the various algorithms on digital signal processors.

For the LMS algorithm, quantization effects compromise the algorithm performance through a higher misadjustment. As machine precision decreases, the influence of the excess MSE due to quantization becomes more noticeable as a third source of error in the solution. In addition, very small values of $\mu$ can cause the adaptation of the coefficients to stop prior to convergence. This phenomenon is especially common when fixed-precision representation is used [28]. For all these reasons, the choice of convergence factor becomes even more difficult.

For the LMSN and RLS algorithms, quantization errors introduced by fixed or floating-point arithmetic can, in addition to the effects described above, lead to divergence of the coefficients, which is unacceptable in many applications whether the

algorithm is able to recover or not. Several studies have attempted to identify the causes of algorithm divergence and to propose solutions. Ljung and Ljung [29] have shown that the RLS algorithm with $\lambda \leq 1$ is stable, since a single error introduced at time instant $n_o$ will decay as time progresses. Notwithstanding this important contribution, it was later verified [30] [31] that interaction and accumulation of errors can destroy the positive definiteness of matrix $\hat{\mathbf{R}}^{-1}(n)$, which can result in divergence. Maintaining the symmetry of $\hat{\mathbf{R}}^{-1}(n)$ is essential to improve robustness; however, in many situations this may not be sufficient. In these cases, rescue procedures, such as adding white noise to the input signal or reinitializing $\hat{\mathbf{R}}^{-1}(n) = \delta\mathbf{I}$, must be used [32].

Since divergence of the RLS algorithm is usually related to the lack of positive definiteness of matrix $\hat{\mathbf{R}}^{-1}(n)$, more robust methods for solving the system of linear equations described in (1.20) can be employed to achieve better results. Given the similarities among the equations that describe the LMSN and RLS algorithms, it is not difficult to find the set of linear equations which is solved by the LMSN algorithm as well. The methods that are applicable to the RLS algorithm can usually be extended to the LMSN algorithm with some effort, if desired. The QR-decomposition RLS algorithm is one of the most popular examples of a technique that has better numerical properties than the conventional RLS algorithm, and yet solves exactly the same set of equations. The orthogonal decomposition is also a fundamental tool in the development and understanding of systolic array structures [8] which are attractive for VLSI implementation. Combined, these properties make the study of the QR-decomposition-based algorithms important and enlightening.

## 1.7 QR-LMSN Algorithm

Given the perfect correspondence between the LMSN and RLS algorithms for $2\mu = \alpha = 1 - \lambda$, this section will describe the derivation of the QR-LMSN algorithm rather than the QR-RLS algorithm, more commonly found in the literature.

As it was briefly mentioned before, the LMSN algorithm can also be regarded as

a least-squares algorithm as it minimizes an objective function defined as

$$\mathcal{E}(n) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} \left[ \frac{2\mu}{\alpha} d(i) + \left(1 - \frac{2\mu}{\alpha}\right) \mathbf{u}^T(i)\mathbf{w}(i-1) - \mathbf{u}^T(i)\mathbf{w}(n) \right]^2 + (1-\alpha)^n \mathbf{w}^T(n)\hat{\mathbf{R}}(0)\mathbf{w}(n) \tag{1.28}$$

by recursively solving the deterministic counterpart of the Wiener-Hopf equation. In this case, $\hat{\mathbf{R}}(n)$ is as in (1.16) and

$$\hat{\mathbf{p}}(n) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} \left[ \frac{2\mu}{\alpha} d(i)\mathbf{u}(i) + \left(1 - \frac{2\mu}{\alpha}\right) \mathbf{u}(i)\mathbf{u}^T(i)\mathbf{w}(i-1) \right]$$

The structure of $\hat{\mathbf{R}}(n)$ given in (1.16) allows for such an elegant and relatively simple method for solving (1.20) using the inverse of $\hat{\mathbf{R}}(n)$ that it is tempting to adopt the LMSN algorithm in its conventional form. Despite its simplicity, the subtraction between the two nonnegative definite matrices that comes naturally in (1.17) is a potential hazard in terms of quantization errors. We must not forget, though, that the LMSN algorithm is just one method for solving $M$ simultaneous linear equations. If machine precision makes the inversion of an ill-conditioned $\hat{\mathbf{R}}(n)$ matrix inaccurate, other methods that have already been used with good results in off-line linear-algebra problems might be more appropriate. One method that has been known for its excellent numerical properties is the QR-decomposition method, which consists of modifying the problem such that the solution can be obtained through the inversion of a triangular matrix. This is possible since every $n \times M$ matrix $\mathbf{U}$ with linearly independent columns can be factored into $\mathbf{U} = \mathbf{Q}^T\mathbf{R}$, where $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is orthogonal and $\mathbf{R} \in \mathbb{R}^{n\times M}$ is upper triangular [33]. No confusion should arise at this point between the QR nomenclature and the spectral decomposition of the autocorrelation matrix mentioned above. For the sake of consistency with the references, let $\mathbf{Q}(n)$ and $\mathbf{R}(n)$ be the QR-decomposition matrices of a data matrix $\mathbf{U}(n)$.

Now let $\mathbf{U}(n)$ be the weighted input-data matrix defined as

$$\mathbf{U}(n) = \mathbf{A}(n) \begin{bmatrix} u(1) & 0 & \cdots & 0 \\ u(2) & u(1) & & 0 \\ \vdots & \vdots & & \vdots \\ u(n) & u(n-1) & \cdots & u(n-M+1) \end{bmatrix}$$

where $\mathbf{A}(n)$ is a weighting matrix defined as

$$\mathbf{A}(n) = \alpha^{\frac{1}{2}} \begin{bmatrix} (1-\alpha)^{\frac{n-1}{2}} & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & (1-\alpha)^{\frac{1}{2}} & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

We can easily verify that $\mathbf{U}^T(n)\mathbf{U}(n)$ is the autocorrelation matrix defined in (1.16) minus the initialization matrix $(1-\alpha)^n\hat{\mathbf{R}}(0)$, and the requirement of linear independence of the columns of $\mathbf{U}(n)$ can be translated as the necessary condition of positive definiteness of (1.16) for a unique solution of the least-squares problems. Furthermore, let the modified weighted reference-data vector be defined as

$$\mathbf{d}(n) = \mathbf{A}(n) \left\{ \frac{2\mu}{\alpha} \begin{bmatrix} d(1) \\ \vdots \\ d(n-1) \\ d(n) \end{bmatrix} + \left(1 - \frac{2\mu}{\alpha}\right) \begin{bmatrix} 0 \\ \mathbf{u}^T(2)\mathbf{w}(1) \\ \vdots \\ \mathbf{u}^T(n)\mathbf{w}(n-1) \end{bmatrix} \right\}$$

If we drop the initialization matrix $\hat{\mathbf{R}}(0)$, the objective function to be minimized, (1.28), can be rewritten as

$$\mathcal{E}(n) = \|\boldsymbol{\epsilon}(n)\|_2^2$$

where

$$\boldsymbol{\epsilon}(n) = \mathbf{d}(n) - \mathbf{U}(n)\mathbf{w}(n)$$

is the modified weighted a posteriori error vector. Since the 2-norm is invariant under orthogonal transformations [33],

$$\mathcal{E}(n) = \|\mathbf{Q}(n)\boldsymbol{\epsilon}(n)\|_2^2$$

can be minimized with respect to $\mathbf{w}(n)$ if $\mathbf{Q}(n)$ is orthogonal. Therefore, by factoring the weighted input-data matrix $\mathbf{U}(n)$ as

$$\mathbf{Q}(n)\mathbf{U}(n) = \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix}$$

where $\mathbf{R}(n)$ is an $M \times M$ upper triangular matrix, the minimization problem becomes

$$\mathcal{E}_{min}(n) = \min_{\mathbf{w}(n)} \|\mathbf{Q}(n)\mathbf{d}(n) - \mathbf{Q}(n)\mathbf{U}(n)\mathbf{w}(n)\|_2^2$$

$$= \min_{\mathbf{w}(n)} \left\| \begin{bmatrix} \mathbf{p}(n) \\ \mathbf{q}(n) \end{bmatrix} - \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix} \mathbf{w}(n) \right\|_2^2$$

where

$$\begin{bmatrix} \mathbf{p}(n) \\ \mathbf{q}(n) \end{bmatrix} = \mathbf{Q}(n)\mathbf{d}(n)$$

The solution is obtained for the vector $\mathbf{w}(n)$ satisfying

$$\mathbf{p}(n) = \mathbf{R}(n)\mathbf{w}(n) \tag{1.29}$$

for which $\mathcal{E}_{min}(n) = \|\mathbf{q}(n)\|_2^2$. For convenience, we have used $\mathbf{R}(n)$ and $\mathbf{p}(n)$ in (1.29), although they do not denote estimates of the autocorrelation matrix of the input signal and the crosscorrelation vector between the input-signal vector and the reference signal. Actually, $\mathbf{R}(n)$ being upper triangular, could never denote an autocorrelation matrix. The solution $\mathbf{w}(n)$ can be easily obtained by applying back substitution to (1.29) [33].

The next step towards a QR-LMSN adaptation algorithm is to find a recursive procedure that can produce the solution with roughly the computational complexity of the conventional LMSN algorithm, i.e., $\mathcal{O}(M^2)$.

Suppose that at instant $n$ the solution obtained at instant $n-1$ is available in the form of matrix $\mathbf{R}(n-1)$ and vector $\mathbf{p}(n-1)$. The updated weighted input-data matrix at instant $n$ is constructed from the previous one as

$$\mathbf{U}(n) = \begin{bmatrix} (1-\alpha)^{\frac{1}{2}}\mathbf{U}(n-1) \\ \alpha^{\frac{1}{2}}\mathbf{u}^T(n) \end{bmatrix}$$

Now let $\mathbf{Q}(n|n-1)$ be the order-updated orthogonal matrix constructed from $\mathbf{Q}(n-1)$ as

$$\mathbf{Q}(n|n-1) = \begin{bmatrix} \mathbf{Q}(n-1) & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

By applying $\mathbf{Q}(n|n-1)$ to $\mathbf{U}(n)$, we have

$$\mathbf{Q}(n|n-1)\mathbf{U}(n) = \begin{bmatrix} (1-\alpha)^{\frac{1}{2}}\mathbf{Q}(n-1)\mathbf{U}(n-1) \\ \alpha^{\frac{1}{2}}\mathbf{u}^T(n) \end{bmatrix} = \begin{bmatrix} (1-\alpha)^{\frac{1}{2}}\mathbf{R}(n-1) \\ \mathbf{0} \\ \alpha^{\frac{1}{2}}\mathbf{u}^T(n) \end{bmatrix} \tag{1.30}$$

Similarly,

$$\mathbf{d}(n) = \begin{bmatrix} (1-\alpha)^{\frac{1}{2}}\mathbf{d}(n-1) \\ \alpha^{\frac{1}{2}}\left[\frac{2\mu}{\alpha}d(n) + \left(1-\frac{2\mu}{\alpha}\right)\mathbf{u}^T(n)\mathbf{w}(n-1)\right] \end{bmatrix}$$

which leads to

$$\mathbf{Q}(n|n-1)\mathbf{d}(n) = \begin{bmatrix} (1-\alpha)^{\frac{1}{2}}\mathbf{p}(n-1) \\ (1-\alpha)^{\frac{1}{2}}\mathbf{q}(n-1) \\ \alpha^{\frac{1}{2}}\left[\frac{2\mu}{\alpha}d(n) + \left(1-\frac{2\mu}{\alpha}\right)\mathbf{u}^T(n)\mathbf{w}(n-1)\right] \end{bmatrix} \tag{1.31}$$

From (1.30) and (1.31) it is clear that to obtain $\mathbf{R}(n)$ and $\mathbf{p}(n)$ from their previous values we must find a transformation matrix $\mathbf{G}(n)$ that makes $\mathbf{Q}(n|n-1)\mathbf{U}(n)$ upper triangular, which would force the elements of the last row of (1.30) to be zeros. The resulting upper triangular matrix is the updated matrix $\mathbf{R}(n)$ in the relation

$$\mathbf{G}(n)\mathbf{Q}(n|n-1)\mathbf{U}(n) = \mathbf{Q}(n)\mathbf{U}(n) = \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix}$$

The vector $\mathbf{p}(n)$ is similarly updated as

$$\mathbf{Q}(n)\mathbf{d}(n) = \begin{bmatrix} \mathbf{p}(n) \\ \mathbf{q}(n) \end{bmatrix}$$

The task of finding $\mathbf{G}(n)$ is much simpler than that of finding $\mathbf{Q}(n)$ since $\mathbf{G}(n)$ need only cancel the last line of (1.30). This can be done by $M$ successive Givens

rotations [8], each of the form

$$\mathbf{G}_k(n) = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & & \cdots & & 0 \\ 0 & 1 & \cdots & 0 & 0 & & \cdots & & 0 \\ \vdots & & \ddots & 0 & 0 & & & & \vdots \\ 0 & & \cdots & 1 & 0 & & \cdots & & 0 \\ 0 & & \cdots & & c_k & 0 & \cdots & 0 & s_k \\ 0 & & \cdots & & 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \cdots & & \vdots & & \ddots & & \vdots \\ 0 & & \cdots & & 0 & & \cdots & 1 & 0 \\ 0 & & \cdots & & -s_k & 0 & \cdots & 0 & c_k \end{bmatrix} ; \quad k = 1, \ldots, M \tag{1.32}$$

where $c_k$ and $s_k$ denote cosines and sines, calculated as

$$c_k = \frac{\beta_{kk}}{\sqrt{\beta_{kk}^2 + \beta_{nk}^2}}$$
$$s_k = \frac{\beta_{nk}}{\sqrt{\beta_{kk}^2 + \beta_{nk}^2}}$$

and $\beta_{kl}$ is the $(k, l)$-element of matrix $\mathbf{G}_{k-1}(n) \cdots \mathbf{G}_1(n)\mathbf{Q}(n|n-1)\mathbf{U}(n)$ which has been modified by the previous rotations. As $\mathbf{G}_k(n)$, $k = 1, \ldots, M$, is very sparse, only the cosines and sines must be calculated. It must be stressed here that although some of the matrices involved grow in size with $n$, only $M \times 1$ vectors and $M \times M$ matrices need to be stored and it is not necessary to compute $\mathbf{q}(n)$.

The last step in the development of the algorithm is to provide an initialization method for $n \leq M$. This is necessary since $\hat{\mathbf{R}}(n)$ is no longer initialized with $\hat{\mathbf{R}}(0)$ and, therefore, the problem is underdetermined for $n < M$. During the initialization an exact procedure can be developed such that $\mathcal{E}_{min}(n) = 0$. The idea is to update $\mathbf{R}(n)$ and $\mathbf{p}(n)$ both in order and in time such that $\mathbf{d}(n) - \mathbf{U}(n)\mathbf{w}(n)$ is always a vector of zeros, starting with $\mathbf{R}(0) = 0$ and $\mathbf{p}(0) = 0$. Although unnecessary for the purposes of the algorithm, it can be easily shown that during this initialization period $\|\mathbf{q}(n)\|_2^2 = 0$.

A summary of the QR-LMSN algorithm is given in Table 1.II.

TABLE 1.II
QR-LMSN ALGORITHM

Available at time instant $n$:
$\mathbf{u}(n) = [u(n)\ u(n-1)\ \cdots\ u(n-M+1)]^T \qquad d(n)$
Initialize:
$\mathbf{R}(0) = \mathbf{0}_{(M\times M)};\ \mathbf{p}(0) = \mathbf{0}_{(M\times 1)};\ \mathbf{x} = \mathbf{0}_{(M\times 1)};\ z = 0;$
$n = 1, 2, \ldots$
{
if $(n <= M)\ N = n$; /* during exact init. */
else $N = M$;
$z = \alpha^{1/2}\left[\frac{2\mu}{\alpha}d(n) + \left(1 - \frac{2\mu}{\alpha}\right)\mathbf{u}^T(n)\mathbf{w}(n-1)\right]$;
$\mathbf{x} = \alpha^{1/2}\mathbf{u}(n)$;
for $(k = 1;\ k \le N;\ k++)$
{
$c = \frac{(1-\alpha)^{1/2}\mathbf{R}_{kk}(n-1)}{\sqrt{(1-\alpha)\mathbf{R}_{kk}^2(n-1)+x_k^2}}$;
$s = \frac{x_k}{\sqrt{(1-\alpha)\mathbf{R}_{kk}^2(n-1)+x_k^2}}$;
for $(j = k;\ j \le N;\ j++)$
{
$\mathbf{R}_{kj}(n) = c(1-\alpha)^{1/2}\mathbf{R}_{kj}(n-1) + sx_j$;
$x_j = -s(1-\alpha)^{1/2}\mathbf{R}_{kj}(n-1) + cx_j$;
}
$\mathbf{p}_k(n) = c(1-\alpha)^{1/2}\mathbf{p}_k(n-1) + sz$;
$z = -s(1-\alpha)^{1/2}\mathbf{p}_k(n-1) + cz$;
}
for $(k = N;\ k >= 1;\ k--)$
{
$\mathbf{w}_k(n) = \mathbf{p}_k(n)$;
for $(j = k+1;\ j <= N;\ j++)$
$\mathbf{w}_k(n) = \mathbf{w}_k(n-1) - \mathbf{R}_{kj}(n)\mathbf{w}_j(n)$;
$\mathbf{w}_k(n) = \mathbf{w}_k(n)/\mathbf{R}_{kk}(n)$;
}
}

## 1.8 Fast Implementations of the LMSN Algorithm

Implementations of Newton-type algorithms which require only $\mathcal{O}(M)$ multiplications per iteration also merit to be discussed here in some detail [8] [34] [35]. They yield computationally more efficient algorithms that can possibly be applied in situations where more time-consuming implementations are prohibitive with the available technology. The diversity of these implementations, individual properties, and origins are enough to set them apart in a new class, called fast algorithms in the literature. Fast implementations of Newton-type algorithms are far too extensive and involved. They will be covered in detail in chapter 4.

## 1.9 Original Contributions

The LSMN and RLS algorithms presented and discussed in the previous sections have much in common, and their differences are often overlooked. The RLS algorithm minimizes an exponentially weighted sum of a posteriori output errors, and its flexibility is limited by $\lambda$ which controls how previous errors are weighted. This algorithm has gained popularity for its high convergence speed. However, high computational complexity, possible instability, and high sensitivity with respect to $\lambda$ have led researchers and engineers to investigate alternative algorithms and implementations (see, e.g., [8] [32] [36]). One such alternative is the LMSN algorithm that employs different convergence factors for updating the coefficients, $\mathbf{w}(n)$, and the estimated autocorrelation matrix, $\hat{\mathbf{R}}(n)$. In the previous sections, it was demonstrated that a condition exists for the two convergence factors that renders the LMSN and the RLS algorithms equivalent. Furthermore, a QR-decomposition LMSN algorithm which is a more robust alternative to the conventional LMSN algorithm was presented.

In chapter 2, the flexibility gained with the two separate convergence factors, $\mu$ and $\alpha$, used for updating $\mathbf{w}(n)$ and $\hat{\mathbf{R}}(n)$, respectively, is further explored. By minimizing the instantaneous a posteriori output error with respect to $\mu$ at every iteration, we obtain a variable convergence factor, $\mu(n)$ [25] [37]. The LMSN algorithm

that employs such convergence factor is analogous to the NLMS algorithm as both rely on an exact line search that yields zero a posteriori output error [15]. When little is known of the environment and an appropriate choice of $\mu$ cannot be made, the variable convergence factor can be especially useful [19]. This chapter also provides a comprehensive study of the convergence in the mean and variance of the algorithm that is general in spite of the scheme used for updating the estimated autocorrelation matrix. Also included is a study of the MSE for fixed-point arithmetic which shows the effects of quantization in all variables upon the total MSE [38]. A similar approach has been used before for the LMS and RLS algorithms [28] [31] [39]. Furthermore, a QR-decomposition LMSN algorithm with variable convergence is developed as an alternative when the finite-precision effects may cause divergence of the conventional implementation [40].

Although the QR-LMSN algorithm shows good results in fixed-point arithmetic with regards to robustness, the algorithm itself is very involved, scaling of internal variables can be very difficult, and robustness cannot be guaranteed once the input signal is highly correlated or nonpersistently exciting. Chapter 3 presents the QN algorithm [41] that can safely operate in fixed-point arithmetic even when the input signal is nonpersistently exciting. This algorithm, developed based on off-line optimization techniques, can be very easily coded and the scaling of its internal variables is trivial. A thorough study of the algorithm is provided, including a proof of stability and a detailed discussion of the behavior of the internal variables of the algorithm [42] [43].

In chapter 4, the subject of fast implementation of the LMSN algorithm with variable convergence factor is investigated. A fast LMSN algorithm with variable convergence factor is presented [44] following closely the approach used for developing the fast RLS algorithms [32] [34] [35]. The necessary initialization of the algorithm during the first iterations is detailed. This chapter provides the basis for understanding fast algorithms in general, and the fast LMSN algorithm with variable convergence factor in particular.

The conclusions of the thesis and several suggestions for further research on

Newton-type adaptation algorithms are summarized in chapter 5.

## 1.10 Conclusions

Numerous structures, algorithms, and implementations have been proposed in recent years for different applications of adaptive filters, each with its own qualities and drawbacks. The choice of the best scheme for a given application must, therefore, be subjected to careful scrutiny. The LMS algorithm has been the most widely used adaptation algorithm, despite its slow convergence in some cases, largely because of its simplicity and robustness. When compared to the LMS algorithm, transform-domain adaptive filters can show improvement in convergence speed at the expense of a relatively small increase in complexity. They have gained popularity especially for processing voice in acoustic echo cancellation. With the development of more robust and computationally efficient techniques and further advancements in VLSI technology, algorithms that employ second-order information are becoming more and more attractive, since they offer improved performance in terms of convergence speed and misadjustment. However, there is still the need to fine tune these algorithms in order to achieve a desired performance. For this reason, the investigation of variable-convergence-factor algorithms is very important. This is the subject of the next chapter.

# Chapter 2

# Variable-Convergence-Factor LMSN Algorithm

## 2.1 Introduction

The appropriate choice of the convergence factor in the LMS and LMSN algorithms and the forgetting factor in the RLS algorithm are key to assuring good performance for the adaptive filter. These choices are environment dependent and optimal fixed values for these factors are difficult to determine especially in nonstationary environments. In order to address this problem, several algorithms have been proposed in the past, in which a variable convergence factor is utilized [45]–[55]. The algorithm in [55] has some attractive properties, such as fast convergence even when the statistics of the input signal are unknown. This algorithm has been successfully applied to adaptive subband filtering, and an improvement in convergence speed over the fixed step-size LMSN algorithm has been observed.

This chapter provides a detailed performance analysis of the LMSN algorithm proposed in [55]. Section 2 describes the use of a variable convergence factor in the LMSN algorithm and proposes a simplified algorithm. The algorithm in [55] as well as the simplified version proposed here [25] [37] [56] are analyzed in stationary and nonstationary environments in section 3 and closed-form formulas for the mean-square error (MSE) are obtained. In section 4, a closed-form formula for the excess MSE

caused by finite-wordlength implementation for the case of fixed-point arithmetic is derived [38]. Section 5 presents experimental results that demonstrate the accuracy of the analysis and the advantages of the algorithms compared with the RLS algorithm.

## 2.2 The Algorithms

A number of problems are associated with the use of fixed convergence factors, as follows:

1. A good choice of convergence factor that results in a high speed of convergence as well as low output MSE is difficult to make and depends on a good knowledge of the environment characteristics.

2. In nonstationary environments, the choice of convergence factor is difficult since the MSE due to nonstationarities can be high if the tracking performance is poor or the MSE due to additive noise can be high if fast convergence is attempted.

3. Algorithms with fixed convergence factors suffer from high sensitivity with respect to their parameters.

4. In certain applications different convergence factors are optimal under different circumstances; for example, in subband filtering a different convergence factor is required for each subband.

Many of the above problems can be eliminated through the use of a variable convergence factor which is adjusted in every iteration according to a certain optimality criterion.

### 2.2.1 Algorithm I

The use of a variable convergence factor in the LMSN algorithm as proposed in [55] will now be examined. The updating formula in (1.14) can be modified to incorporate a time-varying convergence factor

$$\mu(n) = b\alpha(n) \tag{2.1}$$

where $b$ is a constant and $\alpha(n)$ is the convergence factor used to update $\hat{\mathbf{R}}^{-1}(n)$. The choice of $\mu(n)$ must take into account all the available information. One such choice that leads to good results can be deduced by minimizing the square of the a posteriori instantaneous output error, i.e.,

$$\min_{\mu(n)=b\alpha(n)} \epsilon^2(n) = \min_{\mu(n)=b\alpha(n)} \left[d(n) - \mathbf{u}^T(n)\mathbf{w}(n)\right]^2 \tag{2.2}$$

After some manipulation, the solution required is obtained as

$$\alpha(n) = \frac{1}{1 + (2b-1)\tau(n)} \tag{2.3}$$

where

$$\tau(n) = \mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$$

Since matrix $\hat{\mathbf{R}}^{-1}(n-1)$ is positive definite, at least in the case where infinite-precision arithmetic is used, and $\tau(n)$ is a quadratic function of the input-signal vector, the variable convergence factor given by (2.3) is positive and less than one provided that parameter $b$ is chosen to be greater than 0.5.

Note that for $b \neq 0.5$ in (2.1), the algorithm does not satisfy the condition for the equivalence with the RLS algorithm, $2\mu = \alpha = 1 - \lambda$.

It can be shown from (2.3) that

$$\frac{1-\alpha(n)}{\alpha(n)} = (2b-1)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$$

Therefore,

$$\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n) = \frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}{2b\alpha(n)\tau(n)}$$

The equations describing algorithm I can be deduced from those of the LMSN algorithm as follows:

$$\mathbf{t}(n) = \hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n) \tag{2.4}$$

$$\tau(n) = \mathbf{u}^T(n)\mathbf{t}(n) \tag{2.5}$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \frac{\mathbf{t}(n)e(n)}{\tau(n)} \tag{2.6}$$

$$\hat{\mathbf{R}}^{-1}(n) = \frac{1 + (2b-1)\tau(n)}{(2b-1)\tau(n)}\left[\hat{\mathbf{R}}^{-1}(n-1) - \frac{\mathbf{t}(n)\mathbf{t}^T(n)}{2b\tau(n)}\right] \tag{2.7}$$

### 2.2.2 Memory Interpretations

We will now examine the mode by which variable convergence factors in general, and the one examined here in particular, affect matrix $\hat{\mathbf{R}}(n)$. When the convergence factor used in the evaluation of matrix $\hat{\mathbf{R}}(n)$ is constant in all iterations, the estimate of the input autocorrelation matrix is an exponentially-weighted time average of the outer product $\mathbf{u}(n)\mathbf{u}^T(n)$. However, for a variable convergence factor $\alpha(n)$, matrix $\hat{\mathbf{R}}(n)$ represents a different weighted average. In this case, the relation between $\alpha(n)$ and the weights of the past samples, $\lambda(n)$, can be established as

$$\begin{aligned}\lambda(n) &= 1 - \alpha(n) \\ &= \frac{(2b-1)\tau(n)}{1+(2b-1)\tau(n)}\end{aligned}$$

and (1.16) can be rewritten as

$$\hat{\mathbf{R}}(n) = \prod_{i=1}^{n} \lambda(i)\hat{\mathbf{R}}(0) + \sum_{i=1}^{n} \prod_{j=i+1}^{n} \lambda(j)\left[1-\lambda(i)\right]\mathbf{u}(i)\mathbf{u}^T(i)$$

Note that $\lambda(n)$ is always positive and less than unity. The above equation allows a rough analysis of the influence of constant $b$ in the memory of the algorithm that estimates $\hat{\mathbf{R}}(n)$. For example, $b \to 0.5$ results in $\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n) = 1$. When $b$ is large, the algorithm has long memory and $\hat{\mathbf{R}}(n)$ tends to become invariant for large $n$ and stationary input signals. On the other hand, as $b \to 0.5$, we have $\alpha(n) \to 1$ for all $n$. Therefore, the weights $\lambda(n)$ are all equal to zero. In this case, the algorithm has no memory to evaluate $\hat{\mathbf{R}}(n)$ and also there is perfect correspondence between the LMSN and RLS algorithms. In fact, the algorithm would minimize a weighted sum of past a posteriori errors according to the least-squares principle and also the instantaneous a posteriori error due to the variable convergence factor. The result is a solution that does not take into account the past a posteriori errors and, as a consequence, matrix $\hat{\mathbf{R}}(n)$ carries only the information contained in the last input-signal vector, $\mathbf{u}(n)$. The time-dependent characteristic of the forgetting factor $\lambda(n)$ used in the updating of $\hat{\mathbf{R}}^{-1}(n)$ introduces large variations in this matrix which can be undesirable in cases where matrix $\mathbf{R}$ has a high eigenvalue spread since $\hat{\mathbf{R}}^{-1}(n)$ becomes ill-conditioned.

### 2.2.3 Algorithm II

An alternative algorithm to the one proposed in [55] can be derived by noting that the development of the variable convergence factor based on the minimization of (2.2) does not necessarily require (2.1) to be satisfied. Consequently, a variable convergence factor $\mu(n)$ can be chosen to minimize the instantaneous a posteriori error while a fixed $\alpha(n)$, i.e., $\alpha(n) = \alpha$, can be used in the recursion that determines $\hat{\mathbf{R}}^{-1}(n)$. The resulting convergence factor is given by

$$\mu(n) = \frac{1}{2\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n)} \tag{2.8}$$

and yields zero a posteriori error regardless of how matrix $\hat{\mathbf{R}}^{-1}(n)$ is estimated. Note that this is also the case for the convergence factors proposed in [15] for the LMS algorithm and in [55] for the LMSN algorithm for any value of $b$.

The updating equations for algorithm II comprise (2.4)–(2.6) together with the following two equations:

$$z(n) = \frac{1-\alpha}{\alpha} + \tau(n)$$

$$\hat{\mathbf{R}}^{-1}(n) = \frac{1}{1-\alpha}\left[\hat{\mathbf{R}}^{-1}(n-1) - \frac{\mathbf{t}(n)\mathbf{t}^T(n)}{z(n)}\right] \tag{2.9}$$

### 2.2.4 Comparison of Algorithms I and II

Algorithms I and II are, in fact, different versions of the orthogonalized-projection algorithm [24][57] which employ different methods for the estimation of matrix $\mathbf{R}^{-1}$. Algorithm II uses an exponentially weighted average to compute $\hat{\mathbf{R}}(n)$, while algorithm I uses variable weights. The advantages of the one over the other depend heavily on the application. Algorithm II provides better control over matrix $\hat{\mathbf{R}}(n)$ since a constant convergence factor is used, while algorithm I is more attractive when little knowledge about the input signal is available. There is only a single free parameter in each algorithm, $b$ for algorithm I and $\alpha$ for algorithm II. These parameters do not influence directly the speed of convergence and the excess MSE. Their main role is to control the amount of memory that will be used in the evaluation of $\hat{\mathbf{R}}^{-1}(n)$

or, in other words, how accurately $\hat{\mathbf{R}}^{-1}(n)$ approximates $\mathbf{R}^{-1}$. Algorithm I presents an extra cost in terms of computational complexity of two multiplications and one division when compared to the conventional RLS algorithm, whereas algorithm II has the same complexity.

In many applications, it may be desirable to control the misadjustment at the expense of convergence speed. This can be achieved in both algorithms by introducing a reduction factor $q$ in the coefficient updating equation, as will become clear after the analysis to be carried out in the next section. This parameter can be used to perform fine tuning of the algorithms. Eq. (2.6) becomes

$$\mathbf{w}(n) = \mathbf{w}(n-1) + q\frac{\mathbf{t}(n)e(n)}{\tau(n)} \tag{2.10}$$

where $q$ is the reduction factor and $\mathbf{t}(n)$ and $\tau(n)$ are computed as before.

## 2.3 Analysis of Algorithms I and II

In this section, expressions for the MSE in algorithms I and II are derived for the cases of stationary and nonstationary environments.

Owing to the presence of the normalization factor, $\tau(n)$, the computation of $E[1/\tau(n)]$ becomes necessary during the analysis. The approximation

$$E\left[\frac{1}{\tau(n)}\right] \approx \frac{1}{E[\tau(n)]} = \frac{1}{M} \tag{2.11}$$

will be made, which, for ergodic processes, corresponds to interchanging the harmonic and arithmetic averages of an infinitely long sequence $\{\tau(n)\}, n \geq 0$. For white Gaussian noise signals and $M > 20$, the approximation introduces an error of less than 10%, which is reduced further as $M$ is increased above 20. However, for many practical input signals this approximation may not be valid. Similar approximations were used by Bottomley and Alexander [31], and Samson and Reddy [58] for slowly varying denominators. A more accurate approximation, based on the kurtosis of the signal, can be found in [19].

### 2.3.1 Excess MSE in Stationary Environments

Assuming that $\hat{\mathbf{R}}(n-1)$ and $\mathbf{w}(n-1)$ are independent of $\mathbf{u}(n)$ [13] [60], the mean value of the coefficient vector, for both algorithms, is obtained from (2.10) as

$$\begin{aligned} E[\mathbf{w}(n)] &= \prod_{i=1}^{n} \left\{ 1 - E\left[\frac{q}{\tau(i)}\right] \right\} E[\mathbf{w}(0)] \\ &\quad + \sum_{i=1}^{n} \prod_{j=i+1}^{n} \left\{ 1 - E\left[\frac{q}{\tau(j)}\right] \right\} E\left[\frac{q}{\tau(i)}\right] \mathbf{w}_o \\ &\approx \left(\frac{M-q}{M}\right)^n E[\mathbf{w}(0)] + \left[1 - \left(\frac{M-q}{M}\right)^n\right] \mathbf{w}_o \end{aligned} \tag{2.12}$$

The speed of convergence of the algorithm can be roughly measured using (2.12). For an initially relaxed system, we can expect convergence of all coefficients to within 10% of the relative error between $E[\mathbf{w}(n)]$ and $\mathbf{w}_o$ after $n'$ iterations where

$$n' = -\frac{1}{\log\left(1 - \frac{q}{M}\right)}$$

For the LMSN algorithm with fixed convergence factor,

$$n' = -\frac{1}{\log\left(1 - 2\mu\right)}$$

iterations are required.

Defining $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_o$ as the coefficient error vector, we have

$$\mathbf{v}(n) = \left[\mathbf{I} - q\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}\right] \mathbf{v}(n-1) + q\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)e_o(n)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)} \tag{2.13}$$

where $e_o(n) = d(n) - \mathbf{u}^T(n)\mathbf{w}_o$ is considered a zero-mean white sequence. The error in the coefficients $\mathbf{v}(n)$ results in an excess MSE that can be expressed as [12]

$$J_{ex}(n)_r = E\left[\mathbf{v}^T(n)\mathbf{R}\mathbf{v}(n)\right] = \mathrm{tr}\left[\mathbf{R}\mathbf{K}(n)\right] \tag{2.14}$$

where $\mathbf{K}(n)$ is the covariance matrix of $\mathbf{v}(n)$.

From (2.14), the difference equation describing the excess MSE is obtained as

$$
\begin{aligned}
J_{ex}(n)_r &= J_{ex}(n-1)_r \\
&- q\ \mathrm{tr}\left\{\mathbf{R}E\left[\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{v}(n-1)\mathbf{v}^T(n-1)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}\right]\right\} \\
&- q\ \mathrm{tr}\left\{\mathbf{R}E\left[\frac{\mathbf{v}(n-1)\mathbf{v}^T(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}\right]\right\} \\
&+ q^2\ \mathrm{tr}\left\{\mathbf{R}E\left[\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{v}(n-1)\mathbf{v}^T(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\left[\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\right]^2}\right]\right\} \\
&+ q^2\ \mathrm{tr}\left\{\mathbf{R}E\left[\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\left[\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\right]^2}\right]\right\} J_{min}
\end{aligned}
\tag{2.15}
$$

Evaluating each term in the above equation separately and assuming that the model $\hat{\mathbf{R}}(n-1) = \mathbf{R} + \Delta\mathbf{R}(n-1)$ holds [8], it can be easily shown by interchanging the trace and expected value operators that

$$
\begin{aligned}
&q^2\mathrm{tr}\left\{\mathbf{R}E\left[\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{v}(n-1)\mathbf{v}^T(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\left[\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\right]^2}\right]\right\} \\
&\approx q^2\ E\left\{\mathrm{tr}\left[\frac{\mathbf{v}(n-1)\mathbf{v}^T(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)}{\left[\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\right]^2}\right]\right\} \\
&\approx q^2\ E\left\{\mathrm{tr}\left[\frac{\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{v}(n-1)\mathbf{v}^T(n-1)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}\right]\right\} \\
&\approx q^2\ \mathrm{tr}\left\{E\left[\frac{\mathbf{u}(n)\mathbf{u}^T(n)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}\right]\mathbf{K}(n-1)\right\} \\
&\approx q^2 J_{ex}(n-1)_r E\left[\frac{1}{\tau(n)}\right]
\end{aligned}
\tag{2.16}
$$

where the second-order error terms in $\mathbf{R}\hat{\mathbf{R}}^{-1}(n-1)$ were discarded, independence between $\mathbf{v}(n-1)$ and $\mathbf{u}(n)$ was assumed [13] [60], and for large $M$, $\tau(n)$ was considered

independent from each element of $\mathbf{u}(n)\mathbf{u}^T(n)$ taken separately. The fourth term can be similarly simplified to yield

$$q^2 \text{ tr} \left\{ \mathbf{R} E \left[ \frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\left[\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\right]^2} \right] \right\} J_{min} =$$

$$q^2 \; E \left[ \frac{1}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)} \right] J_{min} \quad (2.17)$$

Using (2.16) and (2.17), we can rewrite (2.15) as

$$J_{ex}(n)_r = J_{ex}(n-1)_r - (2q - q^2) E \left[ \frac{1}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)} \right] J_{ex}(n-1)_r$$

$$+ q^2 E \left[ \frac{1}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)} \right] J_{min}$$

After convergence, we obtain

$$J_{ex}(n)_r = \frac{q}{2-q} J_{min} \quad (2.18)$$

This solution is applicable to both algorithms I and II since it is independent of the method used to estimate the input-signal autocorrelation matrix. In fact, it is applicable to orthogonalized projection algorithms in general, as long as the assumptions made in the above analysis are valid.

### 2.3.2 Excess MSE in Nonstationary Environments

In order to study the behavior of algorithms I and II in nonstationary environments, the first-order ARMA model

$$\mathbf{w}_o(n) = \mathbf{w}_o(n-1) + \boldsymbol{\nu}(n)$$

is assumed for the optimal coefficient vector evolution [61], where the elements of $\boldsymbol{\nu}(n)$ are zero-mean white Gaussian noise samples with variance $\sigma_\nu^2$. In this case, the

coefficient error vector is of the form

$$\mathbf{v}(n) = \left[\mathbf{I} - q\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}\right]\mathbf{v}(n-1)$$

$$+ q\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)e_o(n)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)} - \boldsymbol{\nu}(n)$$

where the second term represents an excess MSE due to gradient noise (see eq. (2.13)) and is also present in stationary-environment operation. The excess MSE due to lag [61] can, therefore, be estimated as

$$J_{ex}(n)_l = \mathrm{tr}\left[\mathbf{R}\mathbf{K}'(n)\right]$$

where $\mathbf{K}'(n)$ is the covariance matrix of vector $\mathbf{v}'(n)$ given by

$$\mathbf{v}'(n) = \left[\mathbf{I} - q\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}\right]\mathbf{v}'(n-1) - \boldsymbol{\nu}(n)$$

An analysis similar to that carried out previously for the excess MSE in stationary environments reveals that

$$J_{ex}(n)_l = \frac{\sigma_\nu^2\sigma_u^2M^2}{2q-q^2} \tag{2.19}$$

Therefore, the total excess MSE is given by

$$J_{ex}(n) = \frac{qJ_{min}}{2-q} + \frac{\sigma_u^2\sigma_\nu^2M^2}{2q-q^2} \tag{2.20}$$

### 2.3.3 Minimization of MSE

The excess MSE has a term proportional to the parameter $q$ and a term that is inversely proportional to it. Therefore, an optimal value, $q_o$, that minimizes the total MSE can be deduced as

$$q_o = \frac{aM\sqrt{a^2M^2+4} - a^2M^2}{2}$$

where $a^2 = \sigma_u^2\sigma_\nu^2/J_{min}$.

Applying the same procedure to the conventional LMSN algorithm or the RLS algorithm with $2\mu = 1-\lambda$ yields [61]

$$\mu_o = \frac{a}{2+a} \tag{2.21}$$

In practice, both algorithms must present similar performance if $q/(2M)$ (the average value for the variable convergence factor) in the normalized algorithm is made equal to the value of $\mu$ used in the conventional algorithm. It should be noted that the optimal value of $\mu$, given by (2.21), may sometimes be high. In such cases, the approximations used to obtain (2.21) do not hold, and the true value of the excess MSE would be much larger than expected.

## 2.4 Finite-Precision Effects

In this section, the effect of roundoff errors is examined for the case of fixed-point arithmetic using an approach similar to that in [31]. The following assumptions are made:

1. Numbers are represented using two's-complement format.
2. Additions and subtractions do not introduce quantization errors.
3. Each internal operation is performed with enough bits in the integer part such that no overflow occurs.
4. Quantized versions of all external signals and constants are available.
5. Multiplication and division operations introduce white noise given by

$$\varepsilon(ab) = (ab)_Q - ab \tag{2.22}$$

and

$$\varepsilon(a/b) = (a/b)_Q - a/b$$

respectively.

The quantization error in each variable is defined as the difference between its quantized and nominal values, i.e.,

$$\eta_a = a_Q - a$$

### 2.4.1 Quantization Errors

Algorithms I and II in finite precision are as follows:

$$
\begin{aligned}
e(n)_Q &= d(n) - \left[\mathbf{w}^T(n-1)_Q \mathbf{u}(n)\right]_Q \\
\mathbf{t}(n)_Q &= \left[\hat{\mathbf{R}}^{-1}(n-1)_Q \mathbf{u}(n)\right]_Q \\
\tau'(n)_Q &= \left[\frac{\mathbf{u}^T(n)\mathbf{t}(n)_Q}{N}\right]_Q \\
g'(n)_Q &= \left[\frac{e(n)_Q}{\tau'(n)_Q}\right]_Q \\
\mathbf{r}(n)_Q &= \left[\mathbf{t}(n)_Q g'(n)_Q\right]_Q \\
\mathbf{s}(n)_Q &= \left[\frac{q\mathbf{r}(n)_Q}{N}\right]_Q \\
\mathbf{w}(n)_Q &= \mathbf{w}(n-1)_Q + \mathbf{s}(n)_Q
\end{aligned}
\tag{2.23}
$$

The constant $N$ is a power-of-2 scaling factor that must be introduced to ensure that the values of $\tau'(n)_Q$ lie within the range of the other variables of the algorithm. It can be verified from (2.5) that the mean value of $\tau(n)$ is $M$.

Matrix $\hat{\mathbf{R}}^{-1}(n)_Q$ is updated either as

$$
\hat{\mathbf{R}}^{-1}(n)_Q = \left[\left\{\frac{1 + [(2b-1)\tau(n)_Q]_Q}{[(2b-1)\tau(n)_Q]_Q}\right\}_Q \times \left(\hat{\mathbf{R}}^{-1}(n-1)_Q - \left\{\frac{[\mathbf{t}(n)_Q \mathbf{t}^T(n)_Q]_Q}{[2b\tau(n)_Q]_Q}\right\}_Q\right)\right]_Q
$$

for algorithm I [55] or as

$$
\hat{\mathbf{R}}^{-1}(n)_Q = \left[\frac{1}{1-\alpha}\left(\hat{\mathbf{R}}^{-1}(n-1)_Q - \left\{\frac{[\mathbf{t}(n)_Q \mathbf{t}^T(n)_Q]_Q}{\frac{1-\alpha}{\alpha} + \tau(n)_Q}\right\}_Q\right)\right]_Q
$$

for algorithm II.

Since $e(n) = e(n)_Q - \eta_e(n)$, it follows that

$$
e(n) = d(n) - \mathbf{w}^T(n-1)\mathbf{u}(n) = d(n) - \left[\mathbf{w}^T(n-1)_Q \mathbf{u}(n)\right]_Q - \eta_e(n)
$$

Defining $\varepsilon_1(n)$ as the error introduced when evaluating $\left[\mathbf{w}^T(n-1)_Q\mathbf{u}(n)\right]_Q$ according to (2.22), namely,

$$\varepsilon_1(n) = \varepsilon\left[\mathbf{w}^T(n-1)_Q\mathbf{u}(n)\right] = \left[\mathbf{w}^T(n-1)_Q\mathbf{u}(n)\right]_Q - \mathbf{w}^T(n-1)_Q\mathbf{u}(n)$$

we have

$$d(n) - \mathbf{w}^T(n-1)\mathbf{u}(n) = d(n) - \mathbf{w}^T(n-1)_Q\mathbf{u}(n) - \eta_e(n) - \varepsilon_1(n)$$

which yields

$$\eta_e(n) = -\boldsymbol{\eta}_w^T(n-1)\mathbf{u}(n) - \varepsilon_1(n) \tag{2.24}$$

An analogous procedure leads to similar equations for the errors in $\mathbf{t}(n)$ and $\tau'(n)$, i.e.,

$$\boldsymbol{\eta}_t(n) = \boldsymbol{\eta}_R(n-1)\mathbf{u}(n) + \boldsymbol{\varepsilon}_2(n)$$

$$\eta_{\tau'}(n) = \frac{\mathbf{u}^T(n)\boldsymbol{\eta}_t(n)}{N} + \varepsilon_3(n)$$

where

$$\boldsymbol{\varepsilon}_2(n) = \varepsilon\left[\hat{\mathbf{R}}^{-1}(n-1)_Q\mathbf{u}(n)\right]$$

and

$$\varepsilon_3(n) = \varepsilon\left[\frac{\mathbf{u}^T(n)\mathbf{t}(n)_Q}{N}\right]$$

If we assume that quantization is performed by rounding after the additions, then $\varepsilon_1(n)$, $\varepsilon_3(n)$, and the elements in $\boldsymbol{\varepsilon}_2(n)$ can be modeled as zero-mean white Gaussian noise sources with variance equal to $2^{-2b}/12$ [67].

The errors in $g'(n)$, $\mathbf{r}(n)$, and $\mathbf{s}(n)$ can be modeled as

$$\begin{aligned}\eta_{g'}(n) &= \frac{e(n)_Q}{\tau'(n)_Q} + \varepsilon_4(n) - \frac{e(n)}{\tau'(n)} \\ &\approx \frac{e(n)+\eta_e(n)}{\tau'(n)}\left[1 - \frac{\eta_{\tau'}(n)}{\tau'(n)}\right] - \frac{e(n)}{\tau'(n)} + \varepsilon_4(n) \\ &\approx -\frac{e(n)\eta_{\tau'}(n)}{\tau'^2(n)} + \frac{\eta_e(n)}{\tau'(n)}\left[1 - \frac{\eta_{\tau'}(n)}{\tau'(n)}\right] + \varepsilon_4(n)\end{aligned}$$

$$\boldsymbol{\eta}_r(n) = \mathbf{t}(n)\eta_{g'}(n) + \boldsymbol{\eta}_t(n)g'(n) + \boldsymbol{\varepsilon}_5(n)$$

and

$$\boldsymbol{\eta}_s(n) = \frac{q\boldsymbol{\eta}_r(n)}{N} + \boldsymbol{\varepsilon}_6(n)$$

where

$$\varepsilon_4(n) = \varepsilon\left[\frac{e(n)_Q}{\tau'(n)_Q}\right]$$

$$\boldsymbol{\varepsilon}_5(n) = \varepsilon\left[\mathbf{t}(n)_Q g'(n)_Q\right]$$

and

$$\boldsymbol{\varepsilon}_6(n) = \varepsilon\left[\frac{q\mathbf{r}(n)_Q}{N}\right]$$

From (2.23), the quantization error in the coefficient vector can be expressed in terms of the recursive relation

$$\boldsymbol{\eta}_w(n) = \boldsymbol{\eta}_w(n-1) + \boldsymbol{\eta}_s(n)$$

The quantization errors in the updating of matrix $\hat{\mathbf{R}}^{-1}(n)$, after neglecting all second-order errors, can be represented by

$$\begin{aligned}\boldsymbol{\eta}_R(n) \approx \frac{1}{1-\alpha(n)}\Bigg\{&\boldsymbol{\eta}_R(n-1) - \frac{1}{2b\tau(n)}\left[\mathbf{t}(n)\boldsymbol{\eta}_t^T(n) + \boldsymbol{\eta}_t(n)\mathbf{t}^T(n) + \boldsymbol{\varepsilon}_{11}(n)\right]\\ &+ \frac{\mathbf{t}(n)\mathbf{t}^T(n)}{2b\tau^2(n)}\left[\eta_\tau(n) + \frac{\varepsilon_{12}(n)}{2b}\right] - \boldsymbol{\varepsilon}_{10}(n)\Bigg\}\\ &- \left\{\alpha(n)\left[\frac{\eta_\tau(n)}{\tau(n)} + \frac{\varepsilon_9(n)}{(2b-1)\tau(n)} + \varepsilon_8(n)\right] - \varepsilon_8(n)\right\}\hat{\mathbf{R}}^{-1}(n) + \boldsymbol{\varepsilon}_7(n)\end{aligned}$$

for algorithm I and

$$\begin{aligned}\boldsymbol{\eta}_R(n) \approx \frac{1}{1-\alpha}\Bigg\{&\boldsymbol{\eta}_R(n-1) + \frac{\mathbf{t}(n)\mathbf{t}^T(n)\eta_\tau(n)}{\left[\frac{1-\alpha}{\alpha} + \tau(n)\right]^2}\\ &- \frac{\boldsymbol{\eta}_t(n)\mathbf{t}^T(n) + \mathbf{t}(n)\boldsymbol{\eta}_t^T(n) + \boldsymbol{\varepsilon}_{11}(n)}{\frac{1-\alpha}{\alpha} + \tau(n)} - \boldsymbol{\varepsilon}_{10a}(n)\Bigg\} + \boldsymbol{\varepsilon}_{7a}(n)\end{aligned}$$

for algorithm II, where

$$\boldsymbol{\varepsilon}_7(n) = \varepsilon\left[\left\{\frac{1+[(2b-1)\tau(n)_Q]_Q}{[(2b-1)\tau(n)_Q]_Q}\right\}_Q\left(\hat{\mathbf{R}}^{-1}(n-1)_Q - \left\{\frac{[\mathbf{t}(n)_Q\mathbf{t}(n)_Q^T]_Q}{[2b\tau(n)_Q]_Q}\right\}_Q\right)\right]$$

$$\boldsymbol{\varepsilon}_{7a}(n) = \varepsilon\left[\frac{1}{1-\alpha}\left(\hat{\mathbf{R}}^{-1}(n-1)_Q - \left\{\frac{[\mathbf{t}(n)_Q\mathbf{t}^T(n)_Q]_Q}{\frac{1-\alpha}{\alpha}+\tau(n)_Q}\right\}_Q\right)\right]_Q$$

$$\varepsilon_8(n) = \varepsilon\left\{\frac{1+[(2b-1)\tau(n)_Q]_Q}{[(2b-1)\tau(n)_Q]_Q}\right\}$$

$$\varepsilon_9(n) = \varepsilon\left[(2b-1)\tau(n)_Q\right]$$

$$\boldsymbol{\varepsilon}_{10}(n) = \varepsilon\left\{\frac{[\mathbf{t}(n)_Q\mathbf{t}^T(n)_Q]_Q}{[2b\tau(n)_Q]_Q}\right\}$$

$$\boldsymbol{\varepsilon}_{10a}(n) = \varepsilon\left\{\frac{[\mathbf{t}(n)_Q\mathbf{t}^T(n)_Q]_Q}{\frac{1-\alpha}{\alpha}+\tau(n)_Q}\right\}$$

$$\boldsymbol{\varepsilon}_{11}(n) = \varepsilon\left[\mathbf{t}(n)_Q\mathbf{t}^T(n)_Q\right]$$

$$\varepsilon_{12}(n) = \varepsilon\left[2b\tau(n)_Q\right]$$

### 2.4.2 MSE

By considering the effects of quantization errors on the error signal, the excess MSE due to finite-precision arithmetic can be calculated as

$$\begin{aligned} J_{ex}(n)_p &= E\left[\eta_e^2(n)\right] = E\left\{\mathrm{tr}\left[\mathbf{u}(n)\mathbf{u}^T(n)\boldsymbol{\eta}_w(n-1)\boldsymbol{\eta}_w^T(n-1)\right]\right\} + \sigma_{\varepsilon_1}^2 \\ &= \mathrm{tr}\left[\mathbf{R}\mathbf{K}''(n)\right] + \sigma_{\varepsilon_1}^2 \end{aligned} \tag{2.25}$$

where $\eta_e(n)$ is defined in (2.24) and

$$\mathbf{K}''(n) = E\left[\boldsymbol{\eta}_w(n-1)\boldsymbol{\eta}_w^T(n-1)\right]$$

In (2.25), it was assumed that $\varepsilon_1(n)$ is a zero-mean white Gaussian noise with variance equal to $\sigma_{\varepsilon_1}^2$.

The difference equation that gives $\boldsymbol{\eta}_w(n)$ can be expressed as a function of instantaneous quantization errors, infinite-precision quantities, and $\boldsymbol{\eta}_R(n-1)$ as

$$\begin{aligned}\boldsymbol{\eta}_w(n) \approx & \left[\mathbf{I} - q\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)}{\tau(n)}\right]\boldsymbol{\eta}_w(n-1) \\ & + q\frac{e(n)}{\tau(n)}\left[\mathbf{I} - \frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)}{\tau(n)}\right][\boldsymbol{\eta}_R(n-1)\mathbf{u}(n) + \boldsymbol{\varepsilon}_2(n)] \\ & + q\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\left[\frac{\varepsilon_4(n)}{N} - \frac{\varepsilon_1(n)}{\tau(n)} - \frac{e(n)\varepsilon_3(n)N}{\tau^2(n)}\right] + \frac{q}{N}\boldsymbol{\varepsilon}_5(n) + \boldsymbol{\varepsilon}_6(n) \quad (2.26)\end{aligned}$$

Considering that the error sources in (2.26) are uncorrelated with each other and that the output error after convergence, $e(n)$, can be modeled as a zero-mean white noise, (2.25)–(2.26) give

$$\begin{aligned}J_{ex}(n+1)_p = & \operatorname{tr}\left(\mathbf{R}E\left\{\left[\mathbf{I} - q\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)}{\tau(n)}\right]\boldsymbol{\eta}_w(n-1)\right.\right. \\ & \left.\left.\times\boldsymbol{\eta}_w^T(n-1)\left[\mathbf{I} - q\frac{\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\tau(n)}\right]\right\}\right) \\ & + q^2 J(n)\operatorname{tr}\left(\mathbf{R}E\left\{\frac{1}{\tau^2(n)}\left[\mathbf{I} - \frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)}{\tau(n)}\right][\boldsymbol{\eta}_R(n-1)\mathbf{u}(n) + \boldsymbol{\varepsilon}_2(n)]\right.\right. \\ & \left.\left.\times\left[\mathbf{u}^T(n)\boldsymbol{\eta}_R^T(n-1) + \boldsymbol{\varepsilon}_2^T(n)\right]\left[\mathbf{I} - \frac{\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\tau(n)}\right]\right\}\right) \\ & + q^2 E\left[\left(\frac{\varepsilon_4(n)}{N} - \frac{\varepsilon_1(n)}{\tau(n)} - \frac{e(n)\varepsilon_3(n)N}{\tau^2(n)}\right)^2\right] \\ & \times \operatorname{tr}\left\{\mathbf{R}E\left[\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\right]\right\} \\ & + \frac{q^2}{N^2}\sigma_{\varepsilon_5}^2\operatorname{tr}(\mathbf{R}) + \sigma_{\varepsilon_6}^2\operatorname{tr}(\mathbf{R}) + \sigma_{\varepsilon_1}^2\end{aligned}$$

The above equation can be solved by assuming independence between $\tau(n)$ and each element of $\mathbf{u}(n)\mathbf{u}^T(n)$ taken separately, and by neglecting the second-order terms in

$\mathbf{R}\hat{\mathbf{R}}^{-1}(n-1)$. If we also assume that

$$\frac{J(n)}{M^2} \ll 1$$

and use the approximation in (2.11), then

$$J_{ex}(n)_p = \frac{M^2\sigma_{\varepsilon_6}^2\sigma_u^2 + (M+q^2)\sigma_{\varepsilon_1}^2 + q^2M^2\sigma_{\varepsilon_4}^2/N^2 + q^2M^2\sigma_{\varepsilon_5}^2\sigma_u^2/N^2}{q(2-q)} \tag{2.27}$$

If, in addition, $\sigma_{\varepsilon_i}^2 = \sigma_{\varepsilon_j}^2 \equiv \sigma_\varepsilon^2$, then

$$J_{ex}(n)_p = \sigma_\varepsilon^2 \frac{M^2\sigma_u^2 + M + q^2\left[1 + \frac{M^2}{N^2}(1+\sigma_u^2)\right]}{q(2-q)} \tag{2.28}$$

Unfortunately, for both methods of evaluation of the quantized version of $\mathbf{R}^{-1}(n)$, namely, $\mathbf{R}^{-1}(n)_Q$, either with variable or constant convergence factor, positive definiteness is not guaranteed, as can be shown by using an analysis similar to that in [31]. This may result in divergence of the coefficients and a stabilization scheme must be incorporated, for example, the one proposed in [31]. This strategy is applicable since $\eta_R$ has no significant influence on $J_{ex}(n)_p$, as can be seen in (2.27). Another solution for preventing divergence of the algorithm relies on an implementation based on the QR decomposition. This technique was successfully applied in [40] to the variable-convergence-factor LMSN algorithm discussed here, and found to yield very good results in finite precision.

## 2.5 QR-LMSN Algorithm

As an attempt to prevent divergence caused by the bad numerical properties of the matrix inversion lemma used in the variable-convergence-factor algorithms presented in this chapter, a QR-LMSN algorithm can be derived. It is based on algorithm II, as it uses a constant convergence factor for the computation of matrix $\hat{\mathbf{R}}(n)$. It can be readily verified that algorithm II minimizes an objective function in the form of (1.28), except that $\mu$ is time-dependent. More specifically, $\mu(n)$ is calculated as in (2.8). Since for the QR-LMSN algorithm, the coefficient vector is obtained without explicitly evaluating the autocorrelation matrix or its inverse, the derivation of the

convergence factor must be carried out in a different way. Notice that the upper triangular matrix $\mathbf{R}(n)$ in the QR-decomposition algorithms is not an estimate of the autocorrelation matrix, denoted as $\hat{\mathbf{R}}(n)$ in the previous sections.

From (1.30), it is easy to verify that

$$\mathbf{G}(n)\mathbf{Q}(n|n-1)\mathbf{U}(n) = \mathbf{G}(n)\begin{bmatrix}(1-\alpha)^{\frac{1}{2}}\mathbf{R}(n-1)\\ \mathbf{0}\\ \alpha^{\frac{1}{2}}\mathbf{u}^T(n)\end{bmatrix} = \begin{bmatrix}\mathbf{R}(n)\\ \mathbf{0}\end{bmatrix} \tag{2.29}$$

Similarly, from (1.31)

$$\begin{aligned}\mathbf{G}(n)\mathbf{Q}(n|n-1)\mathbf{d}(n) &= \mathbf{G}(n)\begin{bmatrix}(1-\alpha)^{\frac{1}{2}}\mathbf{p}(n-1)\\ (1-\alpha)^{\frac{1}{2}}\mathbf{q}(n-1)\\ \alpha^{\frac{1}{2}}\left\{\gamma(n)d(n)+[1-\gamma(n)]\,\mathbf{u}^T(n)\mathbf{w}(n-1)\right\}\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{p}(n)\\ \mathbf{q}'(n)\\ q_n(n)\end{bmatrix}\end{aligned} \tag{2.30}$$

where $\mathbf{q}'(n)$ denotes the first $(n-M-1)$ elements of $\mathbf{q}(n)$, $q_n(n)$ is the last element of $\mathbf{q}(n)$, and

$$\gamma(n) = \frac{2\mu(n)}{\alpha}$$

Therefore, from (1.29), (2.29), and (2.30) we have

$$\mathbf{G}(n)\begin{bmatrix}(1-\alpha)^{\frac{1}{2}}\mathbf{R}(n-1)\\ \mathbf{0}\\ \alpha^{\frac{1}{2}}\mathbf{u}^T(n)\end{bmatrix}\mathbf{w}(n) =$$

$$\mathbf{G}(n)\begin{bmatrix}(1-\alpha)^{\frac{1}{2}}\mathbf{p}(n-1)\\ (1-\alpha)^{\frac{1}{2}}\mathbf{q}(n-1)\\ \alpha^{\frac{1}{2}}\left\{\gamma(n)d(n)+[1-\gamma(n)]\,\mathbf{u}^T(n)\mathbf{w}(n-1)\right\}\end{bmatrix} - \begin{bmatrix}\mathbf{0}\\ \mathbf{q}'(n)\\ q_n(n)\end{bmatrix}$$

or, equivalently,

$$\mathbf{G}^{-1}(n)\begin{bmatrix}\mathbf{0}\\ \mathbf{q}'(n)\\ q_n(n)\end{bmatrix} =$$

$$\begin{bmatrix}(1-\alpha)^{\frac{1}{2}}\mathbf{p}(n-1)\\ (1-\alpha)^{\frac{1}{2}}\mathbf{q}(n-1)\\ \alpha^{\frac{1}{2}}\left\{\gamma(n)d(n)+[1-\gamma(n)]\,\mathbf{u}^T(n)\mathbf{w}(n-1)\right\}\end{bmatrix} - \begin{bmatrix}(1-\alpha)^{\frac{1}{2}}\mathbf{R}(n-1)\\ \mathbf{0}\\ \alpha^{\frac{1}{2}}\mathbf{u}^T(n)\end{bmatrix}\mathbf{w}(n)$$

Therefore,

$$\alpha^{\frac{1}{2}}\left\{\gamma(n)d(n)+[1-\gamma(n)]\,\mathbf{u}^T(n)\mathbf{w}(n-1)-\mathbf{u}^T(n)\mathbf{w}(n)\right\} = \mathbf{g}_n^T(n)\begin{bmatrix}\mathbf{0}\\ \mathbf{q}'(n)\\ q_n(n)\end{bmatrix}$$

where $\mathbf{g}_n^T(n)$ is the last row of matrix $\mathbf{G}^{-1}(n)$. But from (1.32) we know that $\mathbf{G}^{-1}(n) = \mathbf{G}^T(n)$, since $\mathbf{G}(n)$ is orthogonal and, therefore, $\mathbf{g}_n^T(n)$ is of the form

$$\mathbf{g}_n^T(n) = \begin{bmatrix} g_{n1}(n)\; g_{n2}(n)\; \cdots\; g_{nM}(n) & \mathbf{0}_{n-M-1} & \prod_{i=1}^{M} c_i\end{bmatrix}$$

where $c_i$ is the cosine of the $i$th Givens rotation as defined in the previous chapter. Given the structure of $\mathbf{g}_n(n)$ described above, we have

$$\mathbf{g}_n^T(n)\begin{bmatrix}\mathbf{0}\\ \mathbf{q}'(n)\\ q_n(n)\end{bmatrix} = \prod_{i=1}^{M} c_i q_n(n)$$

and, consequently,

$$\alpha^{\frac{1}{2}}\left\{\gamma(n)d(n)+[1-\gamma(n)]\,\mathbf{u}^T(n)\mathbf{w}(n-1)-\mathbf{u}^T(n)\mathbf{w}(n)\right\} = \prod_{i=1}^{M} c_i q_n(n) \qquad (2.31)$$

From (2.31), we can obtain a value of $\gamma(n)$ without explicitly calculating $\hat{\mathbf{R}}^{-1}(n)$. The variable convergence factor $\mu(n)$ used in algorithm II is such that the a posteriori

output error is zero, i.e.,

$$\epsilon(n) = d(n) - \mathbf{u}^T(n)\mathbf{w}(n) = 0$$

Therefore, we must force the above relation on (2.31) as well, which gives

$$\alpha^{\frac{1}{2}}[\gamma(n) - 1]e(n) = \prod_{i=1}^{M} c_i q_n(n) \tag{2.32}$$

The difficulty in the equation above is that $q_n(n)$ itself depends on $\gamma(n)$. But from the definition of the Givens rotations $\mathbf{G}_k(n)$ in (1.32) we can readily derive

$$q_n(n) = q_1(n) + q_2(n)$$

where

$$q_1(n) = \prod_{i=1}^{M} c_i \alpha^{\frac{1}{2}} \left[\gamma(n)e(n) + \mathbf{u}^T(n)\mathbf{w}(n-1)\right]$$

and

$$q_2(n) = -(1-\alpha)^{\frac{1}{2}} \sum_{i=1}^{M} s_i p_i(n-1) \prod_{j=i+1}^{M} c_j$$

with $s_i$ and $c_i$ denoting the sines and cosines of the Givens rotations as in (1.32), and $p_i(n-1)$ denoting the $i$th element of $\mathbf{p}(n-1)$. Solving (2.32) for $\gamma(n)$, we get

$$\gamma(n) = \frac{\prod_{i=1}^{M} c_i q_2(n) + \alpha^{\frac{1}{2}} \mathbf{u}^T(n)\mathbf{w}(n-1) \prod_{i=1}^{M} c_i^2 + \alpha^{\frac{1}{2}} e(n)}{\alpha^{\frac{1}{2}} e(n) \left(1 - \prod_{i=1}^{M} c_i^2\right)}$$

The introduction of the variable convergence factor in the QR-LMSN algorithm causes only a small increase in the computational complexity of the algorithm, although the equations are somehow more complex than those of the fixed-convergence-factor algorithm. Notice that variable $q_2(n)$ can be obtained as a byproduct during the computation of $\mathbf{p}(n)$, since

$$\begin{bmatrix} \mathbf{p}_1(n) \\ \mathbf{q}_1'(n) \\ q_1(n) \end{bmatrix} = \mathbf{G}(n) \begin{bmatrix} (1-\alpha)^{\frac{1}{2}} \mathbf{p}(n-1) \\ (1-\alpha)^{\frac{1}{2}} \mathbf{q}(n-1) \\ \mathbf{0} \end{bmatrix}$$

and

$$\begin{bmatrix} \mathbf{p}_2(n) \\ \mathbf{q}_2'(n) \\ q_2(n) \end{bmatrix} = \mathbf{G}(n) \begin{bmatrix} \mathbf{0} \\ \alpha^{\frac{1}{2}} \left\{ \gamma(n) d(n) + [1 - \gamma(n)] \, \mathbf{u}^T(n) \mathbf{w}(n-1) \right\} \end{bmatrix}$$

where

$$\mathbf{p}(n) = \mathbf{p}_1(n) + \mathbf{p}_2(n)$$

$$\mathbf{q}(n) = \mathbf{q}_1(n) + \mathbf{q}_2(n)$$

Therefore, we can first obtain $\mathbf{p}_2(n)$ and $q_2(n)$, calculate the value of $\gamma(n)$ that yields zero a posteriori error according to (2.5), and then use this value to calculate $\mathbf{p}_1(n)$. As opposed to the QR-LMSN algorithm with constant convergence factor, the QR-decomposition version of algorithm II requires that all sines and cosines be stored. However, the increase in computational complexity and memory-allocation space is only marginal. The benefits brought by the variable convergence factor are well worth it, as the experiments described in the next section clearly demonstrate.

As a final note on $\gamma(n)$, from (2.8) and (2.9) it can be easily verified that

$$\gamma(n) = 1 + \frac{1 - \alpha}{\alpha \tau(n)} > 1$$

Therefore, using the result in (2.11) we can obtain

$$E\left[\gamma(n)\right] = 1 + \frac{1 - \alpha}{\alpha M}$$

A summary of the variable-convergence-factor QR-LMSN algorithm is given in Table 2.I.

## 2.6 Simulations

In order to verify the usefulness of the algorithms examined and the accuracy of the formulas presented in the preceding sections, several simulations have been performed. In all of them the adaptive filter was used to identify an unknown system modeled as

TABLE 2.I
VARIABLE-CONVERGENCE-FACTOR QR-LMSN ALGORITHM

Available at time instant $n$:

$\mathbf{u}(n) = [u(n)\ u(n-1)\ \cdots\ u(n-M+1)]^T \qquad d(n)$

Initialize:

$\mathbf{R}(0) = \mathbf{0}_{(M\times M)};\ \mathbf{p}(0) = \mathbf{0}_{(M\times 1)};\ \mathbf{x} = \mathbf{0}_{(M\times 1)};\ z = 0;$

$n = 1, 2, \ldots$

{

if $(n <= M)\ N = n$; /* during exact init. */

else $N = M$;

$x = \alpha^{1/2}\mathbf{u}(n);\ q_2 = 0;\ g = 1;$

for $(k = 1;\ k \le N;\ k++)$

{

$c_k = \frac{(1-\alpha)^{1/2}\mathbf{R}_{kk}(n-1)}{\sqrt{(1-\alpha)\mathbf{R}_{kk}^2(n-1)+x_k^2}};$

$s_k = \frac{x_k}{\sqrt{(1-\alpha)\mathbf{R}_{kk}^2(n-1)+x_k^2}};$

for $(j = k;\ j \le N;\ j++)$

{

$\mathbf{R}_{kj}(n) = c_k(1-\alpha)^{1/2}\mathbf{R}_{kj}(n-1) + s_k x_j;$

$x_j = -s_k(1-\alpha)^{1/2}\mathbf{R}_{kj}(n-1) + c_k x_j;$

}

$\mathbf{p}_k(n) = c_k(1-\alpha)^{1/2}\mathbf{p}_k(n-1) + sq_2;$

$q_2 = -s(1-\alpha)^{1/2}\mathbf{p}_k(n-1) + cq_2;$

$g = gc_k;$

}

$\gamma = \frac{gq_2+\alpha^{1/2}\left(g^2-1\right)\mathbf{u}^T(n)\mathbf{w}(n-1)+\alpha^{1/2}d(n)}{\alpha^{1/2}(1-g^2)[d(n)-\mathbf{u}^T(n)\mathbf{w}(n-1)]}$

$z = \alpha^{1/2}\left[\gamma d(n) + (1-\gamma)\,\mathbf{u}^T(n)\mathbf{w}(n-1)\right];$

for $(k = 1;\ k \le N;\ k++)$

{

$\mathbf{p}_k(n) = \mathbf{p}_k(n) + s_k z;$

$z = c_k z;$

}

$z = z + q_2;$

for $(k = N;\ k >= 1;\ k--)$

{

$\mathbf{w}_k(n) = \mathbf{p}_k(n);$

for $(j = k+1;\ j <= N;\ j++)$

$\mathbf{w}_k(n) = \mathbf{w}_k(n-1) - \mathbf{R}_{kj}(n)\mathbf{w}_j(n);$

$\mathbf{w}_k(n) = \mathbf{w}_k(n)/\mathbf{R}_{kk}(n);$

}

}

a moving average process, with additional noise present. Both the additive noise and the input signal to the unknown system were pseudo-random sequences with normal distribution and zero mean.

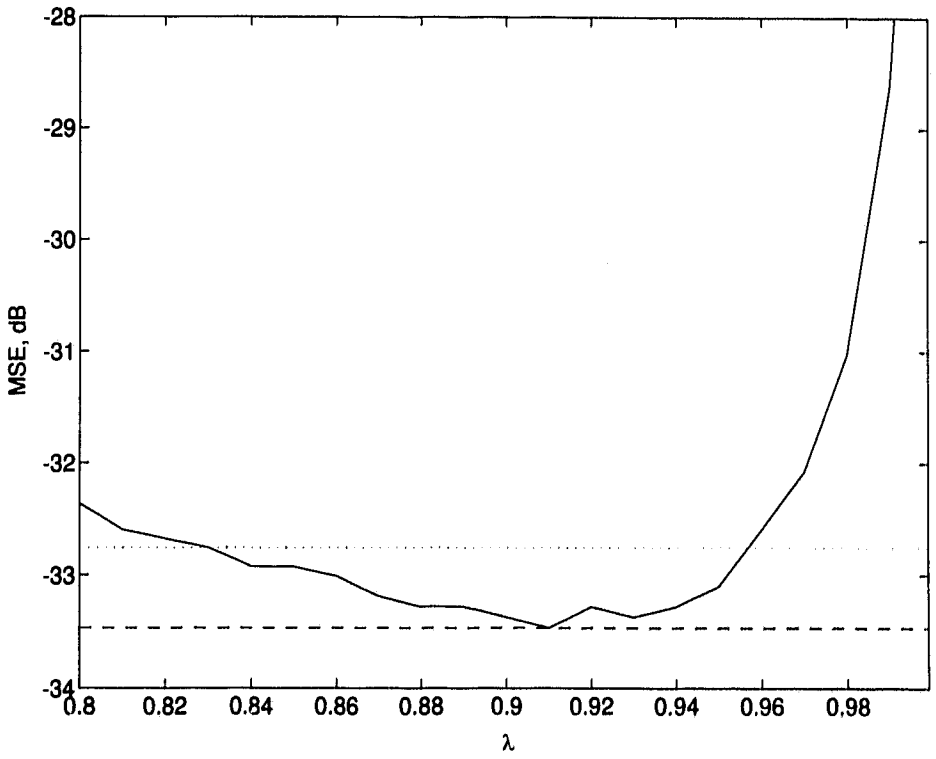



Figure 2.1: Mean-squared error in nonstationary environment ($M = 23$).
RLS algorithm: ——
Best Performance of Algorithm I: - - -
Worst Performance of Algorithm I: · · ·

The first and second experiments compare the mean-squared error of the variable-convergence-factor LMSN and the RLS algorithms. The additive noise variance in the first experiment was −80 dB and the input-signal variance was 0 dB. Figure 2.1 shows the ensemble average of the MSE obtained in 100 simulations. Different values of the forgetting factor are used for the RLS algorithm. The unknown system was a 22nd-order nonstationary system with coefficients modeled as a first-order ARMA process with a relaxation time constant of 1.0. The variance of the white noise used as

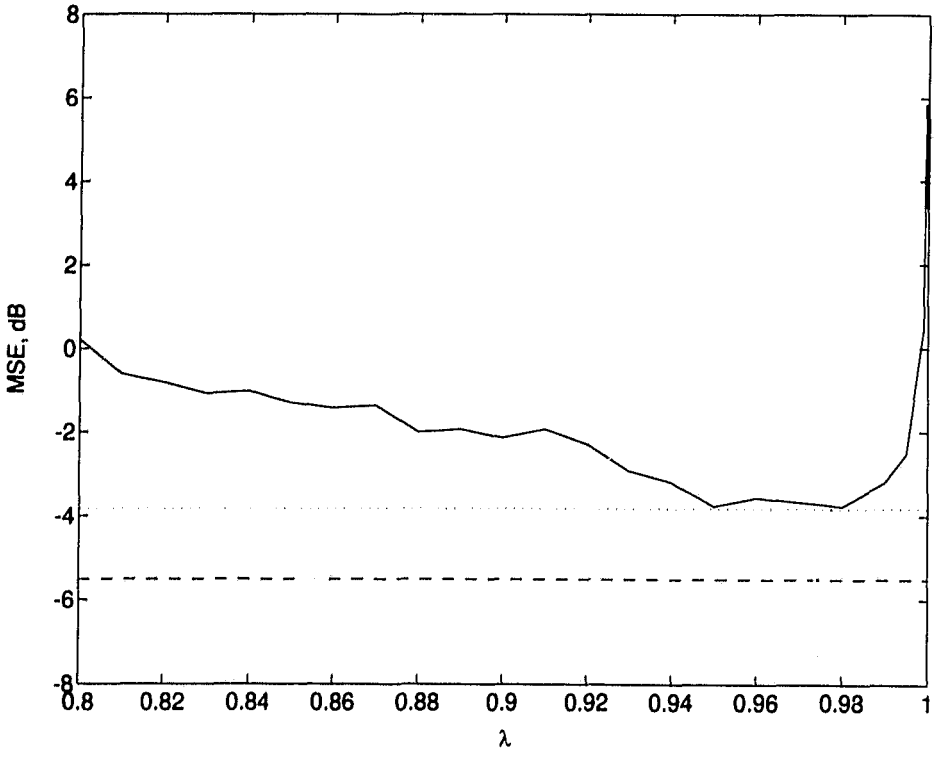



Figure 2.2: Mean-squared error in nonstationary environment ($M = 65$).
RLS algorithm: ——
Best performance of algorithm I: - - -
Worst performance of algorithm I: · · ·

input sequence to the first-order ARMA process was −60 dB. The horizontal dashed and dotted lines represent the best and the worst performances obtained, respectively, with algorithm I for $q = 1$ and $b$ ranging from 0.6 to 100.0. Figure 2.2 shows the results obtained when a 64th-order nonstationary system was to be identified. The additive noise and the input-signal noise variance were −30 dB. The variance of the sequence used as input for the first-order ARMA process was −10 dB. It is easy to note from these examples how badly the conventional RLS algorithm performs when the value of the forgetting factor is not close to its optimal value, and how difficult it is to find an optimal value when the characteristics of the environment are unknown. On the other hand, when a variable convergence factor was used without any tuning,

very good results were obtained especially for the high-order filter.

The results of other experiments are presented in tables, where the theoretical values are compared with those obtained by averaging the results of an ensemble consisting of 100 experiments. The performance parameter shown is the excess MSE, i.e., the difference between the MSE and the variance of the additive noise.

In the third experiment, an adaptive filter based on algorithm II was used to identify a 35th-order nonstationary system with a relaxation time constant of 0.999. The system's time-varying coefficients were modeled as a first-order ARMA process and a white-noise sequence of variance −30 dB was used as input. The variance of the input signal and additional noise were −30 dB. The measured MSE is compared with that predicted by the derived formulas for several values of parameter $q$. Parameter $\alpha$ was chosen such that time convergence of matrix $\hat{\mathbf{R}}^{-1}(n)$ could be achieved. Table 2.II illustrates the results obtained.

TABLE 2.II

$J_{ex}$ IN NONSTATIONARY ENVIRONMENT, dB

| $q$ | Misadjustment | | Lag | | Total | |
|---|---|---|---|---|---|---|
| | Eq. (2.18) | Simulations | Eq. (2.19) | Simulations | Eq. (2.20) | Simulations |
| 0.5 | −34.8 | −34.4 | −27.6 | −28.1 | −26.9 | −27.2 |
| 0.6 | −33.7 | −33.4 | −28.1 | −28.7 | −27.1 | −27.4 |
| 0.7 | −32.7 | −32.3 | −28.5 | −29.5 | −27.1 | −27.6 |
| 0.8 | −31.8 | −31.4 | −28.7 | −29.2 | −27.0 | −27.2 |
| 0.9 | −30.9 | −30.6 | −28.8 | - 29.8 | −26.7 | −27.1 |
| 1.0 | −30.0 | −29.7 | −28.9 | −29.8 | −26.4 | −26.8 |
| 1.1 | −29.1 | −28.8 | −28.8 | −30.0 | −26.0 | −26.4 |

Table 2.III shows the results obtained for a fixed-point implementation for different values of wordlength and $q = 1$. The unknown system in this experiment was

time-invariant of order 20. As can be noted, the simulation results agree with the theoretical results as long as a sufficient wordlength is provided to prevent the coefficients from freezing. Table 2.IV shows the results obtained for different values of parameter $q$ with a constant wordlength of 9 bits. In order to avoid overflow and to simplify the internal scaling, the input-signal variance was made 0 dB and 3 bits were used to represent the integer part in internal registers in all the finite-precision simulations. The variance of the additional noise used in the simulations shown in Tables 2.III and 2.IV is $-20$ dB. Using this scheme, only the register storing $\tau(n)$ had to be treated separately; its content was shifted 3 bits to the right, i.e., $N = 8$. Tables 2.II and 2.IV also show very little sensitivity of the algorithms with respect to parameter $q$.

TABLE 2.III

$J_{ex}$ DUE TO FINITE WORDLENGTH FOR $q = 1$, dB

| Number of Bits | Eq. (2.28) | Simulations |
|---|---|---|
| 6 | −20.1 | −21.9 |
| 7 | −26.2 | −27.1 |
| 8 | −32.2 | −32.8 |
| 9 | −38.2 | −38.6 |
| 12 | −56.3 | −56.5 |

Matrix $\hat{\mathbf{R}}^{-1}(n)_Q$ was evaluated in double-precision floating-point arithmetic and then quantized. This approach prevents divergence. We implicitly assumed that some strategy will be used in practice to guarantee the stability of $\hat{\mathbf{R}}^{-1}(n)_Q$, as discussed previously. One such strategy is the implementation of the algorithm in its QR-decomposition form. The next experiment was designed to illustrate the increased robustness of the variable-convergence-factor QR-LMSN algorithm as compared to its conventional form.

In the last experiment, the variable-convergence-factor QR-LMSN algorithm was

TABLE 2.IV

$J_{ex}$ DUE TO FINITE WORDLENGTH FOR 9 BITS, dB

| $q$ | Eq. (2.28) | Simulations |
|---|---|---|
| 0.5 | −37.0 | −37.9 |
| 0.6 | −37.5 | −38.4 |
| 0.7 | −37.9 | −38.6 |
| 0.8 | −38.1 | −38.7 |
| 0.9 | −38.2 | −38.9 |
| 1.0 | −38.2 | −38.9 |
| 1.1 | −38.1 | −38.8 |
| 1.2 | −38.0 | −38.5 |
| 1.3 | −37.7 | −38.3 |
| 1.4 | −37.3 | −38.0 |
| 1.5 | −36.8 | −37.4 |

used to identify a 31st-order plant when 16-bit fixed-point arithmetic was used. The input-signal variance was −5 dB and the measurement noise variance was −40 dB. The coefficients of the plant were normalized such that the total power was equal to 1. Parameter $\alpha$ was chosen equal to 0.1 and $\gamma(n)$ was scaled down by 2, i.e., shifted by one bit. All other quantities were assumed smaller than 1, and eventual overflows were treated by saturation. Inner products were quantized after all additions were performed. Division by zero was detected prior to occurrence during the calculation of the cosines, sines, and $\gamma(n)$. In the case of the cosines and sines the denominator was calculated again but with $x_k$ and $(1-\alpha)\mathbf{R}_{kk}(n-1)$ shifted up by 8 bits. If the denominator was still equal to zero, then the cosine was forced equal to zero and the sine equal to 1. This causes $\gamma(n) = 1$. The division by zero during the calculation of $\gamma(n)$ can only happen when the output error is zero, since it is expected that the product of the cosines will never be equal to one, specially after being quantized. In

the case when $e(n) = 0$, neither the coefficients of the filter nor $\mathbf{R}(n)$ and $\mathbf{p}(n)$ were adapted. Figure 2.3 shows the smoothed learning-curve for the first 20000 iterations of this experiment. Much longer simulations were run without any signs of instability.

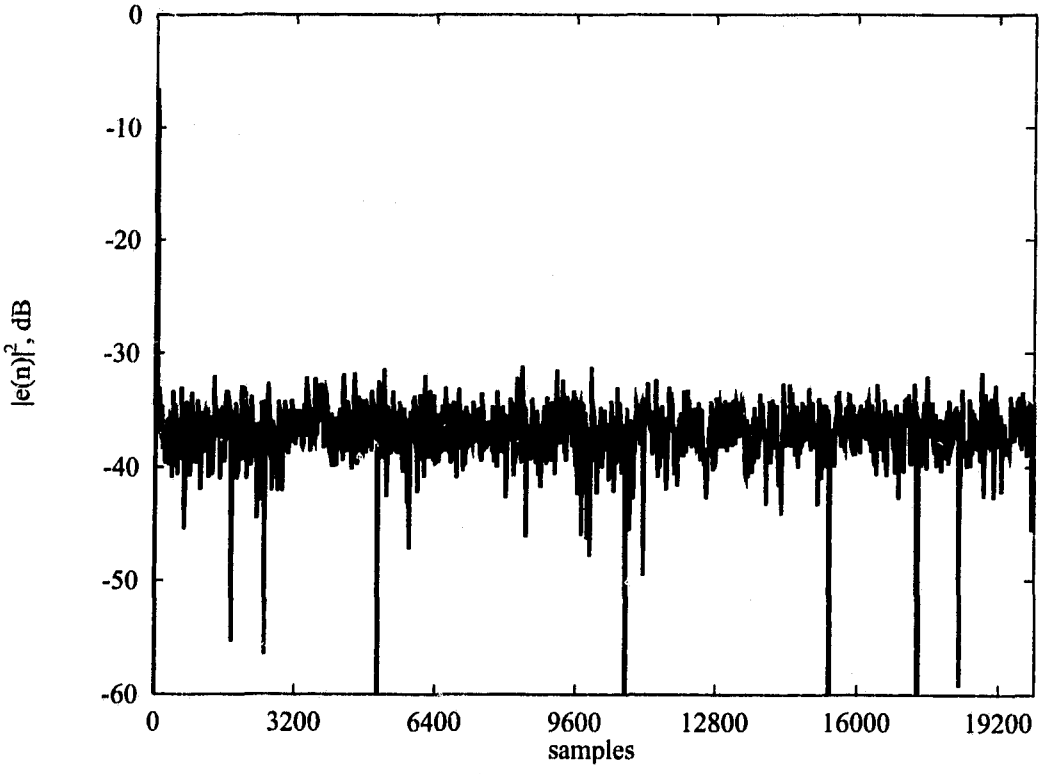



Figure 2.3: Learning curve for the variable-convergence-factor QR-LMSN algorithm for fixed-point arithmetic.

## 2.7 Conclusions

As knowledge of the environment where the adaptive filter will be operating is sometimes restricted, variable-convergence-factor algorithms are certainly an interesting choice. The two algorithms described and analyzed in this chapter are versions of the orthogonalized-projection algorithm that employ different estimates of the input-signal autocorrelation matrix, $\hat{\mathbf{R}}(n)$. Algorithm I uses a variable convergence factor when estimating $\mathbf{R}$, while algorithm II uses a constant convergence factor. Extensive simulations have shown that the two algorithms can perform in most situations as

well as the RLS algorithm without requiring as much knowledge of the signal statistics. The sensitivity of these algorithms with respect to their parameters is greatly reduced by the introduction of this particular convergence factor. The algorithms proved themselves especially suited for applications in nonstationary environments where algorithms with fixed convergence factors can very easily fail.

The analysis of the algorithms would not be complete without an in-depth investigation of the effects of quantization on the output error and stability. The formula for the excess MSE due to quantization suggests that even with fixed-point arithmetic, its influence on the overall MSE is not significant. On the other hand, the effects of quantization on the estimate of $\mathbf{R}$ for both algorithms is a threat to their stability, much in the same way as with the estimate of $\mathbf{R}$ employed by the RLS algorithm. A good solution to this problem is the implementation of the algorithm via the QR decomposition. Since $\hat{\mathbf{R}}^{-1}(n)$ is not explicitly computed in this case, the variable convergence factor had to be incorporated into the algorithm in a different way, but still maintaining the robustness and zero a posteriori output error. However, the complexity of QR-decomposition algorithms in general and the fact that for non-persistently exciting signals all algorithms discussed previously are likely to diverge call for a more definite solution. The development of the quasi-Newton algorithm presented in the next chapter is a step towards this direction.

# Chapter 3

# Quasi-Newton Algorithm

## 3.1 Introduction

Though an accredited tool for numerous applications ranging from control of processes to telephony, adaptive FIR filters still challenge engineering designers with regards to implementation. Among the different properties of the algorithms, robustness when short wordlength is used is certainly of the greatest importance. For this reason, understanding the effects of quantization on the performance of the algorithms is paramount for consolidating their use in practical applications. Such a study must take into account situations where the input signal is highly correlated or even non-persistently exciting. Simple algorithms, such as the LMS algorithm, are robust in most situations, but their convergence is often slow and, therefore, not satisfactory. Conversely, the lack of robustness of fast-converging algorithms, such as the RLS and Newton-type algorithms, have undoubtedly contributed to hinder their application in practice. Moreover, alternative implementations, such as those based on the QR decomposition, are highly complex and may require careful attention with respect to the scaling of internal variables in order to avoid constant overflow or division by zero in fixed-point arithmetic.

The theoretical finite-precision error analysis of adaptation algorithms relies on three basic aspects: the identification of single errors introduced by quantization, the propagation of the errors in time, and the interaction and accumulation of these

errors. Due to the nonlinearity and complexity of the equations involved, most of the analyses carried out so far focus on specific properties of the algorithms and often overlooked important issues regarding stability. For example, it is known that when the propagation of a single error is considered, the RLS algorithm is stable provided that symmetry is preserved when updating the estimate of the input-signal autocorrelation matrix, $\hat{\mathbf{R}}^{-1}(n)$, [29] [66] [68]. However, the fundamental assumption made when arriving at this conclusion is that $\hat{\mathbf{R}}^{-1}(n)$ remains bounded and positive definite. It was demonstrated by Bottomley and Alexander [31] through a comprehensive study of the behavior of the RLS algorithm in fixed-point arithmetic that this might not be a valid assumption after all, for interaction and accumulation of quantization errors can make $\hat{\mathbf{R}}^{-1}(n)$ indefinite. This was found to be true for the LMSN algorithm analyzed in the previous chapter as well. Several schemes to stabilize an algorithm have been proposed for both fixed-point and floating-point arithmetic, e.g., leakage and the QR decomposition. Notwithstanding the importance of the recent contributions in the area, much can undoubtedly be gained if the algorithm is inherently more robust and does not need stabilization countermeasures. It would be highly desirable if stability were guaranteed for this algorithm when low-precision arithmetic is used and highly correlated or even nonpersistently exciting signals are present.

This chapter is devoted to the development, analysis, and testing of a new quasi-Newton (QN) adaptive filtering algorithm [41] [43]. The results show that this algorithm can be successfully applied in several situations where the RLS algorithm, or even the QR-RLS algorithm, can easily fail while achieving a similar convergence speed. The development of the algorithm is based on the standard rank-one quasi-Newton approach, the concept of a posteriori output error, and an exact line search. The result is an algorithm that requires no parameter tuning. Theorems demonstrating the robustness of the algorithm and extensive simulation results and comparisons with the RLS algorithm in different situations are provided [42].

## 3.2 New QN Algorithm

Although adaptive filtering algorithms are, in general, restricted to relatively simple and often rudimentary schemes that operate on-line and entail a minimum amount of computation per output sample, comparisons with off-line optimization methods have often been made. For instance, the slow convergence of the LMS algorithm has always been attributed to the steepest-descent direction used, whereas fast-converging algorithms are associated with Newton-type algorithms. Despite the parallel established, it is surprising that no attempt has been made so far to port well known quasi-Newton optimization schemes to the adaptive filtering platform. Perhaps the reason relates to the widespread use of the LMS algorithm, or to the fact that Newton-type algorithms are computational demanding and usually not reliable. However, as applications become more demanding, and speeds and throughputs are increasing due to the rapid progress in the design of VLSI hardware, considerable attention has recently been devoted to more sophisticated algorithms. One of these algorithms is the QN algorithm which, in its most basic form, is as follows [69]:

**Step 1** Compute the direction to update the coefficient vector $\mathbf{w}(n-1)$ as

$$\mathbf{h}(n) = -\hat{\mathbf{R}}^{-1}(n-1)\mathbf{g}(n-1)$$

where $\mathbf{g}(n-1)$ denotes the gradient evaluated at point $\mathbf{w}(n-1)$ and $\hat{\mathbf{R}}(n-1)$ is an estimate of the input-signal autocorrelation matrix.

**Step 2** Perform a line search along $\mathbf{h}(n)$ and evaluate the convergence factor (or step size), $\mu(n)$, and compute

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu(n)\mathbf{h}(n)$$

**Step 3** Update $\hat{\mathbf{R}}^{-1}(n-1)$ and obtain $\hat{\mathbf{R}}^{-1}(n)$ as a better estimate of the input-signal autocorrelation matrix.

Initialization of $\hat{\mathbf{R}}^{-1}(n)$ as a positive definite matrix is necessary to ensure that the change in $\mathbf{w}(n)$ is always a descent direction. Usually $\hat{\mathbf{R}}^{-1}(0) = \mathbf{I}$ is selected, where $\mathbf{I}$ is the identity matrix.

Unlike the usual optimization problem where an objective function is given, in the case of adaptive filtering the selection of the objective function will determine the algorithm complexity and, therefore, its usefulness. For this reason it is customary to choose a function which is as simple as possible and hope that the algorithm will average out eventual errors. A usual choice is the instantaneous a priori output error defined as

$$\xi(n) = e^2(n)$$

where $e(n)$ is the output a priori error defined as in (1.7), and the gradient is calculated as

$$\mathbf{g}(n-1) = -2e(n)\mathbf{u}(n) \tag{3.1}$$

Let us examine steps 2 and 3, starting with step 3. According to [69], one way of updating $\hat{\mathbf{R}}^{-1}(n-1)$ in step 3 such that an arbitrarily chosen $\hat{\mathbf{R}}^{-1}(0)$ is changed into an approximation of the inverse of the exact input-signal autocorrelation matrix, $\mathbf{R}^{-1}$, is

$$\hat{\mathbf{R}}^{-1}(n) = \hat{\mathbf{R}}^{-1}(n-1) + \frac{\left[\boldsymbol{\delta}(n) - \hat{\mathbf{R}}^{-1}(n-1)\boldsymbol{\gamma}(n)\right]\left[\boldsymbol{\delta}(n) - \hat{\mathbf{R}}^{-1}(n-1)\boldsymbol{\gamma}(n)\right]^T}{\left[\boldsymbol{\delta}(n) - \hat{\mathbf{R}}^{-1}(n-1)\boldsymbol{\gamma}(n)\right]^T \boldsymbol{\gamma}(n)} \tag{3.2}$$

where[1]

$$\boldsymbol{\delta}(n) = \mathbf{w}(n) - \mathbf{w}(n-1)$$

and

$$\boldsymbol{\gamma}(n) = \mathbf{g}'(n) - \mathbf{g}(n-1) \tag{3.3}$$

Notice that in (3.3), $\mathbf{g}'(n)$ was used rather than $\mathbf{g}(n)$ as in [69]. In the context of adaptive filters, $\mathbf{g}(n)$ as defined in (3.1) is not available during iteration $n$, since it

[1]For the sake of consistency with the references, $\gamma(n)$ was used in (3.2) although it bears no relation with $\gamma(n) = 2\mu(n)/\alpha$ in the variable-convergence-factor QR-LMSN algorithm discussed in the previous chapter.

would require $\mathbf{u}(n+1)$ and $d(n+1)$. Instead, $\mathbf{g}'(n)$ was used, which is the gradient calculated at point $\mathbf{w}(n)$, but using $\mathbf{u}(n)$ and $d(n)$. Vector $\mathbf{g}'(n)$ is defined as

$$\mathbf{g}'(n) = -2\epsilon(n)\mathbf{u}(n)$$

where $\epsilon(n)$ is the output a posteriori error defined in (1.18). Independently of how the gradients are calculated, (3.2) assures that the quasi-Newton condition, $\hat{\mathbf{R}}^{-1}(n)\boldsymbol{\gamma}(n) = \boldsymbol{\delta}(n)$, is satisfied. This condition distinguishes the QN algorithm from any other Newton-type algorithm that employs an estimate of matrix $\mathbf{R}$, e.g., the LMS-Newton (LMSN) algorithm or the RLS algorithm.

Now let us examine step 2. As discussed in [25] and [37], an exact line search can be performed in the direction given by $\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$. This is accomplished by setting the gradient of $\epsilon^2(n)$ with respect to $\mu(n)$ to zero, which yields

$$\mu(n) = \frac{1}{2\tau(n)} \tag{3.4}$$

where

$$\tau(n) = \mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$$

Notice that $\mu(n)$ in (3.4) is the optimum convergence factor in the direction of $\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$, not $\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n)$ as in (2.8). However, both convergence factors are optimum in the sense that they yield zero a posteriori error, i.e., $\epsilon(n) = 0$. It is assumed hereafter that the input signal is never equal to zero, such that if $\hat{\mathbf{R}}^{-1}(n)$ is positive definite for all $n \geq 0$, then $\tau(n) > 0$. After some manipulation, it is easy to verify that (3.2) simplifies to

$$\hat{\mathbf{R}}^{-1}(n) = \hat{\mathbf{R}}^{-1}(n-1) + [\mu(n) - 1]\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)} \tag{3.5}$$

The adaptive filtering QN algorithm is summarized as follows: Given $\mathbf{u}(n)$ and $d(n)$ compute

$$e(n) = d(n) - \mathbf{w}^T(n-1)\mathbf{u}(n)$$

$$\mathbf{t}(n) = \hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$$

$$\tau(n) = \mathbf{u}^T(n)\mathbf{t}(n)$$

$$\mu(n) = \frac{1}{2\tau(n)}$$

$$\hat{\mathbf{R}}^{-1}(n) = \hat{\mathbf{R}}^{-1}(n-1) + \frac{[\mu(n)-1]}{\tau(n)}\mathbf{t}(n)\mathbf{t}^T(n)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \frac{e(n)}{\tau(n)}\mathbf{t}(n) \tag{3.6}$$

An interesting point to note is that for this particular definition of $\boldsymbol{\delta}(n)$ and $\boldsymbol{\gamma}(n)$, the Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) formulas [69] can also be adapted to fit the adaptive filtering context, but the resulting update formulas for $\hat{\mathbf{R}}^{-1}(n)$ degenerate in both cases to (3.5). There is no doubt that these updating schemes lose much in the translation from the optimization scenario to the adaptive filtering scenario, where the gradient used is a poor approximation based on instantaneous measurements. However, as will be shown in the following sections, the fact that these equations were initially derived to yield good estimates of the true autocorrelation matrix and also to keep $\hat{\mathbf{R}}^{-1}(n)$ positive definite [70], has a strong effect upon the stability of the newly derived algorithm.

The algorithm outlined above is very general and obviously other choices for calculating the gradient could be made; furthermore, a nonexact line search could be employed. For instance, one can argue that by choosing $\boldsymbol{\gamma}(n)$ and $\boldsymbol{\delta}(n)$ differently, a rank-2 formula for updating $\hat{\mathbf{R}}^{-1}(n)$ could be obtained. Although indisputably true, it is not at all clear at this point if such a scheme is necessary or even advantageous. In the next section the QN algorithm will be analyzed and some of its most important properties will be outlined.

## 3.3 Analysis of QN Algorithm

As a first step towards the analysis of the algorithm, it is illustrative to derive the deterministic objective function minimized. From (3.5) and the matrix-inversion

lemma [8], it is easy to verify that

$$\hat{\mathbf{R}}(n) = \hat{\mathbf{R}}(n-1) + 2\left[1-\mu(n)\right]\mathbf{u}(n)\mathbf{u}^T(n) \tag{3.7}$$

which shows that $\hat{\mathbf{R}}(n)$ is a weighted average of the outer product $\mathbf{u}(n)\mathbf{u}^T(n)$ that uses time-dependent weights which can assume positive as well as negative values. However, the discussions that will follow will show that these weights will never drive $\hat{\mathbf{R}}(n)$ outside the bounded positive definite region.

Premultiplying (3.6) by $\hat{\mathbf{R}}(n)$, a system of equations of the form of (1.20) results, except that here matrix $\hat{\mathbf{R}}(n)$ and vector $\hat{\mathbf{p}}(n)$ are defined as in (3.7) and

$$\hat{\mathbf{p}}(n) = \hat{\mathbf{p}}(n-1) + 2\left[d(n) - \mu(n)\mathbf{u}^T(n)\mathbf{w}(n-1)\right]\mathbf{u}(n)$$

respectively. The solution $\mathbf{w}(n) = \hat{\mathbf{R}}^{-1}(n)\hat{\mathbf{p}}(n)$, obtained recursively by the QN algorithm, minimizes

$$\mathcal{E}(n) = \sum_{i=1}^{n}[1-\mu(i)]\left[\frac{d(i)-\mu(i)\mathbf{u}^T(i)\mathbf{w}(i-1)}{1-\mu(i)} - \mathbf{u}^T(i)\mathbf{w}(n)\right]^2$$

$$+\frac{1}{2}\mathbf{w}^T(n)\hat{\mathbf{R}}(0)\mathbf{w}(n)$$

Other Newton-type algorithms share similar deterministic objective functions [24] [40], e.g., the LMSN algorithms discussed in the previous chapter. The complicated objective function of the QN algorithm bears very little, if any, physical interpretation, which makes it more necessary to show the conditions for convergence and unbiasedness of the solution. We will start by demonstrating the positive definiteness of matrix $\hat{\mathbf{R}}^{-1}(n)$, which is a key assumption for the rest of the analysis as well as one of the most important properties of the algorithm presented.

### 3.3.1 Positive Definiteness of $\hat{\mathbf{R}}^{-1}(n)$

The stability of the algorithm relies on the positive definiteness of the estimate of the autocorrelation matrix, $\hat{\mathbf{R}}^{-1}(n)$. As was previously discussed, this condition can be violated when $\hat{\mathbf{R}}^{-1}(n)$ is estimated as an exponentially weighted average of $\mathbf{u}(n)\mathbf{u}^T(n)$ as, e.g., in the RLS and other Newton-type algorithms. However, the QN algorithm

just derived utilizes an estimated $\hat{\mathbf{R}}^{-1}(n)$ which can remain positive definite even for a nonpersistently exciting input signal. The forthcoming analysis will show that if the initial $\hat{\mathbf{R}}^{-1}(n)$ is positive definite, the positive definiteness of $\hat{\mathbf{R}}^{-1}(n)$ will be maintained indefinitely. For this purpose we need two lemmas and a corollary as follows:

**Lemma 1.** *Let*

$$\mathbf{A} = \begin{bmatrix} \alpha & \mathbf{a}^T \\ \mathbf{a} & \text{diag}(\alpha_i) \end{bmatrix}, \quad i = 1, \ldots; M-1$$

*have eigenvalues denoted by $\lambda_i$ arranged such that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_M$. The eigenvalues of the matrix* $\text{diag}(\alpha_i)$ *separate the eigenvalues of* $\mathbf{A}$*, at least, in the weak sense, i.e., $\lambda_1 \leq \alpha_1 \leq \lambda_2 \leq \alpha_2 \leq \cdots \leq \alpha_{M-1} \leq \lambda_M$.*

**Lemma 2.** *Let $\mathbf{C} = \mathbf{A} + \mathbf{B}$ with $\mathbf{A}$ and $\mathbf{B}$ symmetric, and let $\mathbf{B}$ be of rank unity with a nonzero eigenvalue equal to $\rho$. If the eigenvalues of $\mathbf{A}$, say $\lambda_i$, are those of*

$$\mathbf{A}_1 = \begin{bmatrix} \alpha & \mathbf{a}^T \\ \mathbf{a} & \text{diag}(\alpha_i) \end{bmatrix}, \quad i = 1, \ldots, M-1$$

*then the eigenvalues of $\mathbf{C}$, say $\lambda_i'$, are those of*

$$\mathbf{C}_1 = \begin{bmatrix} \alpha + \rho & \mathbf{a}^T \\ \mathbf{a} & \text{diag}(\alpha_i) \end{bmatrix}, \quad i = 1, \ldots, M-1$$

*and we have*

$$\lambda_i' - \lambda_i = m_i \rho$$

*with*

$$0 \leq m_i \leq 1$$

*and*

$$\sum_{i=1}^{M} m_i = 1$$

The proofs of the above lemmas can be found in [73], pp. 94-98.

A corollary can now be readily obtained from lemmas 1 and 2.

**Corollary 1.** *If* $\mathbf{A}$ *and* $\mathbf{C}$ *are matrices with corresponding eigenvalues equal to* $\lambda_i$ *and* $\lambda'_i$*, and if* $\mathbf{B}$ *is a rank-one matrix with nonzero eigenvalue equal to* $\rho$*, then* $\mathbf{C} = \mathbf{A} + \mathbf{B}$ *implies*

$$\lambda_1 \leq \lambda'_1 \leq \lambda_2 \leq \lambda'_2 \leq \cdots \leq \lambda_M \leq \lambda'_M \quad \textit{for } \rho > 0,$$

$$\lambda'_1 \leq \lambda_1 \leq \lambda'_2 \leq \lambda_2 \leq \cdots \leq \lambda'_M \leq \lambda_M \quad \textit{for } \rho < 0$$

*Proof.* From lemma 1

$$\lambda_1 \leq \alpha_1 \leq \lambda_2 \leq \alpha_2 \leq \cdots \leq \alpha_M \leq \lambda_M$$

$$\lambda'_1 \leq \alpha_1 \leq \lambda'_2 \leq \alpha_2 \leq \cdots \leq \alpha_M \leq \lambda'_M$$

Therefore,

$$\lambda_1 \leq \lambda'_2$$

$$\lambda_2 \leq \lambda'_3$$

$$\vdots$$

$$\lambda_{M-1} \leq \lambda'_M$$

From lemma 2

$$\lambda'_i \geq \lambda_i \quad \text{for } \rho > 0$$

$$\lambda_i \geq \lambda'_i \quad \text{for } \rho < 0$$

which concludes the proof. ■

The theorem demonstrating the positive definiteness of $\hat{\mathbf{R}}^{-1}(n)$ reads as follows:

**Theorem 1.** *Let* $\hat{\mathbf{R}}^{-1}(n)$*, the estimated inverse of the true autocorrelation matrix of order* $M$*, be updated according to* (3.5)*. If* $\hat{\mathbf{R}}^{-1}(n-1)$ *is symmetric positive definite, then* $\hat{\mathbf{R}}^{-1}(n)$ *will also be symmetric and positive definite for all* $n > 0$.

*Proof.* Let $\hat{\mathbf{R}}_1^{-1}$ be an auxiliary matrix defined as

$$\hat{\mathbf{R}}_1^{-1} = \hat{\mathbf{R}}^{-1}(n-1) - \frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}$$

Since $\hat{\mathbf{R}}^{-1}(n-1)$ is positive definite then the smallest eigenvalue of $\hat{\mathbf{R}}_1^{-1}$ is unique, equal to 0, and is associated with eigenvector $\mathbf{u}(n)$. This can be verified by posmultiplying $\hat{\mathbf{R}}_1^{-1}$ by $\mathbf{u}(n)$, which gives $\hat{\mathbf{R}}_1^{-1}\mathbf{u}(n) = \mathbf{0}$. The uniqueness of the zero eigenvalue is due to the fact that if any other eigenvalue of $\hat{\mathbf{R}}_1^{-1}$ is also zero, according to corollary 1, matrix $\hat{\mathbf{R}}^{-1}(n-1)$ would necessarily have at least one eigenvalue equal to 0, which contradicts the initial assumption that $\hat{\mathbf{R}}^{-1}(n-1)$ is positive definite. Now, since $\hat{\mathbf{R}}^{-1}(n)$ is given by

$$\hat{\mathbf{R}}^{-1}(n) = \hat{\mathbf{R}}_1^{-1} + \mu(n)\frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}$$

by once again invoking corollary 1 it is easy to verify that the smallest eigenvalue of $\hat{\mathbf{R}}^{-1}(n)$ must be greater than or equal to zero; the equality being verified if and only if $\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$ is orthogonal to the null space of $\hat{\mathbf{R}}_1^{-1}$. However, if this is true, as $\mathbf{u}(n)$ belongs to the null space of $\hat{\mathbf{R}}_1^{-1}$, then $\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n) = 0$, which once again contradicts the assumption that $\hat{\mathbf{R}}^{-1}(n-1)$ is positive definite. ■

By induction theorem 1 can be extended up from $n = 1$, which proves that if $\hat{\mathbf{R}}^{-1}(0) > 0$, then $\hat{\mathbf{R}}^{-1}(n) > 0$ for all $n > 0$.

From (3.7) it is not difficult to perceive that the influence of the initial state of $\hat{\mathbf{R}}^{-1}(n)$ throughout time is enduring, as opposed to the more temporary effect observed in the RLS algorithm, for example. This is certainly part of the reason why the algorithm can operate safely for highly correlated input signals. Even in the event where the input-signal vector does not excite all the $M$ possible directions, the presence of the initial state assures that $\hat{\mathbf{R}}^{-1}(n)$ is not singular.

Below we show that the boundedness of $\hat{\mathbf{R}}^{-1}(n)$ is also automatically guaranteed.

**Theorem 2.** *The estimated inverse of the autocorrelation matrix, $\hat{\mathbf{R}}^{-1}(n)$, given by* (3.5) *is always bounded, provided that the input signal is bounded.*

*Proof.* If we premultiply (3.5) by $\mathbf{u}^T(n)$ and posmultiply it by $\mathbf{u}(n)$, we have

$$\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n) = \tfrac{1}{2} \tag{3.8}$$

Therefore, at least for persistently exciting input signals, (3.8) guarantees that $\hat{\mathbf{R}}^{-1}(n)$ is bounded. For a nonpersistently exciting input signal, the only potentially hazardous situation would arise if $\hat{\mathbf{R}}^{-1}(n)$ grows large without bound in a direction which does not belong to the set $\{\mathbf{u}(n),\ n = 1,\ 2,\ \ldots\}$. However, a quick look at (3.5) shows that contributions to $\hat{\mathbf{R}}^{-1}(n)$ are of the form of outer products of vectors $\hat{\mathbf{R}}^{-1}(i-1)\mathbf{u}(i)$, $i \in \{1,\ \ldots,\ n\}$, which are never orthogonal to $\mathbf{u}(i) \neq 0$ because $\hat{\mathbf{R}}^{-1}(i-1)$ is guaranteed to be positive definite. ■

### 3.3.2 Stability

Some previous analyses of adaptation algorithms have established the convergence of the coefficients to their optimal values in the mean [8]. In general, these analyses concentrate on a system of equations describing the average dynamics of the coefficient-error vector, and proving that this system is asymptotically stable. In other words, for the vector $\mathbf{v}(n)$ defined as

$$\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_o$$

a difference equation of the form

$$E[\mathbf{v}(n)] \approx E[\mathbf{A}(n)]E[\mathbf{v}(n-1)] \tag{3.9}$$

can be obtained. Usually, global asymptotic convergence of the parameters is guaranteed if and only if $E[\mathbf{A}(n)]$ is time-invariant and its eigenvalues are strictly inside the unity circle. However, for projection algorithms strong independence assumptions have to be made to support a linear time-invariant description of the system such as in (3.9). This difficulty can be circumvented by stating the conditions for convergence with probability 1 of the system describing $\mathbf{v}(n)$, not the one describing $E[\mathbf{v}(n)]$. Besides, this approach allows us to demonstrate the stability of the algorithm even when the input signal is not persistently exciting. Similar criteria have

been used to demonstrate the stability of other projection algorithms, such as the normalized LMS algorithm [74] and the orthogonalized projection algorithm [24].

Below a number of properties of the QN algorithm that relate to stability are examined. First, let us assume that the optimal solution to the estimation problem at hand is such that the desired signal, $d(n)$, can be exactly modeled as

$$d(n) = \mathbf{w}_o^T \mathbf{u}(n)$$

where $\mathbf{w}_o$ is a vector whose $M$ elements are the optimal coefficients. For this model, an autonomous representation can be easily obtained for $\mathbf{v}(n)$ as

$$\begin{aligned}\mathbf{v}(n) &= \left[\mathbf{I} - \frac{\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)}{\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)}\right]\mathbf{v}(n-1) \\ &= \mathbf{T}(n)\mathbf{v}(n-1) \end{aligned} \tag{3.10}$$

**Property 1.** *The state at iteration $n$, denoted as $\mathbf{v}(n)$, is orthogonal to the input signal vector, i.e., $\mathbf{u}(n) \perp \mathbf{v}(n)$.*

**Property 2.** *If $\hat{\mathbf{R}}^{-1}(n-1)$ is a positive definite matrix, then the states remain unaltered from iteration $(n-1)$ to iteration $n$, i.e.,*

$$\mathbf{v}(n) = \mathbf{T}(n)\mathbf{v}(n-1) = \mathbf{v}(n-1)$$

*if and only if $\mathbf{v}(n-1)$ and $\mathbf{u}(n)$ are orthogonal.*

**Property 3.** *If the state $\mathbf{v}(n-1)$ is proportional to the vector $\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$, then the Euclidean norm of the state at instant $n$ is zero, i.e., if $\mathbf{v}(n-1) \sim \hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$ then, $\|\mathbf{v}(n)\|_2^2 = 0$.*

The proofs of the above properties follow from (3.10).

From property 3, we can verify that matrix $\mathbf{T}(n)$ has $(M-1)$ eigenvalues equal to 1, and one eigenvalue equal to 0 associated with eigenvector $\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)$. The eigenvectors associated with the $M-1$ eigenvalues equal to 1 lie in the $(M-1)$-dimensional subspace which is orthogonal to vector $\mathbf{u}(n)$. Although these interesting properties of $\mathbf{T}(n)$ are known, it is difficult to show that $\|\mathbf{T}(n)\|_2^2 \leq 1$ or, equivalently,

that $\|\mathbf{v}(n)\|_2^2 \leq \|\mathbf{v}(n-1)\|_2^2$, which would guarantee the stability of the linear time-dependent system (3.10). Nonetheless, it is possible to guarantee the stability of (3.10) and, consequently, of the QN algorithm based on the stability of an equivalent system. For this purpose we need to extend to discrete systems a very important result presented by Lyapunov for linear continuous time-dependent systems [75] [76]. The proof is trivial and is omitted.

**Lemma 3 (Lyapunov).** *Let the state-space realization of a linear time-dependent system $\chi$ be described by*

$$\mathbf{x}(n+1) = \mathbf{A}(n)\mathbf{x}(n) + \mathbf{B}(n)\mathbf{u}(n)$$

$$y(n) = \mathbf{C}(n)\mathbf{x}(n) + \mathbf{D}(n)\mathbf{u}(n)$$

*and let $\mathbf{P}(n)$ denote a Lyapunov transformation, i.e., $\mathbf{P}(n)$ satisfies*

1. *$\mathbf{P}(n)$ and $\mathbf{P}^{-1}(n)$, $n \geq n_0$, are bounded and*
2. *there exists a constant $k$ such that $0 < k < |det[\mathbf{P}(n)]|$,*

*which relates $\chi$ with an equivalent system $\bar{\chi}$ (in the sense of Lyapunov), i.e.,*

$$\bar{\mathbf{x}}(n+1) = \mathbf{P}^{-1}(n+1)\mathbf{A}(n)\mathbf{P}(n)\bar{\mathbf{x}}(n) + \mathbf{P}^{-1}(n+1)\mathbf{B}(n)\mathbf{u}(n)$$

$$y(n) = \mathbf{C}(n)\mathbf{P}(n)\bar{\mathbf{x}}(n) + \mathbf{D}(n)\mathbf{u}(n)$$

*where $\mathbf{x}(n) = \mathbf{P}(n)\bar{\mathbf{x}}(n)$. If the solution of system $\bar{\chi}$ is stable, asymptotically stable or unstable, then the solution of system $\chi$ has the same property.*

In the light of this result, we can define a new system which is equivalent (in the sense of Lyapunov) to (3.10) as

$$\bar{\mathbf{v}}(n) = \hat{\mathbf{R}}^{1/2}(n-1)\mathbf{v}(n) \tag{3.11}$$

where $\hat{\mathbf{R}}(n-1) = \hat{\mathbf{R}}^{T/2}(n-1)\hat{\mathbf{R}}^{1/2}(n-1)$, for which stability can be proved more easily, as stated in theorem 3.

**Theorem 3.** *The solution of* (3.10) *and, consequently, the QN algorithm is stable. Furthermore, if the input signal is persistently exciting, then the solution of* (3.10) *and, consequently, the QN algorithm is asymptotically stable.*

*Proof.* From theorems 1 and 2 it is easy to verify that $\hat{\mathbf{R}}^{-1}(n)$ and $\hat{\mathbf{R}}(n)$ are Lyapunov transformations. We can obtain $\hat{\mathbf{R}}^{-1/2}(n)$ and $\hat{\mathbf{R}}^{1/2}(n)$ which are also Lyapunov transformations through, for example, an $LDL^T$ decomposition. Using (3.10) and (3.11), we have

$$\begin{aligned}\|\bar{\mathbf{v}}(n)\|_2^2 &= \mathbf{v}^T(n)\hat{\mathbf{R}}^{T/2}(n-1)\hat{\mathbf{R}}^{1/2}(n-1)\mathbf{v}(n)\\ &= \mathbf{v}^T(n)\hat{\mathbf{R}}(n-1)\mathbf{v}(n)\\ &= \mathbf{v}^T(n-1)\mathbf{T}^T(n)\hat{\mathbf{R}}(n-1)\mathbf{T}(n)\mathbf{v}(n-1)\end{aligned}$$

However,

$$\begin{aligned}\mathbf{T}^T(n)\hat{\mathbf{R}}(n-1)\mathbf{T}(n) &= \left[\mathbf{I} - \frac{\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)}{\tau(n)}\right]\hat{\mathbf{R}}(n-1)\mathbf{T}(n)\\ &= \left[\hat{\mathbf{R}}(n-1) - \frac{\mathbf{u}(n)\mathbf{u}^T(n)}{\tau(n)}\right]\mathbf{T}(n)\\ &= \hat{\mathbf{R}}(n-1) - \frac{\mathbf{u}(n)\mathbf{u}^T(n)}{\tau(n)}\end{aligned}$$

Therefore,

$$\|\bar{\mathbf{v}}(n)\|_2^2 = \mathbf{v}^T(n-1)\hat{\mathbf{R}}(n-1)\mathbf{v}(n-1) - \frac{\mathbf{v}^T(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{v}(n-1)}{\tau(n)} \quad (3.12)$$

From (3.7) and (3.12) we have

$$\begin{aligned}\|\bar{\mathbf{v}}(n)\|_2^2 &= \mathbf{v}^T(n-1)\left\{\hat{\mathbf{R}}(n-2) + \left[2 - \frac{1}{\tau(n-1)}\right]\mathbf{u}(n-1)\mathbf{u}^T(n-1)\right\}\mathbf{v}(n-1)\\ &\quad - \frac{\mathbf{v}^T(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{v}(n-1)}{\tau(n)}\\ &= \mathbf{v}^T(n-1)\hat{\mathbf{R}}(n-2)\mathbf{v}(n-1)\\ &\quad + \left[2 - \frac{1}{\tau(n-1)}\right]\mathbf{v}^T(n-1)\mathbf{u}(n-1)\mathbf{u}^T(n-1)\mathbf{v}(n-1)\\ &\quad - \frac{\mathbf{v}^T(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{v}(n-1)}{\tau(n)}\end{aligned}$$

From property 1, we know that $\mathbf{v}^T(n-1)\mathbf{u}(n-1) = 0$ and, therefore,

$$\|\bar{\mathbf{v}}(n)\|_2^2 = \mathbf{v}^T(n-1)\hat{\mathbf{R}}(n-2)\mathbf{v}(n-1) - \frac{\mathbf{v}^T(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{v}(n-1)}{\tau(n)}$$

$$= \|\bar{\mathbf{v}}(n-1)\|_2^2 - \frac{\mathbf{v}^T(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{v}(n-1)}{\tau(n)} \tag{3.13}$$

Consequently,

$$\|\bar{\mathbf{v}}(n)\|_2^2 \leq \|\bar{\mathbf{v}}(n-1)\|_2^2$$

which concludes the stability part of the proof.

As for the asymptotic stability part, from (3.13) we conclude that $\|\bar{\mathbf{v}}(n)\|_2^2$ will remain constant during an interval $[n_1, n_2]$ if and only if $\mathbf{v}(n-1)$ and $\mathbf{u}(n)$ are orthogonal for all $n \in [n_1, n_2]$. However, from property 2 we know that if $\mathbf{v}(n-1)$ and $\mathbf{u}(n)$ are orthogonal, then $\mathbf{v}(n) = \mathbf{v}(n-1)$. Therefore, $\|\bar{\mathbf{v}}(n_2)\|_2^2 = \|\bar{\mathbf{v}}(n_1)\|_2^2$ if and only if $\mathbf{v}(n_1) = \mathbf{v}(n_1+1) = \cdots = \mathbf{v}(n_2) = \mathbf{v}$. However, if the input signal is persistently exciting, we can define an infinite number of sets $S_i = \{\mathbf{u}(n_{1i}), \ldots, \mathbf{u}(n_{2i})\}$ with $M \leq (n_{2i} - n_{1i}) < M'$, such that each set $S_i$ completely spans $\mathbb{R}^M$ for some finite value of $M' > 0$. This makes it impossible to have $\mathbf{v}$ orthogonal to all $\mathbf{u}(n) \in S_i$, and, as a consequence, $\|\bar{\mathbf{v}}(n_{2i})\|_2^2 < \|\bar{\mathbf{v}}(n_{1i})\|_2^2$. In addition, as the number of sets is infinite, $\|\bar{\mathbf{v}}(n)\|_2^2 \longrightarrow 0$ for $n \longrightarrow \infty$, which concludes the second part of the proof. ■

### 3.3.3 Role of $\tau(n)$ and $\hat{\mathbf{R}}^{-1}(n)$

In the previous sections, the behavior of the QN algorithm was studied without making any assumptions regarding matrix $\hat{\mathbf{R}}^{-1}(n)$, except that it is initialized with a positive definite matrix and very little has been said about the quality of $\hat{\mathbf{R}}^{-1}(n)$ as an estimator. Although for most off-line optimization problems the estimated matrix $\hat{\mathbf{R}}^{-1}(n)$ given by the QN algorithm is very close or equal to its true value $\mathbf{R}^{-1}$, the same cannot be said when porting this algorithm to the adaptive filtering platform. It turns out that robustness is gained at the price of having a poorer estimate of $\mathbf{R}^{-1}$.

Unlike the estimated autocorrelation matrix used in the RLS algorithm, which is an exponentially weighted average of outer products of $\mathbf{u}(n)\mathbf{u}^T(n)$, in the case of the

QN algorithm the weights can assume positive as well as negative values and they depend on $\hat{\mathbf{R}}^{-1}(n-1)$. For this reason, very little information can be extracted from (3.7) or (3.5) for a general input signal.

Parameter $\tau(n)$ is of significant importance to the QN algorithm since it provides the means to automatically regulate both the estimates of $\mathbf{w}$ and $\mathbf{R}^{-1}$. In (3.4) the convergence factor, $\mu(n)$, uses $\tau(n)$ as a normalization factor to yield zero a posteriori error. Furthermore, $\tau(n)$ is a good indicator of the positive definiteness of $\hat{\mathbf{R}}^{-1}(n)$ [31], as it is likely to assume a negative value when $\hat{\mathbf{R}}^{-1}(n)$ is no longer positive definite. This section discusses how important this parameter is to the QN algorithm through a more qualitative analysis.

Although theorem 1 suffices to ascertain the positive definiteness of $\hat{\mathbf{R}}^{-1}(n)$ for the QN algorithm, a simple comparison between (3.5) and (1.26) provides a heuristic but illustrative additional argument. Let us define $\phi$ as an auxiliary variable representing the scalar gain applied to the rank-one matrices that will be added to $\hat{\mathbf{R}}^{-1}(n-1)$ in order to form the updated $\hat{\mathbf{R}}^{-1}(n)$ for both the RLS and QN algorithms. In other words, let

$$\phi_{RLS} = -\frac{1}{\lambda + \tau(n)}$$

$$\phi_{QN} = \frac{1 - 2\tau(n)}{2\tau^2(n)}$$

where

$$\hat{\mathbf{R}}^{-1}(n) = \frac{1}{\lambda}\left\{\hat{\mathbf{R}}^{-1}(n-1) + \phi_{RLS}\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)\right\}$$

and

$$\hat{\mathbf{R}}^{-1}(n) = \hat{\mathbf{R}}^{-1}(n-1) + \phi_{QN}\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\hat{\mathbf{R}}^{-1}(n-1)$$

for the RLS and QN algorithms, respectively. Assuming that $\tau(n)$ is in fact a good indicator of the positive definiteness of $\hat{\mathbf{R}}^{-1}(n)$, a plot of $\phi \times \tau(n)$ would show how the two algorithms behave as $\tau(n) \longrightarrow 0$. Fig. 3.1 shows that the RLS algorithm will continuously subtract a rank-one matrix from $\hat{\mathbf{R}}^{-1}(n-1)$, even for arbitrarily small

positive values of $\tau(n)$, as it attempts to estimate $\mathbf{R}^{-1}$. A similar conclusion would have been obtained if the estimate of $\mathbf{R}^{-1}$ used by the LMSN algorithm were used instead. On the other hand, the QN algorithm will at some point detect that $\tau(n)$ is small and will start adding rank-one matrices, which, according to corollary 1, can improve the conditioning of $\hat{\mathbf{R}}^{-1}(n)$. In other words, the QN algorithm uses $\tau(n)$ as an indicator of the conditioning of $\hat{\mathbf{R}}^{-1}(n)$ to preserve its positive definiteness.

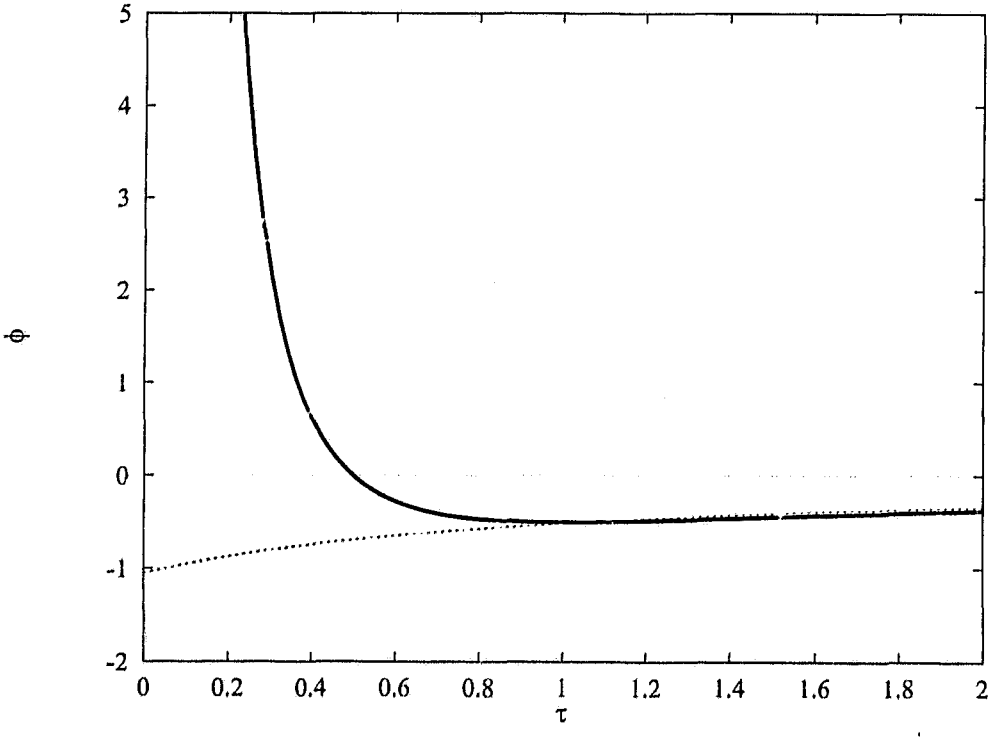



Figure 3.1: Evolution of $\phi_{RLS}$, $\phi_{QN} \times \tau$.
$\phi_{QN}$ : ——
$\phi_{RLS}$ for $\lambda = 0.95$ : - - -

In order to further study the behavior and evolution of matrix $\hat{\mathbf{R}}^{-1}(n)$ and parameter $\tau(n)$, we propose a very simplified model to represent the input signal. A similar model has been employed by Kubin [77] [78] and by Slock [19] when analyzing the normalized LMS algorithm, and excellent results have been obtained. It provides an easy way to gain insight into the properties of the algorithm which would otherwise be hidden in complex nonlinear equations. Let us suppose that the input-signal vectors

lie on only $\bar{M}$ orthogonal directions, say for $\bar{M} = 4$, $\{\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3, \mathcal{U}_4\}$, out of the possible $M$ directions, where $\bar{M} \leq M$. Let us also assume that $\|\mathcal{U}_i\|_2^2 = 1$ and that the input-signal vector is represented as

$$\mathbf{u}(n) = \pm r_n \mathcal{U}_i$$

where $i \in \{1, \ldots, \bar{M}\}$, such that $\|\mathbf{u}(n)\|_2^2 = r^2(n)$. For simplicity, we assume that each direction is equally likely to occur with probability $p = 1/\bar{M}$.

Let $\hat{\mathbf{R}}^{-1}(0) = \mathbf{I}$ and $\mathbf{u}(1) = \pm r(1)\mathcal{U}_i$ be the hypothetical input-signal vector at instant $n = 1$. The estimate of the input-signal autocorrelation matrix at this instant is

$$\hat{\mathbf{R}}^{-1}(1) = \mathbf{I} + \left(\frac{1}{2r^2(1)} - 1\right)\mathcal{U}_i\mathcal{U}_i^T$$

where $i \in \{1, \ldots, 4\}$ for $\bar{M} = 4$, and $\tau(1) = r^2(1)$ with probability $p = 1$. At time instant $n = 2$, the input signal can either lie in the direction of $\mathbf{u}(1)$ or be orthogonal to it. Matrix $\hat{\mathbf{R}}^{-1}(2)$ will, therefore, fall into one of the two possibilities

$$\hat{\mathbf{R}}^{-1}(2) = \begin{cases} \mathbf{I} + \left(\frac{1}{2r^2(2)} - 1\right)\mathcal{U}_i\mathcal{U}_i^T, & i \in \{1, \ldots, 4\}, \\ & p = 1/4 \\ \mathbf{I} + \left(\frac{1}{2r^2(1)} - 1\right)\mathcal{U}_i\mathcal{U}_i^T + \left(\frac{1}{2r^2(2)} - 1\right)\mathcal{U}_j\mathcal{U}_j^T, & i, j \underset{j \neq i}{\in} \{1, \ldots, 4\}, \\ & p = 3/4 \end{cases}$$

and

$$\tau(2) = \begin{cases} r^2(2)/2r^2(1), & p = 1/4 \\ r^2(2), & p = 3/4 \end{cases}$$

At time instant $n = 3$, we have, for $i,\ j,\ k \underset{i\neq j\neq k}{\in} \{1, \ldots, 4\}$,

$$\hat{\mathbf{R}}^{-1}(3) = \begin{cases} \mathbf{I} + \left(\frac{1}{2r^2(3)} - 1\right)\mathcal{U}_i\mathcal{U}_i^T, & p = 1/16 \\ \mathbf{I} + \left(\frac{1}{2r^2(2)} - 1\right)\mathcal{U}_i\mathcal{U}_i^T + \left(\frac{1}{2r^2(3)} - 1\right)\mathcal{U}_k\mathcal{U}_k^T, & p = 3/16 \\ \mathbf{I} + \left(\frac{1}{2r^2(3)} - 1\right)\mathcal{U}_i\mathcal{U}_i^T + \left(\frac{1}{2r^2(2)} - 1\right)\mathcal{U}_j\mathcal{U}_j^T, & p = 3/16 \\ \mathbf{I} + \left(\frac{1}{2r^2(1)} - 1\right)\mathcal{U}_i\mathcal{U}_i^T + \left(\frac{1}{2r^2(3)} - 1\right)\mathcal{U}_j\mathcal{U}_j^T, & p = 3/16 \\ \mathbf{I} + \left(\frac{1}{2r^2(1)} - 1\right)\mathcal{U}_i\mathcal{U}_i^T + \left(\frac{1}{2r^2(2)} - 1\right)\mathcal{U}_j\mathcal{U}_j^T + \left(\frac{1}{2r^2(3)} - 1\right)\mathcal{U}_k\mathcal{U}_k^T, & p = 3/8 \end{cases}$$

and

$$\tau(3) = \begin{cases} r^2(3), & p = 9/16 \\ r^2(3)/2r^2(1), & p = 3/16 \\ r^2(3)/2r^2(2), & p = 1/4 \end{cases}$$

Similarly, for $n > 3$, a general formula for $\tau(n)$ for an arbitrary number of possible directions can be obtained as

$$\tau(n) = \begin{cases} r^2(n), & p = \left(\frac{\tilde{M}-1}{\tilde{M}}\right)^{n-1} \\ r^2(n)/2r^2(i), & i \in \{1, \ldots, n-1\},\ p = \frac{1}{\tilde{M}} \cdot \left(\frac{\tilde{M}-1}{\tilde{M}}\right)^{n-i-1} \end{cases}$$

where $\tilde{M} \leq M$ is the number of possible directions. This formula enables a very good examination of the typical behavior of parameter $\tau(n)$ even for general signals. Initially $\tau(n)$ must be small, assuming $u(n) = 0$ for $n \leq 0$. As the input-signal vector is filled, $\tau(n)$ becomes larger as either $r^2(n)$ or $r^2(n)/2r^2(i)$ is large with a high probability. As time progresses, the highest probability will be that $\tau(n)$ will be a ratio between norms of two input-signal vectors which are not far apart in time, the mean value of $\tau(n)$ being roughly $1/2$. The greater the order of the filter, the closer these norms must be, yielding smaller fluctuations around the mean.

As the number of iterations increases, the model suggests that the effect of the initial value $\hat{\mathbf{R}}^{-1}(0)$ is enduring, being modified only for certain directions. This partially explains the selective memory sometimes attributed to this class of algorithms [78], and illustrates why $\hat{\mathbf{R}}^{-1}(n)$ can remain positive definite even for nonpersistent excitations.

### 3.3.4 Excess Mean-Squared Error

We now examine the mean-squared output error (MSE), which in the final analysis dictates how well the algorithm performs after the initial convergence. In general, the study of MSE is concerned with the contributions made by uncorrelated measurement noise added to the desired response, nonstationarity in the coefficients of the unknown plant, and quantization of internal variables of the algorithm. Usually the effects of quantization can be considered sufficiently small [25] whereas additive measurement noise and nonstationarities can increase considerably and manifest themselves at the output as excess MSE. Conventionally, the excess MSE is used together with the speed of convergence to classify and compare algorithms.

As far as the QN algorithm is concerned, the expressions for the excess MSE due to additive noise or nonstationarity do not differ from those obtained for the LMSN algorithms with variable convergence factor discussed in the previous chapter. The algorithms have the same updating equation for the coefficients, for they follow an approximated Newton direction and they employ an exact line search. The difference resides in the way the inverse of the autocorrelation matrix is estimated. Therefore, if the coefficients of the unknown plant change according to

$$\mathbf{w}_o(n) = \mathbf{w}_o(n-1) + \boldsymbol{\nu}(n) \tag{3.14}$$

the total excess MSE for the QN algorithm is given by

$$J_{ex}(n) = J_{min} + \sigma_u^2 \sigma_\nu^2 M^2 \tag{3.15}$$

where $J_{min}$ is the minimum mean-squared error present even if the coefficients of the adaptive filter were set as $\mathbf{w}_o(n)$, $\sigma_u^2$ is the variance of the input signal, and $\sigma_\nu^2$ is the variance of the elements of $\boldsymbol{\nu}(n)$.

For the LMSN algorithms discussed in the previous chapter, the estimation of $\mathbf{R}^{-1}$ is sufficiently accurate so that $E[\mathbf{R}\hat{\mathbf{R}}^{-1}(n)] = \mathbf{I}$ could be considered a reasonable approximation. However, as was noted before, instability is a real threat for these algorithms. Conversely, the QN algorithm guarantees stability by employing an estimate of the autocorrelation matrix that can, in some cases, be a very poor approximation for $\mathbf{R}$. As a consequence, these formulas can differ from the results obtained in simulations but, as will be shown in the next section, the discrepancies are not of great relevance.

## 3.4 Simulations

In the series of experiments to be described, the adaptive filters tested were used to identify an unknown system which, unless otherwise stated, was a 36th-order lowpass FIR filter with cutoff frequency equal to $0.3\omega_s$ and coefficients normalized such that the total power was equal to 1, and 32-bit floating-point arithmetic was employed. The code for both the RLS and QN algorithms enforced symmetry in matrix $\hat{\mathbf{R}}^{-1}(n)$ as this is a basic assumption used throughout the analysis.

The QN algorithm has also been successfully tested in a decision-feedback application showing excellent bit-error rates and speed of convergence [79].

*Example 1:*

In the first example the input signal was a sinusoid of amplitude equal to 1 and frequency equal to 1/16th of the sampling frequency embedded in noise. The noise had zero mean and variance 0.1, and was correlated by an 11th-order lowpass channel with cutoff frequency equal to $0.5\omega_s$. Measurement noise of zero mean and variance $10^{-8}$ was added to the desired response. Fig. 3.2 shows a comparison of the performance of the QN and RLS algorithms for the case where $\lambda = 0.9$. As can be seen, the RLS algorithm diverged while the QN algorithm remained stable throughout the experiment. This example illustrates the fast convergence and excellent robustness of the QN algorithm.

Fig. 3.3 shows a similar scenario where all the characteristics described above are

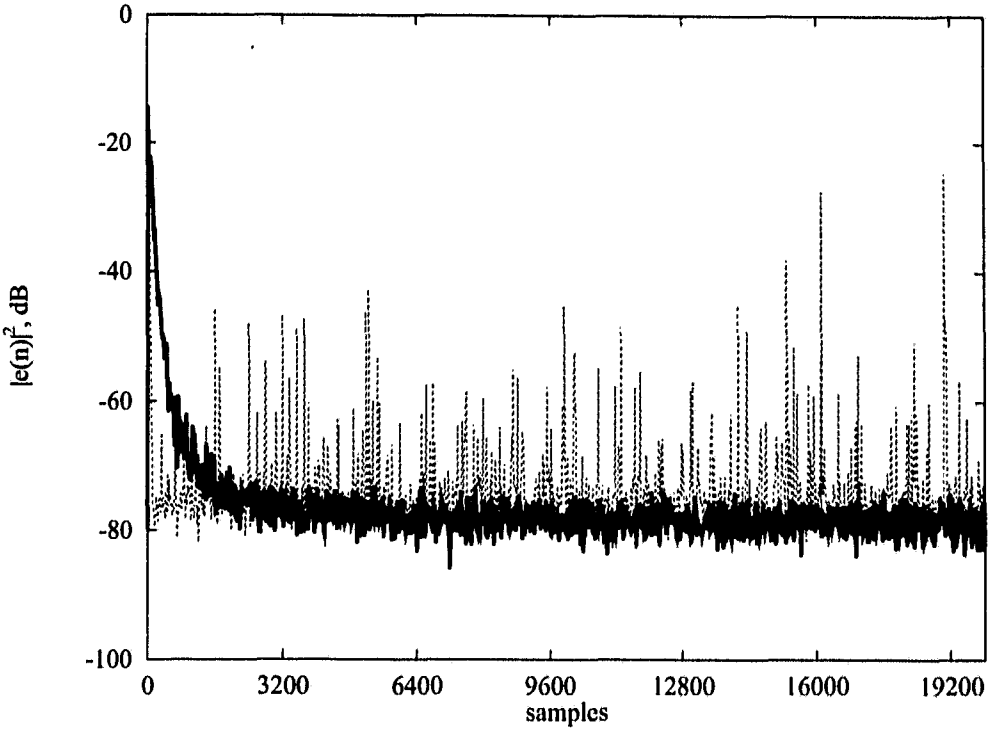



Figure 3.2: Learning curves for a stationary environment.
QN algorithm: ——
RLS algorithm with $\lambda = 0.9$ : - - -

maintained, except that the plant coefficients fluctuated around their values according to a first-order ARMA process. The variance of the noise was $10^{-6}$. Here again the QN algorithm performed favorably when compared to the RLS algorithm since the latter diverged.

Simulations for other values of $\lambda$ for the RLS algorithm were also performed for the stationary and nonstationary scenarios and the algorithm invariably diverged.

*Example 2:*

This experiment was designed to illustrate the behavior and effect of parameter $\tau(n)$ for different input signals. Fig. 3.4 shows a plot of $\tau(n)$ for the case where the input signal was a sinusoid of amplitude 10 and frequency equal to 1/16th of the sampling frequency and the case where the input signal was a white Gaussian noise. It can be verified that when the signal was a sinusoid after few iterations $\tau(n)$ becomes equal

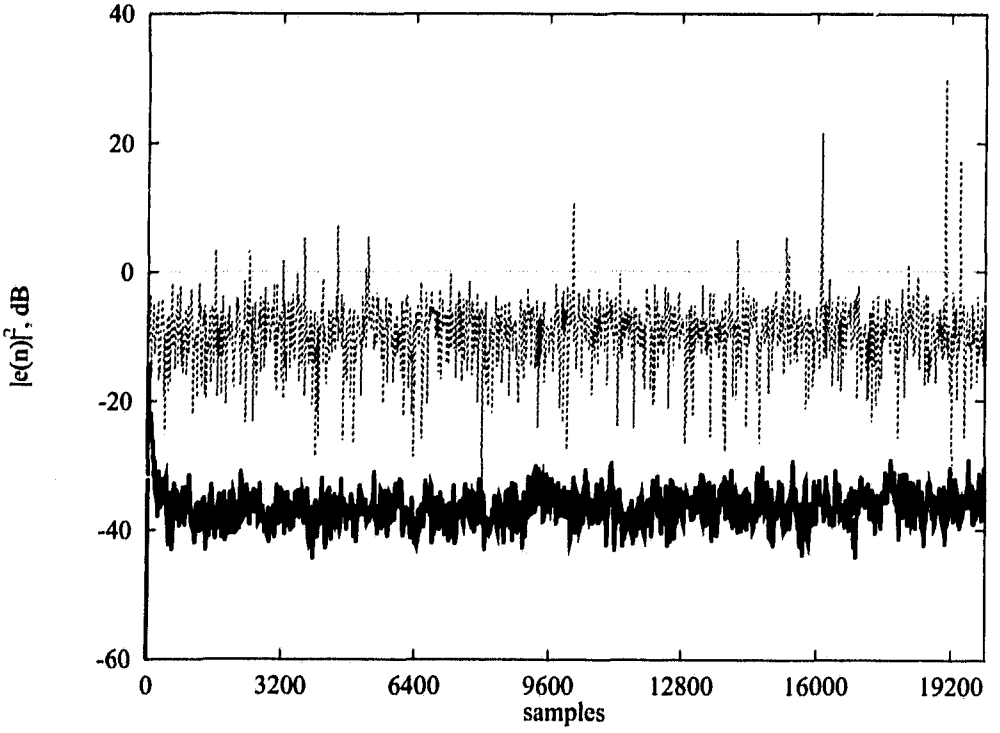



Figure 3.3: Learning curves for a nonstationary environment.
QN algorithm: ——
RLS algorithm with $\lambda = 0.9$ : - - -

to 0.5 which causes the adaptation of $\hat{\mathbf{R}}^{-1}(n)$ to stop. For the white noise case, a behavior very close to what was described previously for the simplified model was obtained. Parameter $\tau(n)$ became large at first, but assumed a mean value of 0.5 after few iterations. This clearly shows that the model described in section 4.3 is a powerful analysis tool.

*Example 3:*

The third example illustrates the performance of the QN and RLS algorithms when 16-bit fixed-point arithmetic with 2's-complement representation is employed. Here the input signal was a zero-mean white Gaussian noise with variance 0.1 correlated by a 11th-order lowpass FIR filter with cutoff frequency equal to $0.5\,\omega_N$. Measurement noise in this case was a zero-mean noise signal with variance $10^{-3}$. No scaling or rescuing procedures were used and overflow was treated by saturating the register. The

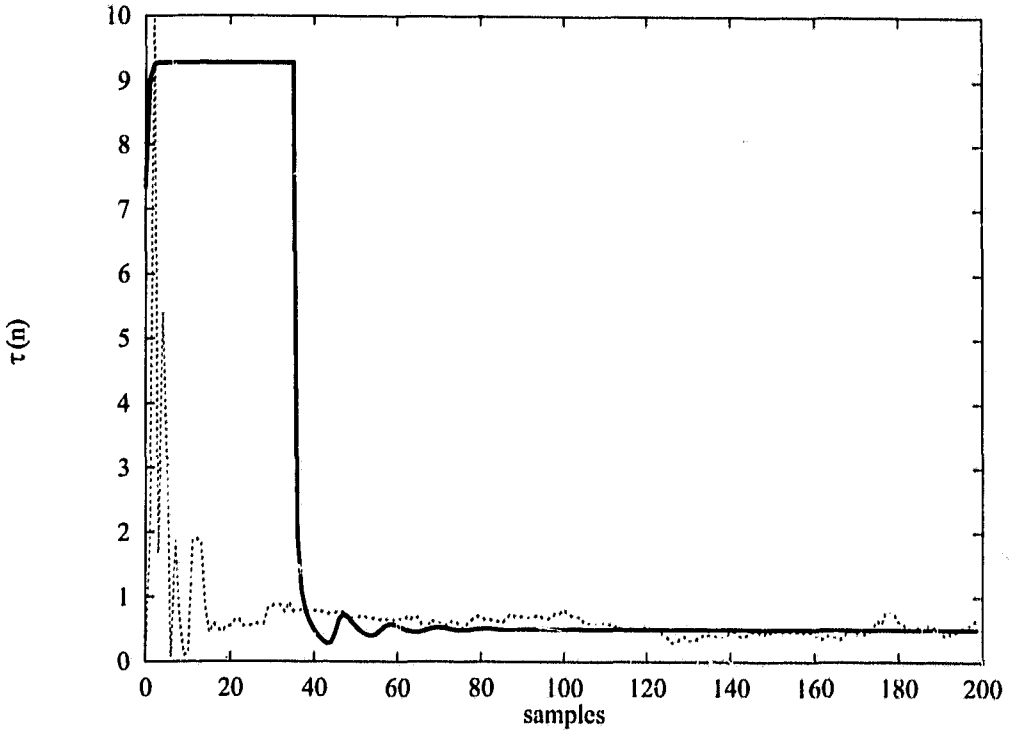



Figure 3.4: Evolution of $\tau(n)$.
Sinusoidal input signal: ——
White Gaussian noise: - - -

only modification introduced in the code of the QN algorithm was the initialization of $\hat{\mathbf{R}}^{-1}(n)$ as $0.5\,\mathbf{I}$ and the detection of a zero value for $\tau(n)$. This could arise when $\tau(n)$ was smaller than the precision used, and in this case $\tau(n)$ was made equal to the smallest machine representable number. It can be seen in Fig. 3.5 that the RLS algorithm diverged, as expected, while the QN algorithm remained stable. Longer simulations employing the QN algorithm were also carried out without any sign of instability. As for the RLS algorithm divergence occurred for all values of $\lambda$ used.

*Example 4:*

The fourth example used the same scenario as in example 3, except that the QN algorithm was tested in fixed-point arithmetic with different number of bits, ranging from 12 to 20, and there was no additive noise. To further stress the excellent robustness properties of the QN algorithm, Fig. 3.6 shows smoothed learning curves for the first

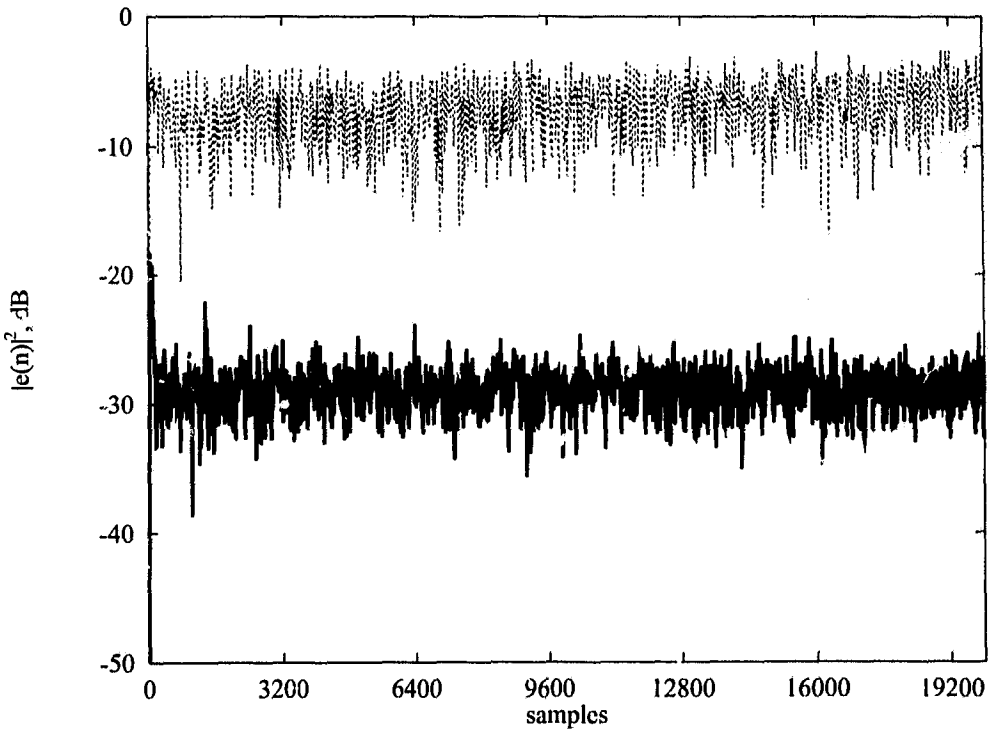



Figure 3.5: Learning curves for fixed-point arithmetic.
QN algorithm: ——
RLS algorithm with $\lambda = 0.9$ : - - -

100,000 iterations. No signs of instability were detected.

*Example 5:*

The fifth and last example was designed to verify the formulas provided for the excess MSE for the QN algorithm against different system orders. The input signal was white Gaussian noise with zero mean and variance 0.1. In order to test the formula for stationary systems, measurement noise of variance $10^{-4}$ was added to the desired response. For nonstationary systems, no additive noise was used, but the coefficients of the plant were modified according to (3.14) with noise variance equal to $10^{-3}$. The results are presented in Table 3.I which shows the outcome of an ensemble of 50 runs.

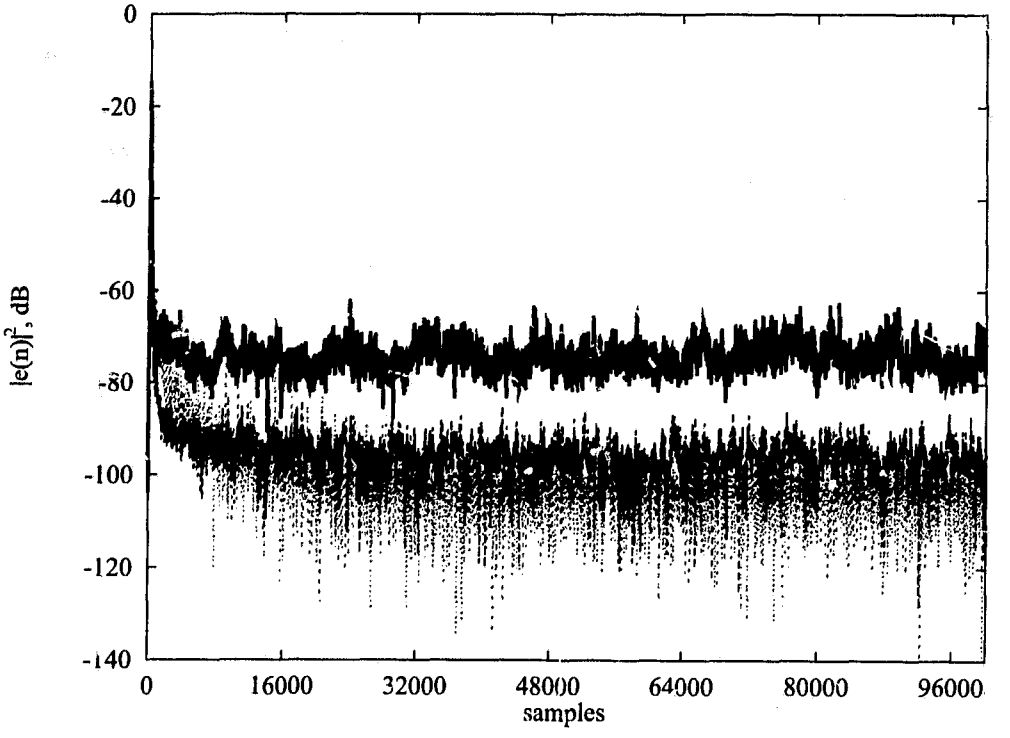



Figure 3.6: Learning curves for fixed-point arithmetic.
12 bits: ——
16 bits: - - -
20 bits: ....

## 3.5 Conclusions

The demand for adaptive filters that can perform reliably and converge fast even for highly correlated signals is increasing rapidly. As an attempt to provide adaptation algorithms with such qualities, several solutions have been proposed in the past which either improve the convergence speed of gradient-type algorithms or the robustness of Newton-type algorithms, most notably the RLS algorithm. However, these schemes are, in general, designed for a particular application or are cumbersome to implement. For instance, QR-decomposition-based algorithms present reasonably good performance with respect to robustness even for fixed-point arithmetic, but at the expense of complex and recursive equations that may require careful scaling. In this chapter, a new QN adaptive filtering algorithm was introduced as a possible alterna-

TABLE 3.I

EXCESS MSE FOR THE QN ALGORITHM, dB

| | Stationary | | Nonstationary | |
|---|---|---|---|---|
| Order, $M$ | Simulation | Eq. (3.15) | Simulation | Eq. (3.15) |
| 31 | −39.52 | −40.00 | −8.42 | −10.17 |
| 63 | −39.80 | −40.00 | −2.31 | −4.01 |

tive to RLS and LMS-Newton algorithms when fast convergence is necessary. The algorithm is based on fairly known classical optimization techniques, but resembles the LMS-Newton algorithm with variable convergence factor discussed in the previous chapter. An essential feature of the algorithm is that it employs an estimate of the inverse autocorrelation matrix $\hat{\mathbf{R}}^{-1}(n)$ which remains positive definite and bounded for a bounded input signal; in addition, the requirement of persistent excitation can be relaxed. Since the positive definiteness of $\hat{\mathbf{R}}^{-1}(n)$ is a necessary condition for stability of adaptation algorithms, the QN algorithm presented can operate in applications where other Newton-type algorithms as well as the RLS algorithm are likely to diverge.

The behavior and effect of important internal variables has been studied through a simplified model for the input signal, and simulations have shown that most of the conclusions can be extended to more general and practical signals. The algorithm does not require special rescuing procedures and a proof of internal stability was provided. The algorithm requires very little interference from the designer since no parameters need to be adjusted, and stability is guaranteed without bias. Furthermore, the algorithm was found to perform reliably with no scaling at all, provided that the desired and input signals are scaled to fit the processor's dynamic range.

Simulations for different and highly correlated input signals were carried out where floating as well as fixed-point arithmetic was employed, and in all of them the QN algorithm remained stable.

# Chapter 4

# Fast LMSN Algorithm

## 4.1 Introduction

The issues of computational complexity and robustness of adaptation algorithms are of great importance and have received, as a consequence, considerable attention in the recent past. In the previous chapters, algorithms were developed, analyzed, and tested, and their robustness was verified in several situations. In this chapter, more efficient implementations of these algorithms will be discussed.

## 4.2 Fast LMSN Algorithm

The family of fast algorithms rely on an efficient method for the solution of (1.20), namely,

$$\hat{\mathbf{R}}(n)\mathbf{w}(n) = \hat{\mathbf{p}}(n)$$

The conventional LMSN and RLS algorithms exploit the recursive nature of the updating formula for $\hat{\mathbf{R}}(n)$ and $\hat{\mathbf{p}}(n)$ in order to obtain a solution with complexity $\mathcal{O}(M^2)$. However, the number of multiplications can be further reduced by updating the gain vector, which is defined as

$$\mathbf{k}(n) = \hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n)$$

instead of updating $\hat{\mathbf{R}}^{-1}(n)$ [34]. The approach used here is algebraic as opposed to the geometric approach used in [32], [80].

The algorithm referred to as fast LMSN algorithm hereafter is, in fact, an implementation of the algorithm II discussed in chapter 2 which has a computational complexity of $\mathcal{O}(M)$ [44].

Before we go deeper into the algebraic details, we discuss the principle behind the equations. Let us define an extended-in-order autocorrelation matrix analogous to (1.16) as

$$\hat{\mathbf{R}}_{M+1}(n) = \alpha \sum_{i=1}^{n} (1-\alpha)^{n-i} \mathbf{u}_{M+1}(i) \mathbf{u}_{M+1}^T(i)$$

where the extended-in-order input vector is defined as

$$\mathbf{u}_{M+1}(n) = [u(n)\ u(n-1)\ \cdots\ u(n-M)]$$

and the initial value $\hat{\mathbf{R}}_{M+1}(0)$ has been dropped. It is easy to verify that $\hat{\mathbf{R}}_{M+1}(n)$ can be partitioned either as

$$\hat{\mathbf{R}}_{M+1}(n) = \begin{bmatrix} \hat{\mathbf{R}}(n) & \boldsymbol{\theta}_b(n) \\ \boldsymbol{\theta}_b^T(n) & \mathcal{D}_b(n) \end{bmatrix} \tag{4.1}$$

or

$$\hat{\mathbf{R}}_{M+1}(n) = \begin{bmatrix} \mathcal{D}_f(n) & \boldsymbol{\theta}_f^T(n-1) \\ \boldsymbol{\theta}_f(n-1) & \hat{\mathbf{R}}(n-1) \end{bmatrix} \tag{4.2}$$

and its inverse can be obtained either as

$$\hat{\mathbf{R}}_{M+1}^{-1}(n) = \begin{bmatrix} \hat{\mathbf{R}}^{-1}(n) & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{\mathbf{a}(n)\mathbf{a}^T(n)}{\mathcal{D}_b(n) - \boldsymbol{\theta}_b^T(n)\hat{\mathbf{R}}^{-1}(n)\boldsymbol{\theta}_b(n)} \tag{4.3}$$

or

$$\hat{\mathbf{R}}_{M+1}^{-1}(n) = \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \hat{\mathbf{R}}^{-1}(n-1) \end{bmatrix} + \frac{\mathbf{c}(n)\mathbf{c}^T(n)}{\mathcal{D}_f(n) - \boldsymbol{\theta}_f^T(n-1)\hat{\mathbf{R}}^{-1}(n-1)\boldsymbol{\theta}_f(n-1)} \tag{4.4}$$

where

$$\mathbf{a}(n) = \begin{bmatrix} -\hat{\mathbf{R}}^{-1}(n)\boldsymbol{\theta}_b(n) \\ 1 \end{bmatrix} \tag{4.5}$$

$$\mathbf{c}(n) = \begin{bmatrix} 1 \\ -\hat{\mathbf{R}}^{-1}(n-1)\boldsymbol{\theta}_f(n-1) \end{bmatrix} \tag{4.6}$$

$$\mathcal{D}_b(n) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} u^2(i-M) \tag{4.7}$$

$$\mathcal{D}_f(n) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} u^2(i) \tag{4.8}$$

$$\boldsymbol{\theta}_b(n) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} u(i-M)\mathbf{u}(i) \tag{4.9}$$

$$\boldsymbol{\theta}_f(n-1) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} u(i)\mathbf{u}(i-1) \tag{4.10}$$

The possibility of partitioning the extended-in-order autocorrelation matrix either as (4.1) or (4.2) and, consequently, its inverse either as (4.3) or (4.4) is key to the development of the fast algorithms. As it will soon become clear, by calculating the extended-in-order Kalman-gain vector

$$\mathbf{k}(n) = \hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n)$$

using (4.3) and (4.4), a recursive formula for updating $\mathbf{k}(n)$ is obtained. But before we discuss the details involved in updating $\mathbf{k}(n)$, let us clarify the meaning of the variables in (4.5)–(4.10).

The subscripts $b$ and $f$ in(4.5)–(4.10) denote backward and forward, respectively, as these quantities have a meaningful interpretation in the backward and forward linear predictors. By comparing (4.5)–(4.10) with (1.20)–(1.22), it is easy to verify that $\mathbf{a}(n)$ and $\mathbf{c}(n)$ are backward and forward prediction-error vectors, while $\boldsymbol{\theta}_b(n)$ and $\boldsymbol{\theta}_f(n-1)$ are the crosscorrelation vectors for the backward and forward predictors. Notice that at instant $i$ the desired signal and input-signal vector for the backward predictor are $u(i-M)$ and $\mathbf{u}(i)$, respectively. Similarly, for the forward predictor the

desired signal and input-signal vector are $u(i)$ and $\mathbf{u}(i-1)$, respectively. By analogy with (1.19), we realize that the objective functions minimized by these predictors are

$$\mathcal{E}_b(n) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} \left[u(i-M) - \mathbf{u}^T(i)\mathbf{w}_b(n)\right]^2 \tag{4.11}$$

and

$$\mathcal{E}_f(n) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} \left[u(i) - \mathbf{u}^T(i-1)\mathbf{w}_f(n)\right]^2 \tag{4.12}$$

respectively. The vectors $\mathbf{w}_b(n)$ and $\mathbf{w}_f(n)$ that yield minimum $\mathcal{E}_b(n)$ and $\mathcal{E}_f(n)$ are the negatives of the upper and lower partitions of vectors $\mathbf{a}(n)$ and $\mathbf{c}(n)$ defined in (4.5) and (4.6), i.e.,

$$\mathbf{w}_b(n) = \hat{\mathbf{R}}^{-1}(n)\boldsymbol{\theta}_b(n) \tag{4.13}$$

$$\mathbf{w}_f(n) = \hat{\mathbf{R}}^{-1}(n-1)\boldsymbol{\theta}_f(n-1) \tag{4.14}$$

The minimum values achieved by $\mathcal{E}_b(n)$ and $\mathcal{E}_f(n)$ are, therefore,

$$\mathcal{E}_{b,min}(n) = \mathcal{D}_b(n) - \boldsymbol{\theta}_b^T(n)\hat{\mathbf{R}}^{-1}(n)\boldsymbol{\theta}_b(n) \tag{4.15}$$

and

$$\xi_{f,min}(n) = \mathcal{D}_f(n) - \boldsymbol{\theta}_f^T(n-1)\hat{\mathbf{R}}^{-1}(n-1)\boldsymbol{\theta}_f(n-1) \tag{4.16}$$

It is interesting to notice that $\mathbf{w}_b(n)$ and $\mathbf{w}_f(n)$ are RLS-like solutions, since (4.11) and (4.12) are scaled versions of the classical weighted RLS-algorithm objective function. More specifically, (4.11) and (4.12) are of the form of (1.28) with $2\mu = \alpha$, which is the condition for equivalence between the RLS and LMSN algorithms.

### 4.2.1 The Algorithm

Since $\mathbf{w}_b(n)$ and $\mathbf{w}_f(n)$ minimize the backward and forward objective functions, we can define the a priori and a posteriori output errors related to the predictors as

$$e_b(n) = u(n-M) - \mathbf{u}^T(n)\mathbf{w}_b(n-1) \tag{4.17}$$

$$\epsilon_b(n) = u(n-M) - \mathbf{u}^T(n)\mathbf{w}_b(n) \tag{4.18}$$

$$e_f(n) = u(n) - \mathbf{u}^T(n-1)\mathbf{w}_f(n-1) \tag{4.19}$$

$$\epsilon_f(n) = u(n) - \mathbf{u}^T(n-1)\mathbf{w}_f(n) \tag{4.20}$$

and obtain the updating expressions from (1.14) with $2\mu = \alpha$, as

$$\mathbf{w}_b(n) = \mathbf{w}_b(n-1) + \alpha e_b(n)\mathbf{k}(n) \tag{4.21}$$

$$\mathbf{w}_f(n) = \mathbf{w}_f(n-1) + \alpha e_f(n)\mathbf{k}(n-1) \tag{4.22}$$

The backward and forward predictors are important tools for deriving an efficient scheme for updating the Kalman-gain vector. By premultiplying the extended input vector $\mathbf{u}_{M+1}(n)$ by $\hat{\mathbf{R}}_{M+1}^{-1}(n)$ as in (4.3), we have

$$\mathbf{k}_{M+1}(n) = \begin{bmatrix} \mathbf{k}(n) \\ 0 \end{bmatrix} + \frac{u(n-M) - \boldsymbol{\theta}_b^T(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{u}(n)}{\mathcal{D}_b(n) - \boldsymbol{\theta}_b^T(n)\hat{\mathbf{R}}^{-1}(n)\boldsymbol{\theta}_b(n)}\mathbf{a}(n) \tag{4.23}$$

Equivalently, by premultiplying $\mathbf{u}_{M+1}(n)$ by $\hat{\mathbf{R}}_{M+1}^{-1}(n)$ as in (4.4), we get

$$\mathbf{k}_{M+1}(n) = \begin{bmatrix} 0 \\ \mathbf{k}(n-1) \end{bmatrix} + \frac{u(n) - \boldsymbol{\theta}_f^T(n-1)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{u}(n-1)}{\mathcal{D}_f(n) - \boldsymbol{\theta}_f^T(n-1)\hat{\mathbf{R}}^{-1}(n-1)\boldsymbol{\theta}_f(n-1)}\mathbf{c}(n) \tag{4.24}$$

The relation between $\mathbf{k}(n-1)$ and $\mathbf{k}(n)$ is obtained by comparing (4.23) and (4.24), which can be rewritten in terms of the backward and forward predictors as

$$\mathbf{k}_{M+1}(n) = \begin{bmatrix} \mathbf{k}(n) \\ 0 \end{bmatrix} + \frac{\epsilon_b(n)}{\mathcal{E}_{b,min}(n)} \begin{bmatrix} -\mathbf{w}_b(n) \\ 1 \end{bmatrix} \tag{4.25}$$

and

$$\mathbf{k}_{M+1}(n) = \begin{bmatrix} 0 \\ \mathbf{k}(n-1) \end{bmatrix} + \frac{\epsilon_f(n)}{\mathcal{E}_{f,min}(n)} \begin{bmatrix} 1 \\ -\mathbf{w}_f(n) \end{bmatrix} \tag{4.26}$$

In order to obtain recursive relations for all the variables, we can express $\mathcal{E}_{b,min}(n)$ and $\mathcal{E}_{f,min}(n)$ as functions of $\mathcal{E}_{b,min}(n-1)$ and $\mathcal{E}_{f,min}(n-1)$, respectively. From (4.7), (4.9), and (4.15), we have

$$\begin{aligned}\mathcal{E}_{b,min}(n) &= \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i}u^2(i-M) - \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i}u(i-M)\mathbf{u}^T(i)\mathbf{w}_b(n) \\ &= (1-\alpha)\mathcal{D}_b(n-1) + \alpha u^2(n-M) - (1-\alpha)\theta_b^T(n-1)\mathbf{w}_b(n) \\ &\quad - \alpha u(n-M)\mathbf{u}^T(n)\mathbf{w}_b(n)\end{aligned} \tag{4.27}$$

By combining (4.13), (4.21), and (4.27), we obtain

$$\begin{aligned}\mathcal{E}_{b,min}(n) &= (1-\alpha)\mathcal{E}_{b,min}(n-1) + \alpha u^2(n-M) - \alpha(1-\alpha)e_b(n)\theta_b^T(n-1)\mathbf{k}(n) \\ &\quad - \alpha u(n-M)\mathbf{u}^T(n)\mathbf{w}_b(n) \\ &= (1-\alpha)\mathcal{E}_{b,min}(n-1) + \alpha\omega(n-M)\epsilon_b(n) \\ &\quad - \alpha(1-\alpha)e_b(n)\theta_b^T(n-1)\mathbf{k}(n)\end{aligned}$$

Now from (4.13) and (1.15), we have

$$\begin{aligned}\alpha(1-\alpha)e_b(n)\theta_b^T(n-1)\mathbf{k}(n) &= \alpha(1-\alpha)e_b(n)\mathbf{w}_b^T(n-1)\hat{\mathbf{R}}(n-1)\mathbf{k}(n) \\ &= \alpha e_b(n)\mathbf{w}_b^T(n-1)\left[\hat{\mathbf{R}}(n) - \alpha\mathbf{u}(n)\mathbf{u}^T(n)\right]\mathbf{k}(n) \\ &= \alpha e_b(n)\left[1-\alpha\mathbf{u}^T(n)\mathbf{k}(n)\right]\mathbf{w}_b^T(n-1)\mathbf{u}(n)\end{aligned}$$

Furthermore, from (4.18) and (4.21) we can obtain the relation

$$\begin{aligned}\epsilon_b(n) &= u(n-M) - \mathbf{u}^T(n)\left[\mathbf{w}_b(n-1) + \alpha e_b(n)\mathbf{k}(n)\right] \\ &= e_b(n)\psi(n)\end{aligned}$$

where

$$\psi(n) = 1 - \alpha\mathbf{u}^T(n)\mathbf{k}(n)$$

Therefore,

$$\mathcal{E}_{b,min}(n) = (1-\alpha)\mathcal{E}_{b,min}(n-1) + \alpha e_b(n)\epsilon_b(n) \tag{4.28}$$

Similarly, for the forward predictor

$$\mathcal{E}_{f,min}(n) = (1-\alpha)\mathcal{E}_{f,min}(n-1) + \alpha e_f(n)\epsilon_f(n) \tag{4.29}$$

where

$$\epsilon_f(n) = \psi(n-1)e_f(n)$$

The bridge between $\psi(n-1)$ and $\psi(n)$ is also based on the extended vectors. If we let

$$\psi_{M+1}(n) = 1 - \alpha \mathbf{u}_{M+1}^T(n)\mathbf{k}_{M+1}(n)$$

we can show, by using (4.25) and (4.28), that

$$\begin{aligned}
\psi_{M+1}(n) &= 1 - \alpha \mathbf{u}^T(n)\mathbf{k}(n) - \frac{\alpha\epsilon_b(n)}{\mathcal{E}_{b,min}(n)}\left[u(n-M) - \mathbf{u}^T(n)\mathbf{w}_b(n)\right] \\
&= \psi(n) - \frac{\alpha\epsilon_b^2(n)}{\mathcal{E}_{b,min}(n)} \\
&= \psi(n)\left[1 - \frac{\alpha e_b(n)\epsilon_b(n)}{\mathcal{E}_{b,min}(n)}\right] \\
&= \frac{(1-\alpha)\psi(n)\mathcal{E}_{b,min}(n-1)}{\mathcal{E}_{b,min}(n)}
\end{aligned} \tag{4.30}$$

Similarly, by using (4.26) and (4.29), we can show that

$$\psi_{M+1}(n) = \frac{(1-\alpha)\psi(n-1)\mathcal{E}_{f,min}(n-1)}{\mathcal{E}_{f,min}(n)} \tag{4.31}$$

Furthermore, $\epsilon_b(n)$ does not need to be explicitly calculated once the last element of $\mathbf{k}_{M+1}(n)$ is available from (4.25). Therefore,

$$\begin{aligned}
e_b(n) &= \frac{k_{M+1,M+1}(n)\mathcal{E}_{b,min}(n)}{\psi(n)} \\
&= (1-\alpha)\mathcal{E}_{b,min}(n-1)\tilde{k}_{M+1,M+1}(n)
\end{aligned} \tag{4.32}$$

where

$$\tilde{\mathbf{k}}_{M+1}(n) = \mathbf{k}_{M+1}(n)/\psi_{M+1}(n)$$

with $k_{M+1,M+1}(n)$ and $\tilde{k}_{M+1,M+1}(n)$ denoting the $(M+1)$th element of $\mathbf{k}_{M+1}(n)$ and $\tilde{\mathbf{k}}_{M+1}(n)$, respectively. In the light of the definition of the normalized Kalman-gain vector, $\tilde{\mathbf{k}}_{M+1,M+1}(n)$, the following updating relations can be derived for all variables, completing the set of equations for the fast LMSN algorithm.

From (4.22), (4.26), (4.29), and (4.31), we have

$$\tilde{\mathbf{k}}_{M+1}(n) = \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}(n-1) \end{bmatrix} + \frac{e_f(n)}{(1-\alpha)\mathcal{E}_{f,min}(n-1)} \begin{bmatrix} 1 \\ -\mathbf{w}_f(n-1) \end{bmatrix} \tag{4.33}$$

Similarly, from (4.21), (4.25), (4.30), and (4.28), we get

$$\begin{bmatrix} \tilde{\mathbf{k}}(n) \\ 0 \end{bmatrix} = \tilde{\mathbf{k}}_{M+1}(n) - \tilde{k}_{M+1,M+1}(n) \begin{bmatrix} -\mathbf{w}_b(n-1) \\ 1 \end{bmatrix}$$

Parameter $\psi(n)$ can also be more efficiently updated by using $\tilde{\mathbf{k}}_{M+1}(n)$. From (4.28) and (4.32), we can obtain

$$\psi(n) = \frac{\psi_{M+1}(n)}{1 - \alpha\tilde{k}_{M+1,M+1}(n)\psi_{M+1}(n)e_b(n)}$$

The equations comprising the fast transversal LMSN algorithm are summarized in Table 4.I. Note that a rescue procedure [32] whereby the algorithm is reinitialized but the previous solution for $\mathbf{w}$ is maintained is also included.

An interesting point to notice is that in updating the fast LMSN algorithm, we must implement four filters: the backward prediction-error filter, the forward prediction-error filter, the Kalman-gain vector, and the original transversal filter. The Kalman gain vector is a filter on its own, for it can be easily shown that it is a least-squares estimator of a hypothetical desired signal described as

$$d_k(i) = \begin{cases} 1/\alpha, & i = n \\ 0, & \text{otherwise} \end{cases}$$

TABLE 4.1

FAST TRANSVERSAL LMSN ALGORITHM

$$e_f(n) = u(n) - \mathbf{w}_f^T(n-1)\mathbf{u}(n-1)$$

$$\epsilon_f(n) = \psi(n-1)e_f(n)$$

$$\mathcal{E}_f(n) = (1-\alpha)\mathcal{E}_f(n-1) + \alpha\epsilon_f(n)e_f(n)$$

$$\psi_{M+1}(n) = \frac{(1-\alpha)\mathcal{E}_f(n-1)}{\mathcal{E}_f(n)}\psi(n-1)$$

$$\tilde{\mathbf{k}}_{M+1}(n) = \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}(n-1) \end{bmatrix} + \frac{e_f(n)}{(1-\alpha)\mathcal{E}_f(n-1)} \begin{bmatrix} 1 \\ -\mathbf{w}_f(n-1) \end{bmatrix}$$

$$\mathbf{w}_f(n) = \mathbf{w}_f(n-1) + \alpha\epsilon_f(n)\tilde{\mathbf{k}}(n-1)$$

$$e_b(n) = (1-\alpha)\mathcal{E}_b(n-1)\tilde{k}_{M+1,M+1}(n)$$

$$\psi(n) = \frac{\psi_{M+1}(n)}{1-\alpha\tilde{k}_{M+1,M+1}(n)e_b(n)\psi_{M+1}(n)}$$

if $[1 - \alpha\tilde{k}_{M+1,M+1}(n)e_b(n)\psi_{M+1}(n)] > 0.0$:

$$\epsilon_b(n) = \psi(n)e_b(n)$$

$$\mathcal{E}_b(n) = (1-\alpha)\mathcal{E}_b(n-1) + \alpha\epsilon_b(n)e_b(n)$$

$$\begin{bmatrix} \tilde{\mathbf{k}}(n) \\ 0 \end{bmatrix} = \tilde{\mathbf{k}}_{M+1}(n) - \tilde{k}_{M+1,M+1}(n) \begin{bmatrix} -\mathbf{w}_b(n-1) \\ 1 \end{bmatrix}$$

$$\mathbf{w}_b(n) = \mathbf{w}_b(n-1) + \alpha\epsilon_b(n)\tilde{\mathbf{k}}(n)$$

$$\epsilon(n) = \psi(n)[d(n) - \mathbf{w}^T(n-1)\mathbf{u}(n)]$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \frac{\alpha\epsilon(n)}{1-\psi(n)}\tilde{\mathbf{k}}(n)$$

else: (Rescue)

$$\mathbf{w}(n) = \mathbf{w}(n-1)$$

$$\mathbf{w}_b(n) = \mathbf{w}_f(n) = \tilde{\mathbf{k}}(n) = \mathbf{0}$$

$$\psi(n) = 1.0$$

$$\mathcal{E}_f(n) = (1-\alpha)^M\eta$$

$$\mathcal{E}_b(n) = \eta$$

Note: $\eta$ is a positive factor that serves as a weight for $\mathbf{w}(n-1)$ in the objective function after the rescue.

that takes $\mathbf{u}(n)$ as the input-signal vector. The estimates of the input-signal autocorrelation matrix and the crosscorrelation vector between the desired signal and the input-signal vector are $\hat{\mathbf{R}}(n)$ and

$$\mathbf{u}(n) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i}\mathbf{u}(i)$$

Accordingly, $\psi(n)$ is the estimation error of the process. Other interesting interpretations of $\psi(n)$ have been established for the RLS algorithm [8] and can be easily extended for the LMSN algorithm.

As for the adaptation of the coefficients of $\mathbf{w}(n)$, the convergence factor $\mu(n)$ can be calculated once the Kalman-gain vector is available. From (2.8) we have

$$\mu(n) = \frac{1}{2\mathbf{u}^T(n)\mathbf{k}(n)}$$

$$= \frac{1}{2\psi(n)\mathbf{u}^T(n)\tilde{\mathbf{k}}(n)}$$

A more efficient formula for $\mu(n)$ can be derived if we notice that

$$\mathbf{u}^T(n)\mathbf{k}(n) = \frac{1-\psi(n)}{\alpha}$$

Therefore,

$$\mu(n) = \frac{\alpha}{2\left[1-\psi(n)\right]}$$

which avoids the unnecessary calculation of the inner product $\mathbf{u}^T(n)\mathbf{k}(n)$. Vector $\mathbf{w}(n)$ is updated as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \frac{\alpha\epsilon(n)}{\left[1-\psi(n)\right]}\tilde{\mathbf{k}}(n)$$

### 4.2.2 Initialization

As with the QR-LMSN algorithm described previously, the fast LMSN algorithm also requires a special initialization procedure. Unfortunately, though, the initialization for the latter poses difficulties that must be discussed in detail. The main problem, as reported for the fast transversal filter (FTF) algorithm [8] [32], is that the equations

that update the backward prediction-error filter and the Kalman-gain vector become ambiguous for $n = M + 1$, and the algorithm cannot proceed.

Let us assume zero initial conditions for all states of the filter, except for $\psi(0)$ which is required to be 1, as will be soon explained. In the first iteration, $n = 1$, the variables of the filter assume the following values. From (4.19),

$$e_f(1) = u(1)$$

Since $\mathbf{w}_f(0) = \mathbf{0}$ and $\mathbf{k}(0) = \mathbf{0}$, from (4.22) we have

$$\mathbf{w}_f(1) = \mathbf{0}$$

and, therefore,

$$\epsilon_f(1) = e_f(1) = u(1)$$

which is in accordance with our assumption that $\psi(0) = 1$. From (4.29),

$$\mathcal{E}_{f,min}(1) = \alpha u^2(1)$$

and from (4.26), we have

$$\mathbf{k}_{M+1}(1) = \left[\frac{1}{\alpha u(1)} \; 0 \; \cdots \; 0\right]^T$$

Furthermore, from (4.21) and (4.25), we obtain

$$\mathbf{w}_b(1) = \mathbf{0}$$

and

$$\mathbf{k}(1) = \left[\frac{1}{\alpha u(1)} \; 0 \; \cdots \; 0\right]^T$$

From the definition of $\psi(n)$ or $\psi_{M+1}(n)$, we realize that

$$\psi(1) = \psi_{M+1}(1) = 0$$

and, finally, the output error and the coefficients of the filter can be calculated as

$$e(1) = d(1)$$

and

$$\mathbf{w}(1) = \alpha e(1)\mathbf{k}(1) = \left[\frac{d(1)}{u(1)} \; 0 \; \cdots \; 0\right]^T \tag{4.34}$$

The reason why the variable convergence factor was not used in the updating of $\mathbf{w}(1)$ is that for $n \leq M$ the system of equations to be recursively solved by the LMSN algorithm is underdetermined or exactly determined (in case $n = M$), and the a posteriori output error can be made equal to zero even with a constant convergence factor [8] [32]. In (4.34) the condition for equivalence with the RLS algorithm was observed, i.e., $2\mu = \alpha$. Another point we need to stress is that $\mathbf{k}(n)$ was used instead of $\tilde{\mathbf{k}}(n)$, since the latter cannot be determined once $\psi(1)$ and $\psi_{M+1}(1)$ are equal to zero.

In the subsequent iterations up to $n = M$, the algorithm proceeds such that $\psi(n) = \psi_{M+1}(n) = 0$, i.e., the a priori output errors $\epsilon_b(n)$, $\epsilon_f(n)$, and $\epsilon(n)$ are all equal to zero. Furthermore, from (4.26), we have

$$\mathbf{k}(n) = \left[0 \; \cdots \; 0 \; \frac{1}{\alpha u(1)} \; 0 \; \cdots \; 0\right]^T \tag{4.35}$$

with the nonzero entry being the $n$th element of the vector. However, for $n = M+1$, the ambiguity for updating the backward prediction-error filter and the Kalman-gain vector arises. At this iteration, the desired response for the backward prediction-error filter is nonzero for the first time and, as a consequence, the least-squares problem of both filters become the same as both operate on the same input signal vector and on desired signals that differ by a scalar. For the backward prediction-error filter, the desired response is

$$d_b(i) = \begin{cases} u(1), & i = n \\ 0, & \text{otherwise} \end{cases}$$

and, for the Kalman-gain vector, the desired response is

$$d_k(i) = \begin{cases} 1/\alpha, & i = n \\ 0, & \text{otherwise} \end{cases}$$

It is easier to verify the consequences of this fact by examining the equations. By substituting (4.21) in (4.25), we have

$$\begin{bmatrix} \mathbf{k}(n) \\ 0 \end{bmatrix} = \mathbf{k}_{M+1}(n) - \frac{\epsilon_b(n)}{\mathcal{E}_{b,min}(n)} \begin{bmatrix} -\mathbf{w}_b(n-1) \\ 1 \end{bmatrix} + \frac{\alpha \epsilon_b(n) e_b(n)}{\mathcal{E}_{b,min}(n)} \begin{bmatrix} \mathbf{k}(n) \\ 0 \end{bmatrix}$$

or, equivalently,

$$\left[1 - \frac{\alpha e_b(n)\epsilon_b(n)}{\mathcal{E}_{b,min}(n)}\right] \begin{bmatrix} \mathbf{k}(n) \\ 0 \end{bmatrix} = \mathbf{k}_{M+1}(n) - \frac{\epsilon_b(n)}{\mathcal{E}_{b,min}(n)} \begin{bmatrix} -\mathbf{w}_b(n-1) \\ 1 \end{bmatrix}$$

However, from (4.28) we know that

$$\left[1 - \frac{\alpha e_b(n)\epsilon_b(n)}{\mathcal{E}_{b,min}(n)}\right] = \frac{(1-\alpha)\mathcal{E}_{b,min}(n-1)}{\mathcal{E}_{b,min}(n)} = 0, \qquad \text{for } n = M+1$$

and, therefore, $\mathbf{k}(n)$ cannot be calculated. In order to overcome this problem, an order and time updating can be used during the initialization period, as originally proposed by Cioffi and Kailath for the RLS algorithm [32].

The trick for the successful initialization is to start the updating of the backward and forward prediction-error filters only at the second iteration. At the first iteration, these filters are of zero dimension, denoted as $\mathbf{w}_{b,0}(1)$ and $\mathbf{w}_{f,0}(1)$. We will use subscripts to denote the order of the vector hereafter. The order and time updating proceeds as follows.

For $n = 1$, the a priori and a posteriori forward prediction errors are calculated based on the zero-dimensional predictor, i.e.,

$$e_{f,0}(1) = u(1)$$

and

$$\epsilon_{f,0}(1) = u(1)$$

Therefore,

$$\mathcal{E}_{f,min,0}(1) = \alpha u^2(1)$$

A zero-dimensional Kalman-gain vector can also be used to calculate $\psi_0(1)$ as

$$\psi_0(1) = 1 - \alpha \mathbf{u}_0^T(1)\mathbf{k}_0(1) = 1$$

Vector $\mathbf{w}(n)$, on the other hand, is updated at this iteration based on $\mathbf{k}_1(1)$ obtained from (4.26) as

$$\mathbf{k}_1(1) = \frac{1}{\alpha u(1)}$$

and the output error,

$$e_1(1) = d(1) - \mathbf{u}_1^T(1)\mathbf{w}_1(0) = d(1)$$

Therefore,

$$\mathbf{w}_1(1) = \alpha e_1(1)\mathbf{k}_1(1) = \frac{d(1)}{u(1)}$$

For $n = 2, \ldots, M+1$, if we let

$$\mathbf{w}_{f,n-1}(n-1) = \begin{bmatrix} \mathbf{w}_{f,n-2}(n-1) \\ 0 \end{bmatrix}$$

then

$$\begin{aligned} e_{f,n-1}(n) &= u(n) - \mathbf{w}_{f,n-1}^T(n-1)\mathbf{u}_{n-1}(n-1) \\ &= u(n) - \mathbf{w}_{f,n-2}^T(n-1)\mathbf{u}_{n-2}(n-1) \\ &= e_{f,n-2}(n) \end{aligned}$$

From (4.22),

$$\begin{aligned} \epsilon_{f,n-2}(n) &= u(n) - \mathbf{w}_{f,n-2}^T(n)\mathbf{u}_{n-2}(n-1) \\ &= u(n) - \mathbf{w}_{f,n-2}^T(n-1)\mathbf{u}_{n-2}(n-1) \\ &\quad - \alpha e_{f,n-2}(n)\mathbf{k}_{n-2}^T(n-1)\mathbf{u}_{n-2}(n-1) \\ &= e_{f,n-2}(n)\left[1 - \alpha \mathbf{u}_{n-2}^T(n-1)\mathbf{k}_{n-2}(n-1)\right] \\ &= \psi_{n-2}(n-1)e_{f,n-2}(n) \end{aligned} \tag{4.36}$$

where

$$\psi_{n-2}(n-1) = 1 - \alpha \mathbf{u}_{n-2}^T(n-1)\mathbf{k}_{n-2}(n-1)$$

The coefficients of the forward prediction-error filter are updated according to

$$\mathbf{w}_{f,n-1}(n) = \mathbf{w}_{f,n-1}(n-1) + \alpha e_{f,n-1}(n)\mathbf{k}_{n-1}(n-1) \tag{4.37}$$

By using (4.35), (4.37) can be rewritten as

$$\begin{aligned}\mathbf{w}_{f,n-1}(n) &= \begin{bmatrix} \mathbf{w}_{f,n-2}(n-1) \\ 0 \end{bmatrix} + \alpha e_{f,n-1}(n) \begin{bmatrix} \mathbf{0} \\ \frac{1}{\alpha u(1)} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{w}_{f,n-2}(n-1) \\ \frac{e_{f,n-1}(n)}{u(1)} \end{bmatrix}\end{aligned}$$

One important point to notice here is that although $\epsilon_{f,n-2}(n) \neq 0$, we have $\epsilon_{f,n-1}(n) = 0$ for $2 \leq n \leq M$. This can be easily verified as

$$\begin{aligned}\epsilon_{f,n-1}(n) &= u(n) - \mathbf{w}_{f,n-1}^T(n)\mathbf{u}_{n-1}(n-1) \\ &= u(n) - \mathbf{w}_{f,n-1}^T(n-1)\mathbf{u}_{n-1}(n-1) - e_{f,n-1}(n) \\ &= 0\end{aligned}$$

The fact that $\epsilon_{f,n-1}(n) = 0$ for $2 \leq n \leq M$ is the reason why we have, from (4.26),

$$\mathbf{k}_1(1) = \frac{1}{\alpha u(1)}$$

and

$$\mathbf{k}_n(n) = \left[0 \cdots 0 \frac{1}{\alpha u(1)}\right]^T, \qquad n = 2, \ldots, M \tag{4.38}$$

On the other hand,

$$\mathbf{k}_{n-1}(n) = \begin{bmatrix} 0 \\ \mathbf{k}_{n-2}(n-1) \end{bmatrix} + \frac{\epsilon_{f,n-2}(n)}{\mathcal{E}_{f,n-2}(n)} \begin{bmatrix} 1 \\ -\mathbf{w}_{f,n-2}(n) \end{bmatrix} \tag{4.39}$$

which is different from the first $n-1$ elements of $\mathbf{k}_n(n)$. Accordingly,

$$\begin{aligned}\psi_{n-1}(n) &= 1-\alpha\mathbf{u}_{n-1}^T(n)\mathbf{k}_{n-1}(n)\\ &= 1-\alpha\mathbf{u}_{n-2}^T(n-1)\mathbf{k}_{n-2}(n-1)-\frac{\alpha\epsilon_{f,n-2}^2(n)}{\mathcal{E}_{f,n-2}(n)}\\ &= \psi_{n-2}(n-1)\left[1-\frac{\alpha\epsilon_{f,n-2}(n)e_{f,n-2}(n)}{\mathcal{E}_{f,n-2}(n)}\right]\\ &= \frac{\psi_{n-2}(n-1)(1-\alpha)\mathcal{E}_{f,n-2}(n-1)}{\mathcal{E}_{f,n-2}(n)}\end{aligned} \tag{4.40}$$

The updating of $\mathcal{E}_{f,n-2}(n)$ follows from (4.16) as

$$\begin{aligned}\mathcal{E}_{f,n-2}(n) &= \sum_{i=1}^{n}\alpha(1-\alpha)^{n-i}u^2(i)-\sum_{i=1}^{n}\alpha(1-\alpha)^{n-i}u(i)\mathbf{u}_{n-2}^T(i-1)\mathbf{w}_{f,n-2}(n)\\ &= \sum_{i=1}^{n-1}\alpha(1-\alpha)^{n-i}u^2(i)+\alpha u^2(n)\\ &\quad-\sum_{i=1}^{n-1}\alpha(1-\alpha)^{n-i}u(i)\mathbf{u}_{n-2}^T(i-1)\mathbf{w}_{f,n-2}(n)\\ &\quad-\alpha u(n)\mathbf{u}_{n-2}^T(n-1)\mathbf{w}_{f,n-2}(n)\\ &= (1-\alpha)\mathcal{D}_f(n-1)+\alpha u^2(n)-(1-\alpha)\boldsymbol{\theta}_{f,n-2}^T(n-2)\mathbf{w}_{f,n-2}(n)\\ &\quad-\alpha u(n)\mathbf{u}_{n-2}^T(n-1)\mathbf{w}_{f,n-2}(n)\end{aligned}$$

However,

$$\mathbf{w}_{f,n-2}(n)=\mathbf{w}_{f,n-2}(n-1)+\alpha e_{f,n-2}(n)\mathbf{k}_{n-2}(n-1)$$

Therefore,

$$\begin{aligned}\mathcal{E}_{f,n-2}(n) &= (1-\alpha)\mathcal{E}_{f,n-2}(n-1)+\alpha u(n)\left[u(n)-\mathbf{u}_{n-2}^T(n-1)\mathbf{w}_{f,n-2}(n)\right]\\ &\quad-\alpha(1-\alpha)e_{f,n-2}(n)\boldsymbol{\theta}_{f,n-2}^{\prime T}(n-2)\mathbf{k}_{n-2}(n-1)\end{aligned} \tag{4.41}$$

By using (4.10) and (4.14) on (4.41), we obtain

$$\begin{aligned}\mathcal{E}_{f,n-2}(n) &= (1-\alpha)\mathcal{E}_{f,n-2}(n-1) + \alpha u(n)\epsilon_{f,n-2}(n) \\ &\quad - \alpha e_{f,n-2}(n)\left[\boldsymbol{\theta}_{f,n-2}^T(n-1) - \alpha u(n)\mathbf{u}_{n-2}^T(n-1)\right]\mathbf{k}_{n-2}(n-1) \\ &= (1-\alpha)\mathcal{E}_{f,n-2}(n-1) + \alpha u(n)\epsilon_{f,n-2}(n) \\ &\quad - \alpha e_{f,n-2}(n)\left[\mathbf{w}_{f,n-2}^T(n)\mathbf{u}_{n-2}(n-1)\right. \\ &\quad \left. -\alpha u(n)\mathbf{u}_{n-2}^T(n-1)\mathbf{k}_{n-2}(n-1)\right] \qquad (4.42)\end{aligned}$$

If we substitute (4.36) in (4.42), we finally obtain

$$\begin{aligned}\mathcal{E}_{f,n-2}(n) &= (1-\alpha)\mathcal{E}_{f,n-2}(n-1) + \alpha u(n)e_{f,n-2}(n) \\ &\quad - \alpha^2 u(n)e_{f,n-2}(n)\mathbf{u}_{n-2}^T(n-1)\mathbf{k}_{n-2}(n-1) \\ &\quad - \alpha e_{f,n-2}(n)\left[\mathbf{w}_{f,n-2}^T(n)\mathbf{u}_{n-2}(n-1)\right. \\ &\quad \left. -\alpha u(n)\mathbf{u}_{n-2}^T(n-1)\mathbf{k}_{n-2}(n-1)\right] \\ &= (1-\alpha)\mathcal{E}_{f,n-2}(n-1) + \alpha e_{f,n-2}(n)\epsilon_{f,n-2}(n)\end{aligned}$$

On the other hand, we have

$$\begin{aligned}\mathcal{E}_{f,n-1}(n) &= \sum_{i=1}^{n}\alpha(1-\alpha)^{n-i}u^2(i) - \sum_{i=1}^{n}\alpha(1-\alpha)^{n-i}u(i)\mathbf{u}_{n-1}^T(i-1)\mathbf{w}_{f,n-1}(n) \\ &= \sum_{i=1}^{n-1}\alpha(1-\alpha)^{n-i}u^2(i) + \alpha u^2(n) \\ &\quad - \sum_{i=1}^{n-1}\alpha(1-\alpha)^{n-i}u(i)\mathbf{u}_{n-1}^T(i-1)\mathbf{w}_{f,n-1}(n) \\ &\quad - \alpha u(n)\mathbf{u}_{n-1}^T(n-1)\mathbf{w}_{f,n-1}(n)\end{aligned}$$

or, equivalently,

$$\begin{aligned}\mathcal{E}_{f,n-1}(n) &= (1-\alpha)\mathcal{D}_f(n-1) - (1-\alpha)\boldsymbol{\theta}_{f,n-1}^T(n-2)\mathbf{w}_{f,n-1}(n) \\ &\quad + \alpha u(n)\left[u(n) - \mathbf{u}_{n-1}^T(n-1)\mathbf{w}_{f,n-1}(n)\right] \\ &= (1-\alpha)\mathcal{D}_f(n-1) - (1-\alpha)\boldsymbol{\theta}_{f,n-1}^T(n-2)\mathbf{w}_{f,n-1}(n) \\ &\quad + \alpha u(n)\epsilon_{f,n-1}(n)\end{aligned}$$

However, since $\epsilon_{f,n-1}(n) = 0$ and

$$\begin{aligned}\boldsymbol{\theta}_{f,n-1}(n-2) &= \sum_{i=1}^{n-1}\alpha(1-\alpha)^{n-1-i}u(i)\mathbf{u}_{n-1}(i-1) \\ &= \sum_{i=1}^{n-1}\alpha(1-\alpha)^{n-1-i}u(i)\begin{bmatrix}\mathbf{u}_{n-2}(i-1)\\ 0\end{bmatrix} \\ &= \begin{bmatrix}\boldsymbol{\theta}_{f,n-2}(n-2)\\ 0\end{bmatrix}\end{aligned}$$

we have

$$\begin{aligned}\mathcal{E}_{f,n-1}(n) &= (1-\alpha)\mathcal{D}_f(n-1) - (1-\alpha)\left[\boldsymbol{\theta}_{f,n-2}^T(n-2)\ 0\right]^T\begin{bmatrix}\mathbf{w}_{f,n-2}(n-1)\\ \frac{e_{f,n-1}(n)}{u(1)}\end{bmatrix} \\ &= (1-\alpha)\mathcal{E}_{f,n-2}(n-1)\end{aligned}$$

In order to make the initialization consistent with the algorithm itself, $\tilde{\mathbf{k}}_{n-1}(n)$ must be updated instead of $\mathbf{k}_{n-1}(n)$. Therefore, from (4.39) and (4.40), we have

$$\begin{aligned}\tilde{\mathbf{k}}_{n-1}(n) &= \frac{\mathcal{E}_{f,n-2}(n)}{(1-\alpha)\mathcal{E}_{f,n-2}(n-1)}\begin{bmatrix}0\\ \tilde{\mathbf{k}}_{n-2}(n-1)\end{bmatrix} \\ &\quad + \frac{\epsilon_{f,n-2}(n)}{(1-\alpha)\psi_{n-2}(n-1)\mathcal{E}_{f,n-2}(n-1)}\begin{bmatrix}1\\ -\mathbf{w}_{f,n-2}(n)\end{bmatrix} \qquad (4.43)\end{aligned}$$

If we substitute (4.37) in (4.43), we obtain

$$\tilde{\mathbf{k}}_{n-1}(n) = \frac{\mathcal{E}_{f,n-2}(n)}{(1-\alpha)\mathcal{E}_{f,n-2}(n-1)} \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_{n-2}(n-1) \end{bmatrix}$$

$$+ \frac{\epsilon_{f,n-2}(n)}{(1-\alpha)\psi_{n-2}(n-1)\mathcal{E}_{f,n-2}(n-1)} \begin{bmatrix} 1 \\ -\mathbf{w}_{f,n-2}(n-1) \end{bmatrix}$$

$$- \frac{\alpha\epsilon_{f,n-2}(n)e_{f,n-2}(n)}{(1-\alpha)\mathcal{E}_{f,n-2}(n-1)} \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_{n-2}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_{n-2}(n-1) \end{bmatrix} + \frac{e_{f,n-2}(n)}{(1-\alpha)\mathcal{E}_{f,n-2}(n-1)} \begin{bmatrix} 1 \\ -\mathbf{w}_{f,n-2}(n-1) \end{bmatrix}$$

Finally, if we let

$$\mathbf{w}_n(n-1) = \begin{bmatrix} \mathbf{w}_{n-1}(n-1) \\ 0 \end{bmatrix}$$

then from (4.38), we have

$$\mathbf{w}_n(n) = \begin{bmatrix} \mathbf{w}_{n-1}(n-1) \\ \frac{e_{n-1}(n)}{u(1)} \end{bmatrix}, \qquad n = 2, \ \ldots, \ M$$

and, for $n = M+1$, we get

$$\mathbf{w}_{n-1}(n) = \mathbf{w}_{n-1}(n-1) + \alpha e_{n-1}(n)\psi_{n-1}(n)\tilde{\mathbf{k}}_{n-1}(n)$$

As the forward prediction-error filter and the Kalman-gain vector were order-updated starting at iteration $n = 2$, they need to be initialized until $n = M+1$. The backward prediction-error filter is only updated at iteration $n = M+1$, since for all $n < M+1$ the desired response of the backward predictor is zero. From (4.21), we have

$$\mathbf{w}_{b,n-1}(n) = \alpha u(1)\psi_{n-1}(n)\tilde{\mathbf{k}}_{n-1}(n), \qquad n = M+1$$

Since for $n = M+1$ we have $\mathcal{E}_{b,n-1}(n-1) = 0$ and $e_{b,n-1}(n) = u(1)$, from (4.28) the value of $\mathcal{E}_{b,n-1}(n)$ can be obtained as

$$\mathcal{E}_{b,n-1}(n) = \alpha\psi_{n-1}(n)u^2(1)$$

A summary of the initialization procedure is presented in Table (4.II).

TABLE 4.II

INITIALIZATION OF THE FAST TRANSVERSAL LMSN ALGORITHM

$n = 1$:

$\mathbf{w}_{f,0}(1) = \tilde{\mathbf{k}}_0(1)$ zero dimensional

$\mathcal{E}_{f,0}(1) = \alpha u^2(1)$

$\psi_0(1) = 1$

$e_0(1) = d(1)$

$\mathbf{w}_1(1) = d(1)/u(1)$

$n = 2,\ 3,\ \ldots,\ M+1$:

$e_{f,n-2}(n) = u(n) - \mathbf{w}_{f,n-2}^T(n-1)\mathbf{u}_{n-2}(n-1) = e_{f,n-1}(n)$

$\epsilon_{f,n-2}(n) = \psi_{n-2}(n-1)e_{f,n-2}(n)$

$\mathcal{E}_{f,n-2}(n) = (1-\alpha)\mathcal{E}_{f,n-2}(n-1) + \alpha\epsilon_{f,n-2}(n)e_{f,n-2}(n)$

$\mathcal{E}_{f,n-1}(n) = (1-\alpha)\mathcal{E}_{f,n-2}(n-1)$

$\psi_{n-1}(n) = \frac{(1-\alpha)\mathcal{E}_{f,n-2}(n-1)}{\mathcal{E}_{f,n-2}(n)}\psi_{n-2}(n-1)$

$$\tilde{\mathbf{k}}_{n-1}(n) = \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_{n-2}(n-1) \end{bmatrix} + \frac{e_{f,n-2}(n)}{(1-\alpha)\mathcal{E}_{f,n-2}(n-1)} \begin{bmatrix} 1 \\ -\mathbf{w}_{f,n-2}(n-1) \end{bmatrix}$$

$$\mathbf{w}_{f,n-1}(n) = \begin{bmatrix} \mathbf{w}_{f,n-2}(n-1) \\ \frac{e_{f,n-2}(n)}{u(1)} \end{bmatrix}$$

$e_{n-1}(n) = d(n) - \mathbf{w}_{n-1}^T(n-1)\mathbf{u}_{n-1}(n)$

$\epsilon_{n-1}(n) = \psi_{n-1}(n)e_{n-1}(n)$

if $(n < (M+1))$

$$\mathbf{w}_n(n) = \begin{bmatrix} \mathbf{w}_{n-1}(n-1) \\ \frac{e_{n-1}(n)}{u(1)} \end{bmatrix}$$

else

$\mathbf{w}_{n-1}(n) = \mathbf{w}_{n-1}(n-1) + \alpha\epsilon_{n-1}(n)\tilde{\mathbf{k}}_{n-1}(n)$

$\mathbf{w}_{b,n-1}(n) = \alpha u(1)\psi_{n-1}(n)\tilde{\mathbf{k}}_{n-1}(n)$

$\mathcal{E}_{b,n-1}(n) = \alpha\psi_{n-1}(n)u^2(1)$

## 4.3 Simulations

In order to test the algorithm, the results of a simulation example are provided in Fig. 4.1 where the adaptive filter was used to identify a 15th-order lowpass filter with cutoff frequency equal to $0.5\omega_s$. The input sequence was a white Gaussian noise of zero mean and unit variance. Noncorrelated noise with variance equal to 1E-6 was added to the desired response. Parameter $\alpha$ was chosen to be 0.05. In this example and in most other simulations. the rescue procedure was not necessary. Although this fact is not sufficient to assure stability, it most certainly accounts for the better tracking performance of the algorithm.

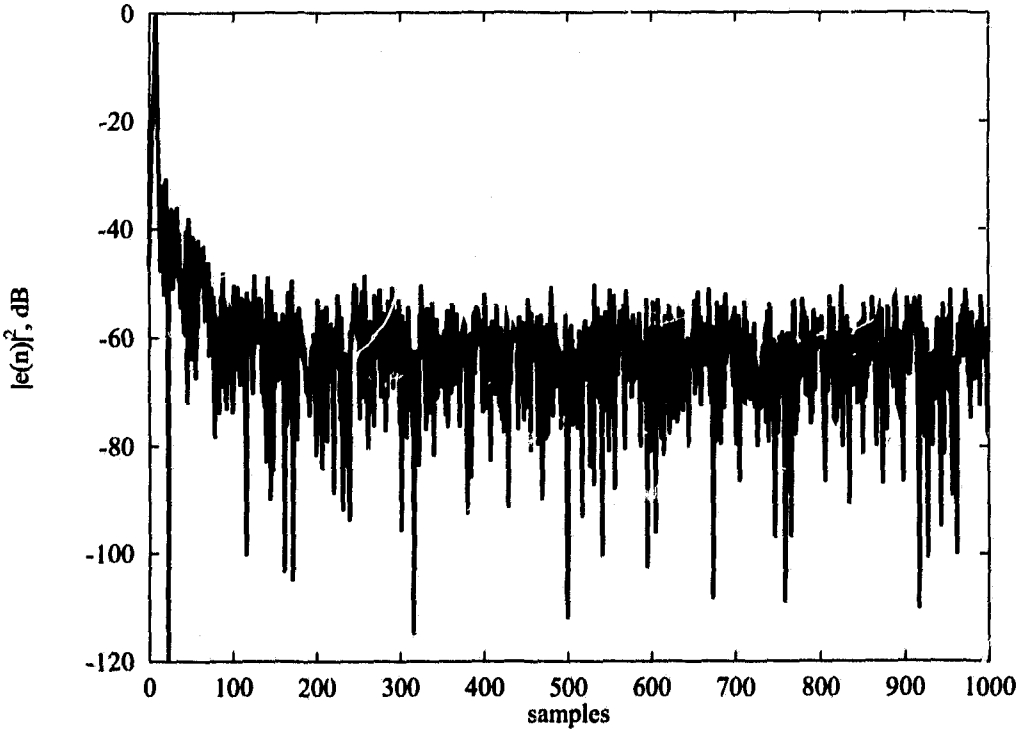



Figure 4.1: Learning curve for the fast LMSN algorithm.

## 4.4 Conclusions

The LMSN algorithm with variable convergence factor discussed in chapter 2 has been reported to perform very satisfactorily in several applications. In this chapter, we have shown that a fast implementation of this algorithm, i.e., one with order of complexity $\mathcal{O}(M)$, can be obtained based on the derivation of the FTF algorithm [32]. An exact initialization routine which updates the variables both in order and in time was also derived. We have shown that the variable convergence factor acts only upon the coefficients of the filter $\mathbf{w}(n)$. The backward and forward prediction-error filters and the Kalman gain vector minimize RLS-like objective functions and are updated using a constant convergence factor, as in the FTF algorithm. The close relationship between the LMSN and FTF algorithms facilitates the understanding, analysis, and stabilization of the proposed algorithm. For instance, the extension of stabilization schemes originally proposed for the fast RLS algorithm (see, e.g., [32], [68], [81], [82], and [83]) can be trivially extended for the variable-convergence-factor LMSN algorithm.

# Chapter 5

# Conclusions and Suggestions for Future Research

This chapter summarizes the conclusions of the thesis and highlights a number of outstanding problems and possible future research.

## 5.1 Conclusions

The demand for adaptive filters that can converge rapidly has certainly increased as they became a necessity in many state-of-the-art designs in communications, radar, and speech processing, to name just a few areas. Despite its high computational complexity, the LMSN algorithm has benefited from recent advancements in VLSI technology to become a powerful alternative to more rudimentary adaptation algorithms. However, as knowledge of the environment where the adaptive filter will be operating is sometimes restricted, variable-convergence-factor algorithms are certainly an interesting choice. Two variable-convergence-factor LMSN algorithms, which are versions of the orthogonalized-projection algorithm that employ different estimates of the input-signal autocorrelation matrix, $\hat{\mathbf{R}}(n)$, have been described and analyzed in chapter 2. Algorithm I uses a variable convergence factor when estimating $\mathbf{R}$ while algorithm II uses a constant convergence factor. Extensive simulations have shown that the two algorithms can perform in most situations as well as the RLS algorithm

without requiring as much knowledge of the signal statistics. The sensitivity of these algorithms with respect to their parameters is greatly reduced by the introduction of the variable convergence factor. The algorithms proved themselves especially suited for applications in nonstationary environments where algorithms with fixed convergence factors can very easily fail. The closed-form formulas obtained for the excess MSE due to measurement noise and nonstationarity have proved to be very accurate when compared to simulation results. These formulas can also accurately predict the excess MSE for other Newton-type algorithms that employ a similar convergence factor, independently of the estimate of the input-signal autocorrelation matrix used. A formula for the excess MSE due to quantization has been provided which suggests that even with fixed-point arithmetic the influence of quantization on the overall MSE is not significant. On the other hand, the effects of quantization on the estimate of $\mathbf{R}$ for both algorithms is a threat to their stability much in the same way as with the estimate of $\mathbf{R}$ employed by the RLS algorithm. A good solution to this problem is the implementation of the algorithm via the QR decomposition. Since $\hat{\mathbf{R}}^{-1}(n)$ is not explicitly computed in this case, the variable convergence factor had to be incorporated into the algorithm for achieving zero a posteriori output error while maintaining the characteristic robustness of QR-decomposition algorithms.

The RLS algorithm has been reported unstable for $\lambda = 1$ and stable for $\lambda < 1$, as long as accumulation and interaction of quantization errors are not considered [29] [68]. However, for fixed-point arithmetic, the RLS algorithm is known to be unstable even for $\lambda < 1$ [31]. For floating-point arithmetic, at least for highly correlated signals, simulations have shown that the RLS and LMSN algorithms can diverge. QR-decomposition-based algorithms present reasonably good performance with respect to robustness even for fixed-point arithmetic, but at the expense of complex and recursive equations that may require careful scaling. Furthermore, since the QR-LMSN and QR-RLS algorithms implement exactly the LMSN and RLS algorithms, respectively, problems must be expected when the input signal lacks persistent excitation. In chapter 3, a new QN adaptive filtering algorithm was introduced as a possible alternative to RLS and LMS-Newton algorithms when fast convergence is

necessary. The algorithm is based on fairly known classical optimization techniques, but resembles the LMS-Newton algorithm with variable convergence factor discussed in chapter 2. An essential feature of the algorithm is that it employs an estimate of the inverse of the autocorrelation matrix $\hat{\mathbf{R}}^{-1}(n)$ which remains positive definite and bounded for a bounded input signal; in addition, the requirement of persistent excitation can be relaxed. Since the positive definiteness of $\hat{\mathbf{R}}^{-1}(n)$ is a necessary condition for the stability of adaptation algorithms, the QN algorithm presented can operate in applications where other Newton-type algorithms as well as the RLS algorithm are likely to diverge. The issue of positive definiteness of the estimated input-signal autocorrelation matrix has been discussed from a qualitative point of view. We could show that the RLS algorithm cannot automatically reverse a situation when quantization errors are driving $\hat{\mathbf{R}}^{-1}(n)$ to become nonpositive definite. On the other hand, the study shows that the QN algorithm can reverse such a situation by adding rank-one matrices to the previous estimate of the inverse of the input-signal autocorrelation matrix.

The behavior and effect of important internal variables has been studied through a simplified model for the input signal, and simulations have shown that most of the conclusions can be extended to more general and practical signals. The QN algorithm does not require special rescuing procedures and a proof of internal stability has been provided that does not require averaging of the transition matrix [68]. The algorithm requires very little interference from the designer since no parameters need to be adjusted, and stability is guaranteed without bias. Furthermore, the algorithm was found to perform reliably with no scaling at all, provided that the desired and input signals are scaled to fit the processor's dynamic range.

Simulations for different and highly correlated input signals were carried out where floating as well as fixed-point arithmetic was employed, and in all of them the QN algorithm remained stable. It must be emphasized that several applications for adaptive filters require FIR filters with a small order where the utilization of a $\mathcal{O}(M^2)$ algorithm does not represent a significant increase in terms of computational complexity when compared to $\mathcal{O}(M)$ algorithms. This is especially true for $\mathcal{O}(M)$ algorithms

that employ stabilization routines requiring many divisions per iteration, as divisions are, in general, more costly than multiplications. In these situations, a robust $\mathcal{O}(M^2)$ algorithm, such as the QN algorithm, should undoubtedly be considered.

In chapter 4, we have shown that a fast implementation of the LMSN algorithm with variable convergence factor, i.e., one with order of complexity $\mathcal{O}(M)$, can be obtained based on the derivation of the FTF algorithm [32]. An exact initialization routine which updates the variables both in order and in time was derived. We have shown that the variable convergence factor acts only upon the coefficients of the filter $\mathbf{w}(n)$. The backward and forward prediction-error filters and the Kalman gain vector minimize RLS-like objective functions and are updated using a constant convergence factor, as in the FTF algorithm. The close relationship between the LMSN and FTF algorithms facilitates the understanding, analysis, and stabilization of the proposed algorithm. For instance, the extension of stabilization schemes originally proposed for the fast RLS algorithm (see, e.g., [32], [68], [81], [82], and [83]) can be trivially extended to the variable-convergence-factor LMSN algorithm.

## 5.2 Suggestions for Future Research

Although chapter 2 presents an extensive analysis of the LMSN algorithm with variable convergence factor, the coverage of the fast implementation of this algorithm in chapter 4 is far from complete. The redundancies in the equations that were so astutely studied by Slock and Kailath for the FTF algorithm (see [68]) open numerous possibilities for modifying and stabilizing the LMSN algorithm as well. In [68], these authors report a successful implementation of the stabilized FTF algorithm using 16-bit fixed-point arithmetic. It is expected from the conclusions of chapter 4 that similar results could be achieved with the fast LMSN algorithm with variable convergence factor. This possibility should be explored.

Chapters 1 and 2 have shown that any Newton-type algorithm that employs an estimate of the input-signal autocorrelation matrix that possesses some regularity properties can be associated with an objective function and can be implemented using the

QR decomposition. Therefore, the extension of the work to fast QR-decomposition LMSN algorithms seems an interesting means to achieve low computational complexity while maintaining the robustness of QR-decomposition algorithms and the good performance of the variable-convergence-factor LMSN algorithm in nonstationary environments. This should be pursued.

The derivation of a fast QN algorithm would be an important contribution since the results obtained with this algorithm have shown excellent performance in low-precision arithmetic. Although the estimated input-signal autocorrelation matrix used by the QN algorithm does not satisfy the regularity requirements necessary for a straightforward implementation with mathematical complexity $\mathcal{O}(M)$, modifications can certainly be introduced in the algorithm such that a $\mathcal{O}(M)$ version becomes possible.

In order to complete the analysis of the QN algorithm, an implementation-specific and arithmetic-specific analysis of quantization errors must be carried out to establish lower limits for the wordlength required for all the variables of the algorithm. Such a study would provide a basis for designing economical dedicated hardware in VLSI for implementing the QN algorithm.

# References


[1] M. L. R. de Campos and A. Antoniou, "Adaptive Filters," Chap. 6, *Handbook of Electrical Filters.* John Taylor ed., CRC Press, Inc., in press.

[2] K. F. Gauss, *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections — A Translation of Theoria Motus.* New York: Dover, 1963.

[3] J. J. Shynk, "Adaptive IIR filtering," *IEEE Signal Processing Magazine*, vol. 6, pp. 4–21, Apr. 1989.

[4] S. L. Netto, P. S. R. Diniz, and P. Agathoklis, "Adaptive IIR filtering algorithms for system identification: A general framework," *IEEE Trans. Education*, vol. 38, pp. 54–66, Feb. 1995.

[5] E. Hansler, "The hands-free telephone problem — an annotated bibliography," *Signal Processing*, vol. 27, pp. 259–271, June 1992.

[6] S. U. H. Qureshi, "Adaptive equalization," *Proc. IEEE*, vol. 73, pp. 1349–1385, Sept. 1985.

[7] C. S. Modlin and J. M. Cioffi, "A restructured decision feedback equalizer for facilitating the LMS algorithm," in *Proc. Twenty-Eighth Asilomar Conference on Signals, Systems, & Computers*, Pacific Grove, USA, pp. 1525–1529, 1994.

[8] S. Haykin, *Adaptive Filter Theory.* New Jersey: Prentice-Hall, Englewood-Cliffs, 2nd ed., 1991.

[9] T. Kailath, "A view of three decades of linear filtering theory," *IEEE Trans. Information Theory*, vol. IT-20, pp. 145–181, Mar. 1974.

[10] A. Papoulis, *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, 3rd ed., 1991.

[11] B. Widrow and M. E. Hoff Jr., "Adaptive switching circuits," in *IRE WESCON Conv. Rec., pt. 4*, pp. 96–104, 1960.

[12] B. Widrow and S. D. Stearns, *Adaptive Signal Processing.* New Jersey: Prentice-Hall, Englewood-Cliffs, 1985.

[13] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson Jr., "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proc. IEEE*, vol. 64, pp. 1151–1162, Aug. 1976.

[14] B. Widrow and E. Walach, "On the statistical efficiency of the LMS algorithm with nonstationary inputs," *IEEE Trans. Information Theory*, vol. IT-30, pp. 211–221, Mar. 1984.

[15] F. F. Yassa, "Optimality in the choice of the convergence factor for gradient-based adaptive algorithms," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-35, pp. 48–59, Jan. 1987.

[16] W. B. Mikhael, F. H. Wu, L. G. Kazovsky, G. S. Kang, and L. J. Fransen, "Adaptive filters with individual adaptation of parameters," *IEEE Trans. Circuits and Systems*, vol. CAS-33, pp. 677–685, July 1986.

[17] W. B. Mikhael and F. H. Wu, "A unified approach for generating optimum gradient FIR adaptive algorithms with time-varying convergence factors," *Journal of Circuits, Systems, and Computers*, vol. 1, no. 1, pp. 19–42, 1991.

[18] D. T. M. Slock, "On the convergence behaviour of the LMS and NLMS algorithms," in *Proc. 5th European Signal Processing Conference*, Barcelona, Spain, pp. 197–200, 1990.

[19] D. T. M. Slock, "On the convergence behaviour of the LMS and the normalized LMS algorithms," *IEEE Trans. Signal Processing*, vol. 41, pp. 2811–2825, Sept. 1993.

[20] B. Farhang-Boroujeny and S. Gazor, "Selection of orthonormal transforms for improving the performance of the transform domain normalised LMS algorithm," *IEE Proc.*, pt. F, vol. 139, pp. 327–335, Oct. 1992.

[21] D. F. Marshall, W. K. Jenkins, and J. J. Murphy, "The use of orthogonal transforms for improving performance of adaptive filters," *IEEE Trans. Circuits and Systems*, vol. 36, pp. 474–483, Apr. 1989.

[22] S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-31, pp. 609–615, June 1983.

[23] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[24] G. C. Goodwin and S. K. Sin, *Adaptive Filtering Prediction and Control.* New Jersey: Prentice-Hall, Englewood-Cliffs, 1984.

[25] P. S. R. Diniz, M. L. R. de Campos, and A. Antoniou, "Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor," *IEEE Trans. Signal Processing*, vol. 43, pp. 617–627, Mar. 1995.

[26] S. S. Reddi, "A time-domain adaptive algorithm for rapid convergence," *Proc. IEEE*, vol. 72, pp. 533–535, Apr. 1984.

[27] H. W. Sorenson, "Least-squares estimation: From Gauss to Kalman," *IEEE Spectrum*, pp. 63–68, July 1970.

[28] C. Caraiscus and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-32, pp. 34–41, Feb. 1984.

[29] S. Ljung and L. Ljung, "Error propagation properties of recursive least-squares adaptation algorithms," *Automatica*, vol. 21, no. 2, pp. 157–167, 1985.

[30] G. E. Bottomley, *Roundoff Error Problems and Solutions for Conventional Recursive Least Squares Filters.* Ph.D. thesis, North Carolina State University, Raleigh, USA, 1989.

[31] G. E. Bottomley and S. T. Alexander, "A novel approach for stabilizing recursive least squares filters," *IEEE Trans. Signal Processing*, vol. 39, pp. 1770–1779, Aug. 1991.

[32] J. M. Cioffi and T. Kailath, "Fast, recursive-least-squares transversal filters for adaptive filtering," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-32, pp. 304–337, Apr. 1984.

[33] G. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore: The Johns Hopkins University Press, 2nd ed., 1989.

[34] D. D. Falconer and L. Ljung, "Application of fast Kalman estimation to adaptive equalization," *IEEE Trans. Communications*, vol. COM-26, pp. 1439–1446, Oct. 1978.

[35] G. Carayannis, D. G. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least-squares filtering and prediction," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-31, pp. 1394–1402, Dec. 1983.

[36] M. Bellanger, "A survey of QR based fast least squares adaptive filters: From principles to realization," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Toronto, Canada, pp. 1833–1836, 1991.

[37] M. L. R. de Campos, P. S. R. Diniz, and A. Antoniou, "Performance of LMS-Newton adaptation algorithms with variable convergence factor in nonstationary environments," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, USA, pp. 392–395, 1993.

[38] M. L. R. de Campos, P. S. R. Diniz, and A. Antoniou, "A finite wordlength analysis of an LMS-Newton adaptive filtering algorithm," in *Proc. International Symposium on Circuits and Systems*, Chicago, USA, pp. 870–873, 1993.

[39] J. M. Cioffi, "Limited-precision effects in adaptive filtering," *IEEE Trans. Circuits and Systems*, vol. CAS-34, pp. 821–833, July 1987.

[40] M. L. R. de Campos, M. G. Siqueira, A. Antoniou, and A. N. Willson Jr., "A QR-decomposition LMS-Newton adaptive filtering algorithm with variable convergence factor," in *Proc. IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*, Victoria, Canada, pp. 350–353, 1993.

[41] M. L. R. de Campos and A. Antoniou, "A robust quasi-Newton adaptive filtering algorithm," in *Proc. International Symposium on Circuits and Systems*, London, England, pp. 229–232, 1994.

[42] M. L. R. de Campos and A. Antoniou, "Analysis of a new quasi-Newton adaptive filtering algorithm," submitted to *International Symposium on Circuits and Systems*, Atlanta, USA, 1996.

[43] M. L. R. de Campos and A. Antoniou, "A new quasi-Newton adaptive filtering algorithm," submitted to *IEEE Trans. Circuits and Systems.*

[44] M. L. R. de Campos and A. Antoniou, "Fast transversal LMS-Newton adaptive filtering algorithm with variable convergence factor," in *Proc. European Conference on Circuit Theory and Design*, Istanbul, Turkey, pp. 733–736, 1995.

[45] T. R. Fortescue, L. S. Kershenbaum, and B. E. Ydstie, "Implementation of self-tuning regulators with variable forgetting factors," *Automatica*, vol. 17, no. 6, pp. 831–835, 1981.

[46] B. Toplis and S. Pasupathy, "Tracking improvements in fast RLS algorithms using a variable forgetting factor," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 36, pp. 206–227, Feb. 1988.

[47] J. B. Evans and B. Liu, "Variable step size methods for the LMS adaptive algorithm," in *Proc. International Symposium on Circuits and Systems*, Philadelphia, USA, pp. 422–425, 1987.

[48] J. B. Evans, "A new variable step size method suitable for efficient VLSI implementation," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Toronto, Canada, pp. 2105–2108, 1991.

[49] T. J. Shan and T. Kailath, "Adaptive algorithms with an automatic gain control feature," *IEEE Trans. Circuits and Systems*, vol. 35, pp. 122–127, Jan. 1988.

[50] S. Karni and G. Zeng, "A new convergence factor for adaptive filters," *IEEE Trans. Circuits and Systems*, vol. 36, pp. 1011–1012, July 1989.

[51] S. Karni and G. Zeng, "Comments on 'adaptive algorithms with an automatic gain control feature'," *IEEE Trans. Circuits and Systems*, vol. 37, pp. 974–975, July 1990.

[52] T. J. Shan and T. Kailath, "Comments on "comments on 'adaptive algorithms with an automatic gain control feature' "," *IEEE Trans. Circuits and Systems*, vol. 37, p. 975, July 1990.

[53] V. J. Mathews and Z. Xie, "Stochastic gradient adaptive filters with gradient adaptive step sizes," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, USA, pp. 1385–1388, 1990.

[54] P. S. R. Diniz and L. W. P. Biscainho, "Optimal convergence factor for the LMS/Newton algorithm," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Toronto, Canada, pp. 2221–2224, 1991.

[55] P. S. R. Diniz and L. W. P. Biscainho, "Optimal variable step size for the LMS/Newton algorithm with application to subband adaptive filtering," *IEEE Trans. Signal Processing*, vol. 40, pp. 2825–2829, Nov. 1992.

[56] M. L. R. de Campos, "Análise de algoritmos tipo Newton para processamento adaptativo com passo variável," Master's thesis, in Portuguese, COPPE/UFRJ, Rio de Janeiro, Brazil, Dec. 1991.

[57] D. F. Marshall and W. K. Jenkins, "A fast quasi-Newton adaptive filtering algorithm," *IEEE Trans. Signal Processing*, vol. 40, pp. 1652–1662, July 1992.

[58] C. G. Samson and V. U. Reddy, "Fixed point error analysis of the normalized ladder algorithm," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-31, pp. 1177–1191, Oct. 1983.

[59] C. E. Bottomley and S. T. Alexander, "A theoretical basis for the divergence of conventional recursive least squares filters," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Glasgow, Scotland, pp. 908–911, 1989.

[60] J. E. Mazo, "On the independence theory of equalizer convergence," *The Bell System Technical Journal*, vol. 58, pp. 962–993, May–June 1979.

[61] E. Eleftheriou and D. D. Falconer, "Tracking properties and steady-state performance of RLS adaptive filter algorithms," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-34, pp. 1097–1110, Oct. 1986.

[62] T. Adali and S. H. Ardalan, "Steady state and convergence characteristics of the fixed point RLS algorithm," in *Proc. International Symposium on Circuits and Systems*, New Orleans, USA, pp. 788–791, 1990.

[63] T. Adali and S. H. Ardalan, "Fixed-point roundoff error analysis of the RLS algorithm with time-varying channels," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Toronto, Canada, pp. 1865–1868, 1991.

[64] S. H. Ardalan and S. T. Alexander, "Fixed-point roundoff error analysis of the exponentially windowed RLS algorithm for time-varying systems," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-35, pp. 770–783, June 1987.

[65] M. H. Verhaegen, "Improved understanding of the loss-of-symmetry phenomenon in the conventional Kalman filter," *IEEE Trans. Automatic Control*, vol. 34, pp. 331–333, Mar. 1989.

[66] M. H. Verhaegen, "Round-off error propagation in four generally-applicable, recursive, least-squares estimation schemes," *Automatica*, vol. 25, no. 3, pp. 437–444, 1989.

[67] A. Antoniou, *Digital Filters: Analysis, Design, and Applications.* New York: McGraw-Hill, 2nd ed., 1993.

[68] D. T. M. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. Signal Processing*, vol. 39, pp. 92–114, Jan. 1991.

[69] R. Fletcher, *Practical Methods of Optimization.* New York: Wiley, 2nd ed., 1987.

[70] R. Fletcher, "A new approach to variable metric algorithms," *Computer Journal*, vol. 13, pp. 317–322, 1970.

[71] W. C. Davidon, "Variance algorithm for minimization," *Computer Journal*, vol. 10, pp. 406–410, 1968.

[72] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms," *J. Inst. Maths Applics.*, no. 6, pp. 222–231, 1970.

[73] J. H. Wilkinson, *The Algebraic Eigenvalue Problem.* New York: Oxford University Press, 1965.

[74] P. Voltz and F. Kozin, "Almost-sure convergence of adaptive algorithms by projections," *IEEE Trans. Automatic Control*, vol. 34, pp. 325–327, Mar. 1989.

[75] T. Kailath, *Linear Systems.* New Jersey: Prentice-Hall, Englewood-Cliffs, 1980.

[76] F. R. Gantmacher, *The Theory of Matrices*, vol. II. New York: Chelsea Publishing Company, 1959.

[77] G. Kubin, "Stabilization of the RLS algorithm in the absence of persistent excitation," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, New York, USA, pp. 1369–1372, 1988.

[78] G. Kubin, *Recursive Algorithms for Adaptive Transversal Filters: Optimality and Time Variance.* Ph.D. thesis, University of Technology, Vienna, Austria, June 1990.

[79] S. Subramanian, M. L. R. de Campos, D. J. Shpak, and A. Antoniou, "A robust quasi-Newton adaptive filtering algorithm for indoor wireless communications," in *Proc. International Conference on Signal Processing Applications and Technology*, Dallas, USA, pp. 328–333, 1994.

[80] S. T. Alexander, "Fast adaptive filters: A geometrical approach," *IEEE Signal Processing Magazine*, vol. 3, pp. 18–28, Oct. 1986.

[81] D. T. Slock and T. Kailath, "Numerically stable fast recursive least-squares transversal filters," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, New York, USA, pp. 1365–1368, 1988.

[82] J.-L. Botto, "Stabilization of fast recursive least-squares transversal filters for adaptive filtering," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Dallas, USA, pp. 403–407, 1987.

[83] A. Benallal and A. Gilloire, "A new method to stabilize fast RLS algorithms based on a first-order model of the propagation of numerical errors," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, New York, USA, pp. 1373–1376, 1988.

# VITA

Surname: de Campos Given Names: Marcello Luiz Rodrigues

Place of Birth: Niterói, Rio de Janeiro, Brazil

Educational Institutions Attended:

| | |
|---|---|
| COPPE/Federal University of Rio de Janeiro | 1990 to 1991 |
| Federal University of Rio de Janeiro | 1985 to 1990 |

Degrees Awarded:

| | | |
|---|---|---|
| M.Sc. | COPPE/Federal University of Rio de Janeiro | 1991 |
| Elec.Eng. (cum laude) | Federal University of Rio de Janeiro | 1990 |

Publications:


1. S. L. Netto, M. L. R. de Campos, and P. S. R. Diniz, "PACPAS: Um pacote de apoio para processamento adaptativo de sinais," in *Anais do 8o. Congresso Brasileiro de Automática*, in Portuguese, Belem, Brazil, pp. 159–163, 1990.

2. L. P. Calôba, M. L. R. de Campos, and A. C. M. de Queiroz, "Low sensitivity biquads based on doubly loaded LC filters," in *Proc. 34th Midwest Symposium on Circuits and Systems*, Monterey, USA, pp. 1065–1068, 1991.

3. M. L. R. de Campos, P. S. R. Diniz, and A. Antoniou, "Performance of LMS-Newton adaptation algorithms with variable convergence factor in nonstationary environments," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, USA, pp. 392–395, 1993.

4. M. L. R. de Campos, P. S. R. Diniz, and A. Antoniou, "A finite wordlength analysis of an LMS-Newton adaptive filtering algorithm," in *Proc. International Symposium on Circuits and Systems*, Chicago, USA, pp. 870–873, 1993.

5. M. L. R. de Campos, M. G. Siqueira, A. Antoniou, and A. N. Willson Jr., "A QR-decomposition LMS-Newton adaptive filtering algorithm with variable convergence factor," in *Proc. IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*, Victoria, Canada, pp. 350–353, 1993.

6. M. L. R. de Campos and A. Antoniou, "A robust quasi-Newton adaptive filtering algorithm," in *Proc. International Symposium on Circuits and Systems*, London, England, pp. 229–232, 1994.

7. S. Subramanian, M. L. R. de Campos, D. J. Shpak, and A. Antoniou, "A robust quasi-Newton adaptive filtering algorithm for indoor wireless communications," in *Proc. International Conference on Signal Processing Applications and Technology*, Dallas, USA, pp. 328–333, 1994.

8. P. S. R. Diniz, M. L. R. de Campos, and A. Antoniou, "Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor," *IEEE Trans. Signal Processing*, vol. 43, pp. 617–627, Mar. 1995.

9. S. Subramanian, M. L. R. de Campos, D. Shpak, and A. Antoniou, "Investigation of the autocorrelation matrix in the presence of correlated co-channel interference in indoor wireless systems," in *Proc. IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*, Victoria, Canada, pp. 165–168, 1995.

10. S. L. Netto, M. L. R. de Campos, A. Antoniou, and P. Agathoklis, "A stable RLC parameterization of the direct structure for adaptive IIR filters," in *Proc. 38th Midwest Symposium on Circuits and Systems*, Rio de Janeiro, Brazil, 1995.

11. M. L. R. de Campos and A. Antoniou, "Fast transversal LMS-Newton adaptive filtering algorithm with variable convergence factor," in *Proc. European Conference on Circuit Theory and Design*, Istanbul, Turkey, pp.733–736, 1995.

12. M. L. R. de Campos and A. Antoniou, "Adaptive Filters," Chap. 6, *Handbook of Electric Filters*, John Taylor ed., CRC Press, Inc., in press.

13. M. L. R. de Campos and A. Antoniou, "Analysis of a new quasi-Newton adaptive filtering algorithm," submitted to *International Symposium on Circuits and Systems*, Atlanta, USA, 1996.

14. M. L. R. de Campos and A. Antoniou, "A new quasi-Newton adaptive filtering algorithm," submitted to *IEEE Trans. Circuits and Systems.*

# PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis (or dissertation) to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis/Dissertation:
Development and Analysis of Fast and Robust Newton-Type Adaptation Algorithms

Author:

Marcello Luiz Rodrigues de Campos
December 19, 1995.