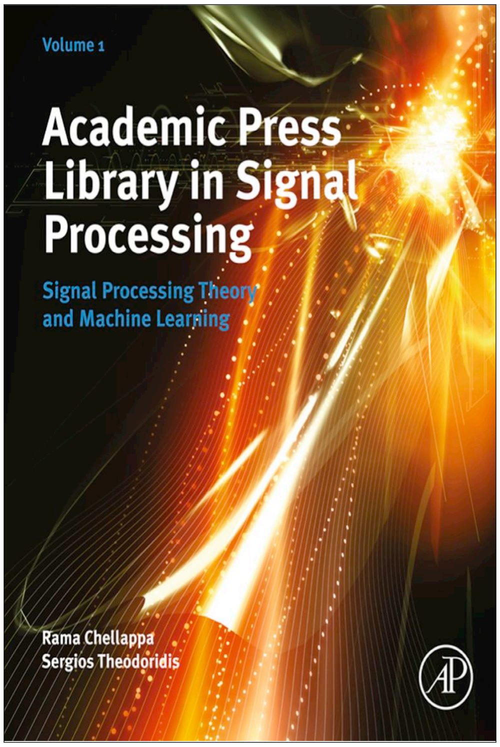

Volume 1
Academic Press Library in Signal Processing
Signal Processing Theory and Machine Learning
Rama Chellappa
Sergios Theodoridis

From Vítor H. Nascimento and Magno T. M. Silva, Adaptive Filters. In: Rama Chellappa, Sergios Theodoridis, editors, Academic Press Library in Signal Processing. Vol 1, Signal Processing Theory and Machine Learning, Chennai: Academic Press, 2014, p. 619-761.
ISBN: 978-0-12-396502-8
© Copyright 2014 Elsevier Ltd
Academic Press.

CHAPTER

Adaptive Filters

12

Vítor H. Nascimento and Magno T. M. Silva

Department of Electronic Systems Engineering, University of São Paulo, São Paulo, SP, Brazil

1.12.1 Introduction

Adaptive filters are employed in situations in which the environment is constantly changing, so that a fixed system would not have adequate performance. As they are usually applied in real-time applications, they must be based on algorithms that require a small number of computations per input sample.

These algorithms can be understood in two complementary ways. The most straightforward way follows directly from their name: an adaptive filter uses information from the environment and from the very signal it is processing to change itself, so as to optimally perform its task. The information from the environment may be sensed in real time (in the form of a so-called desired signal), or may be provided *a priori*, in the form of previous knowledge of the statistical properties of the input signal (as in blind equalization).

On the other hand, we can think of an adaptive filter also as an algorithm to separate a mixture of two signals. The filter must have some information about the signals to be able to separate them; usually this is given in the form of a reference signal, related to only one of the two terms in the mixture. The filter has then two outputs, corresponding to each signal in the mixture (see Figure 12.1). As a byproduct of separating the signals, the reference signal is processed in useful ways. This way of viewing an adaptive filter is very useful, particularly when one is learning the subject.

In the following sections we give an introduction to adaptive filters, covering from basic principles to the most important recent developments. Since the adaptive literature is vast, and the topic is still an active area of research, we were not able to treat all interesting topics. Along the text, we reference important textbooks, classic papers, and the most promising recent literature. Among the many very good textbooks on the subject, four of the most popular are [1–4].

We start with an application example, in the next section. In it, we show how an adaptive filter is applied to cancel acoustic echo in hands-free telephony. Along the text we return frequently to this example, to illustrate new concepts and methods. The example is completed by an overview of how adaptive filters work, in which we use only deterministic arguments and concepts from basic linear systems theory, in Section 1.12.1.2. This overview shows many of the compromises that arise in adaptive filter design, without the more complicated math. Section 1.12.1.3 gives a small glimpse of the wide variety of applications of adaptive filters, describing a few other examples. Of course, in order to fully understand the behavior of adaptive filters, one must use estimation theory and stochastic processes,

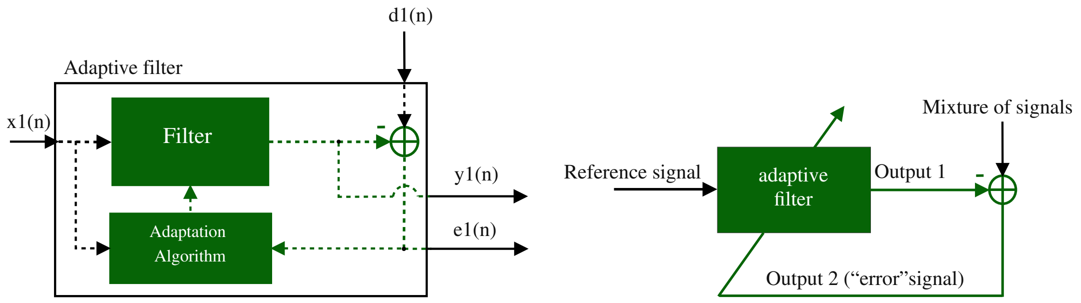


FIGURE 12.1

Inputs and outputs of an adaptive filter. Left: detailed diagram showing inputs, outputs and internal variable. Right: simplified diagram.

which is the subject of Sections 1.12.2 and 1.12.3. The remaining sections are devoted to extensions to the basic algorithms and recent results.

Adaptive filter theory brings together results from several fields: signal processing, matrix analysis, control and systems theory, stochastic processes, and optimization. We tried to provide short reviews for most of the necessary material in separate links ("boxes"), that the reader may follow if needed.

1.12.1.1 Motivation—acoustic echo cancellation

Suppose that you are talking with a person who uses a hands-free telephone inside a car. Let us assume that the telephone does not have any acoustic echo canceler. In this case, the sound of your voice will reach the loudspeaker, propagate inside the car, and suffer reflections every time it encounters the walls of the car. Part of the sound can be absorbed by the walls, but part of it is reflected, resulting in an echo signal. This signal is fed back to the microphone and you will hear your own voice with delay and attenuation, as shown in Figure 12.2.

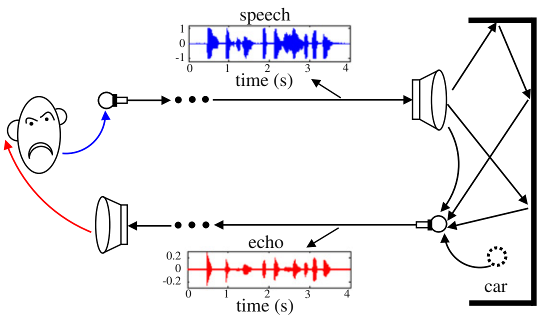


FIGURE 12.2

Echo in hands-free telephony. Click speech.wav to listen to a speech signal and echo.wav to listen to the corresponding echo signal.

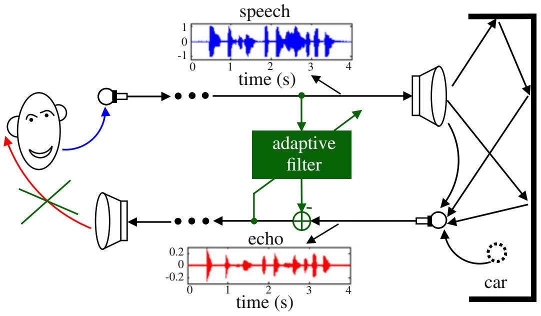


FIGURE 12.3

Acoustic echo cancellation using an adaptive filter.

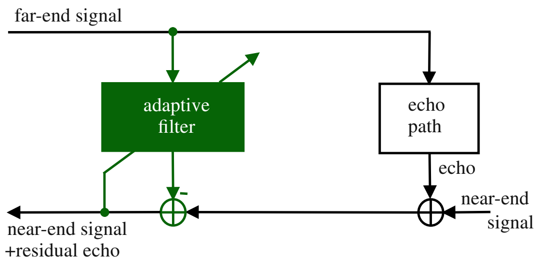


FIGURE 12.4

General echo canceler configuration.

The echo signal depends on the acoustic characteristics of each car and also on the location of the loudspeaker and microphone. Furthermore, the acoustic characteristics can change during a phone call, since the driver may open or close a window, for example. In order to follow the variations of the environment, the acoustic echo must be canceled by an adaptive filter, which performs on-line updating of its parameters through an algorithm. Figure 12.3 shows the insertion of an adaptive filter in the system of Figure 12.2 to cancel the echo signal. In this configuration, the adaptive filter must identify the echo path and provide at its output an estimate of the echo signal. Thus, a synthesized version of the echo is subtracted from the signal picked up by the car microphone. In an ideal scenario, the resulting signal will be free of echo and will contain only the signal of interest. The general set-up of an echo canceler is shown in Figure 12.4. The *near-end* signal (near with respect to the echo canceler) may be simply noise when the person inside the car is in silence, as considered in Figures 12.2 and 12.3, or it can be a signal of interest, that is, the voice of the user in the car.

You most likely have already heard the output of an adaptive echo canceler. However, you probably did not pay attention to the initial period, when you can hear the adaptive filter learning the echo impulse response, if you listen carefully. We illustrate this through a simulation, using a measured impulse response of 256 samples (sampled at 8 kHz), and an echo signal produced from a recorded

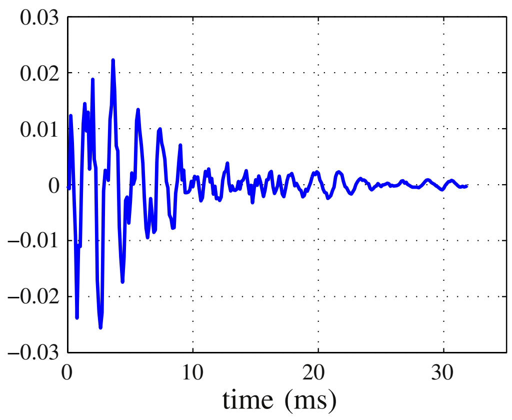


FIGURE 12.5

Measured impulse response (256 samples).

voice signal. The measured impulse response is shown in Figure 12.5. In the first example, the near-end signal is low-power noise (more specifically, white Gaussian noise with zero mean and variance $\sigma^2 = 10^{-4}$). Listen to `e_nlms.wav`, and pay attention to the beginning. You will notice that the voice (the echo) is fading away, while the adaptive filter is learning the echo impulse response. This file is the output of an echo canceler adapted using the *normalized least-mean squares algorithm* (*NLMS*), which is described in Section 1.12.3.2.

Listen now to `e_lsl.wav` in this case the filter was adapted using the modified *error-feedback least-squares lattice algorithm* (*EF-LSL*) [5], which is a low-cost and stable version of the *recursive least-squares algorithm* (*RLS*) described in Section 1.12.3.3. Now you will notice that the echo fades away much faster. This fast convergence is characteristic of the EF-LSL algorithm, and is of course a very desirable property. On the other hand, the NLMS algorithm is simpler to implement and more robust against imperfections in the implementation. An adaptive filtering algorithm must satisfy several requirements, such as convergence rate, tracking capability, noise rejection, computational complexity, and robustness. These requirements are conflicting, so that there is no best algorithm that will outperform all the others in every situation. The many algorithms available in the literature are different compromises between these requirements [1–3].

Figure 12.6 shows 4 s of the echo signal, prior to cancellation (bottom) and a measure of the quality of echo cancellation, the so-called *echo return loss enhancement* (*ERLE*) (top). The ERLE is defined as the ratio

$$\text{ERLE} = \frac{\text{Power of original echo}}{\text{Power of residual echo}}.$$

The higher the ERLE, the better the performance of the canceler. The ERLE drops when the original echo is small (there is nothing to cancel!), and increases again when the echo is strong. The figure shows that EF-LSL performs better than NLMS (but at a higher computational cost).

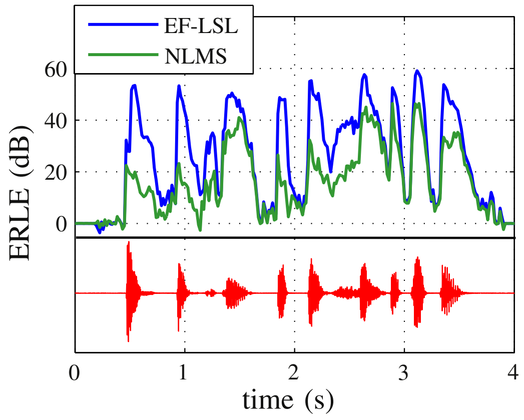


FIGURE 12.6

Echo return loss enhancement for NLMS ($\tilde{\mu} = 0.5$, $\delta = 10^{-5}$, $M = 256$) and EF-LSL ($\lambda = 0.999$, $\delta = 10^{-5}$) algorithms. Bottom trace: echo signal prior to cancellation. The meaning of the several parameters is explained in Sections 1.12.3.1, 1.12.3.2, and 1.12.3.3.

Assuming now that the near-end signal is another recorded voice signal (Click DT_echo.wav to listen to this signal), we applied again the EF-LSL algorithm to cancel the echo. This case simulates a person talking inside the car. Since the speech of this person is not correlated to the far-end signal, the output of the adaptive filter converges again to an estimate of the echo signal. Thus, a synthesized version of the echo is subtracted from the signal picked up by the car microphone and the resulting signal converges to the speech of the person inside the car. This result can be verified by listening to the file `e_lslDT.wav`.

This is a typical example of use of an adaptive filter. The environment (the echo impulse response) changes constantly, requiring a constant re-tuning of the canceling filter. Although the echo itself is not known, we have access to a reference signal (the clean speech of the far-end user). This is the signal that is processed to obtain an estimate of the echo. Finally, the signal captured by the near-end microphone (the so-called *desired* signal) is a mixture of two signals: the echo, plus the near-end speech (or ambient noise, when the near-end user is silent). The task of the adaptive filter can be seen as to separate these two signals.

We should mention that the name desired signal is somewhat misleading, although widespread in the literature. Usually we are interested in separating the "desired" signal into two parts, one of which is of no interest at all!

We proceed now to show how an algorithm can perform this task, modeling all signals as periodic to simplify the arguments.

1.12.1.2 A quick tour of adaptive filtering

Before embarking in a detailed explanation of adaptive filtering and its theory, it is a good idea to present the main ideas in a simple setting, to help the reader create intuition. In this section, we introduce the least-mean squares (LMS) algorithm and some of its properties using only deterministic arguments and basic tools from linear systems theory.

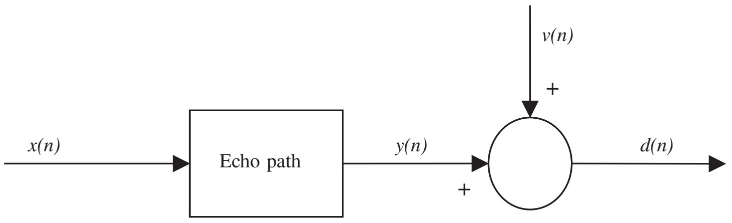


FIGURE 12.7

A model for the echo.

1.12.1.2.1 *Posing the problem*

Consider again the echo-cancellation problem, seen in the previous section (see Figure 12.7). Let $y(n)$ represent the echo, $v(n)$ represent the voice of the near-end user, and $x(n)$ the voice of the far-end user (the reference). $d(n)$ represents the signal that would be received by the far-end user without an echo canceler (the mixture, or desired signal).

The echo path represents the changes that the far-end signal suffers when it goes through the digital-to-analog (D/A) converter, the loudspeaker, the air path between the loudspeaker and the microphone (including all reflections), the microphone itself and its amplifier, and the analog-to-digital (A/D) converter. The microphone signal $d(n)$ will always be a mixture of the echo $y(n)$ and an unrelated signal $v(n)$, which is composed of the near-end speech plus noise. Our goal is to remove $y(n)$ from $d(n)$. The challenge is that we have no means to measure $y(n)$ directly, and the echo path is constantly changing.

1.12.1.2.2 *Measuring how far we are from the solution*

How is the problem solved, then? Since we cannot measure $y(n)$, the solution is to estimate it from the measurable signals $x(n)$ and $d(n)$. This is done as follows: imagine for now that all signals are periodic, so they can be decomposed as Fourier series

$$x(n) = \sum_{k=1}^{K_0} A_k \cos(k\omega_0 n + \varphi_k), \quad v(n) = \sum_{\ell=1}^{K_1} B_\ell \cos(\ell\omega_1 n + \theta_\ell), \tag{12.1}$$

for certain amplitudes A_k and B_ℓ, phases φ_k and θ_ℓ, and frequencies ω_0 and ω_1. The highest frequencies appearing in $x(n)$ and $v(n)$ must satisfy $K_0\omega_0 < \pi$, $K_1\omega_1 < \pi$, since we are dealing with sampled signals (Nyquist criterion). We assume for now that the echo path is fixed (not changing), and can be modeled by an unknown linear transfer function $H(z)$. In this case, $y(n)$ would also be a periodic sequence with fundamental frequency ω_0,

$$y(n) = \sum_{k=1}^{K_0} C_k \cos(k\omega_0 n + \psi_k),$$

with $C_k e^{j\psi_k} = H(e^{jk\omega_0})A_k e^{j\varphi_k}$. The signal picked by the microphone is then

$$d(n) = y(n) + v(n) = \sum_{k=1}^{K_0} C_k \cos(k\omega_0 n + \psi_k) + \sum_{\ell=1}^{K_1} B_\ell \cos(\ell\omega_1 n + \theta_\ell).$$

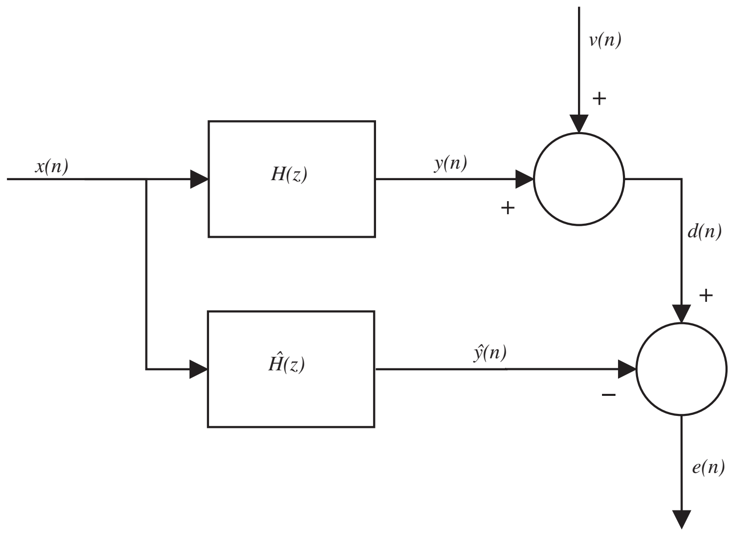


FIGURE 12.8

Echo canceler.

Since we know $x(n)$, if we had a good approximation $\widehat{H}(z)$ to $H(z)$, we could easily obtain an approximation $\hat{y}(n)$ to $y(n)$ and subtract it from $d(n)$, as in Figure 12.8.

How could we find $\widehat{H}(z)$ in real time? Recall that only $d(n)$, $x(n)$, $\hat{y}(n)$, and $e(n)$ can be observed. For example, how could we know that we have the exact filter, that is, that $\widehat{H}(z) = H(z)$, by looking only at these signals? To answer this question, take a closer look at $e(n)$. Let the output of $\hat{H}(z)$ be

$$\hat{y}(n) = \sum_{k=1}^{k_0} \hat{C}_k \cos(k\omega_0 n + \hat{\psi}_k), \quad \text{and thus}$$

$$\begin{aligned} e(n) &= d(n) - \hat{y}(n) = y(n) - \hat{y}(n) + v(n) \\ &= \sum_{k=1}^{K_0} [C_k \cos(k\omega_0 n + \psi_k) - \hat{C}_k \cos(k\omega_0 n + \hat{\psi}_k)] \\ &\quad + \sum_{\ell=1}^{K_1} B_\ell \cos(\ell\omega_1 n + \theta_\ell) \\ &= \sum_{k=1}^{K_0} \widetilde{C}_k \cos(k\omega_0 n + \tilde{\psi}_k) + \sum_{\ell=1}^{K_1} B_\ell \cos(\ell\omega_1 n + \theta_\ell), \end{aligned} \tag{12.2}$$

where $\widetilde{C}_k e^{j\tilde{\psi}_k} = C_k e^{j\psi_k} - \hat{C}_k e^{j\hat{\psi}_k}$, $k = 1, \ldots, K_0$.

Assuming that the cosines in $x(n)$ and $v(n)$ have no frequencies in common, the simplest approach is to measure the average power of $e(n)$. Indeed, if all frequencies $k\omega_0$ and $\ell\omega_1$ appearing in $e(n)$ are different from one another, then the average power of $e(n)$ is

$$P = \sum_{k=1}^{K_0} \frac{\widetilde{C}_k^2}{2} + \sum_{\ell=1}^{K_1} \frac{B_\ell^2}{2}. \tag{12.3}$$

If $\hat{y}(n) = y(n)$, then P is at its minimum,

$$P_{\mathrm{o}} = \min_{\widehat{H}(z)} P = \sum_{\ell=1}^{K_1} \frac{B_\ell^2}{2},$$

which is the average power of $v(n)$. In this case, $e(n)$ is at its optimum: $e(n) = e_{\mathrm{o}}(n) = v(n)$, and the echo is completely canceled.

It is important to see that this works because the two signals we want to separate, $y(n)$ and $v(n)$, are *uncorrelated*, that is, they are such that

$$\lim_{N\to\infty} \frac{1}{N+1} \sum_{-N/2}^{N/2} y(n)v(n) = \lim_{N\to\infty} \frac{1}{N+1} \sum_{-N/2}^{N/2} y(n)e_{\mathrm{o}}(n) = 0. \tag{12.4}$$

This property is known as the *orthogonality condition*. A form of orthogonality condition will appear in general whenever one tries to minimize a quadratic function, as is the case here. This is discussed in more detail in Section 1.12.2.

The average power P could then be used as a measure of how good is our approximation $\hat{y}(n)$. The important point is that it is very easy to find a good approximation to P. It suffices to low-pass filter $e^2(n)$, for example,

$$\widehat{P}(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^2(n-k) \tag{12.5}$$

is a good approximation if the window length N is large enough.

The adaptive filtering problem then becomes an optimization problem, with the particularity that the cost function that is being minimized is not known exactly, but only through an approximation, as in (12.5). In the following sections, we discuss in detail the consequences of this fact.

It is also important to remark that the average error power is not the only possible choice for cost function. Although it is the most popular choice for a number of reasons, in some applications other choices are more adequate. Even posing the adaptive filtering problem as an optimization problem is not the only alternative, as shown by recent methods based on projections on convex sets [6].

1.12.1.2.3 Choosing a structure for the filter

Our approximation $\hat{y}(n)$ will be built by minimizing the estimated average error power (12.5) as a function of the echo path model $\widehat{H}(z)$, that is, our estimated echo path will be the solution of

$$\widehat{H}_{\mathrm{o}}(z) = \arg\min_{\widehat{H}(z)} \widehat{P}(n). \tag{12.6}$$

Keep in mind that we are looking for a real-time way of solving (12.6), since $\widehat{P}(n)$ can only be obtained by measuring $e(n)$ for a certain amount of time. We must then be able to implement the current approximation $\widehat{H}(z)$, so that $\hat{y}(n)$ and $e(n)$ may be computed.

This will impose practical restrictions on the kind of function that $\widehat{H}(z)$ may be: since memory and processing power are limited, we must choose beforehand a structure for $\widehat{H}(z)$. Memory and processing power will impose a constraint on the maximum order the filter may have. In addition, in order to write the code for the filter, we must decide if it will depend on past outputs (IIR filters) or only on past inputs (FIR filters). These choices must be made based on some knowledge of the kind of echo path that our system is likely to encounter.

To explain how these choices can be made, let us simplify the problem a little more and restrict the far-end signal $x(n)$ to a simple sinusoid (i.e., assume that $K_0 = 1$). In this case,

$$y(n) = C_1 \cos(\omega_0 n + \varphi_1),$$

with $C_1 e^{j\psi_1} = H(e^{j\omega_0}) A_1 e^{j\varphi_1}$. Therefore, $\widehat{H}(z)$ must satisfy

$$\left[\widehat{H}(e^{j\omega}) = H(e^{j\omega})\right]_{\omega=\omega_0} \tag{12.7}$$

only for $\omega = \omega_0$—the value of $\widehat{H}(e^{j\omega})$ for other frequencies is irrelevant, since the input $x(n)$ has only one frequency. Expanding (12.7), we obtain

$$\mathrm{Re}\{\widehat{H}(e^{j\omega_0})\} = \mathrm{Re}\{H(e^{j\omega_0})\}, \quad \mathrm{Im}\{\widehat{H}(e^{j\omega_0})\} = \mathrm{Im}\{H(e^{j\omega_0})\}. \tag{12.8}$$

The optimum $\widehat{H}(z)$ is defined through two conditions. Therefore an FIR filter with just two coefficients would be able to satisfy both conditions for any value of $H(e^{j\omega_0})$, so we could define

$$\widehat{H}(z) = w_0 + w_1 z^{-1},$$

and the values of w_0 and w_1 would be chosen to solve (12.6). Note that the argument generalizes for $K_0 > 1$: in general, if $x(n)$ has K_0 harmonics, we could choose $\widehat{H}(z)$ as an FIR filter with length $2K_0$, two coefficients per input harmonic.

This choice is usually not so simple: in practice, we would not know K_0, and in the presence of noise (as we shall see later), a filter with fewer coefficients might give better performance, even though it would not completely cancel the echo. The structure of the filter also affects the dynamic behavior of the adaptive filter, so simply looking at equations such as (12.7) does not tell the whole story. Choosing the structure for $\widehat{H}(z)$ is one of the most difficult steps in adaptive filter design. In general, the designer must test different options, initially perhaps using recorded signals, but ultimately building a prototype and performing some tests.

1.12.1.2.4 *Searching for the solution*

Returning to our problem, assume then that we have decided that an FIR filter with length M is adequate to model the echo path, that is,

$$\widehat{H}(z) = \sum_{k=0}^{M-1} w_k z^{-k}.$$

Our minimization problem (12.6) now reduces to finding the coefficients $w_0, \ldots, w_{M-1}$ that solve

$$\min_{w_0,\ldots,w_{M-1}} \widehat{P}(n). \tag{12.9}$$

Since $\widehat{P}(n)$ must be computed from measurements, we must use an iterative method to solve (12.9). Many algorithms could be used, as long as they depend only on measurable quantities ($x(n)$, $d(n)$, $\hat{y}(n)$, and $e(n)$). We will use the steepest descent algorithm (also known as gradient algorithm) as an example now. Later we show other methods that also could be used. If you are not familiar with the gradient algorithm, see Box 1 for an introduction.

The cost function in (12.9) is

$$\widehat{P}(n) = \frac{1}{N}\sum_{k=0}^{N-1} e^2(n-k) = \frac{1}{N}\sum_{k=0}^{N-1}[d(n-k) - \hat{y}(n-k)]^2.$$

We need to rewrite this equation so that the steepest descent algorithm can be applied. Recall also that now the filter coefficients will change, so define the vectors

$$\boldsymbol{w}(n) = [w_0(n) \quad w_1(n) \quad \cdots \quad w_{M-1}(n)]^T,$$

and

$$\boldsymbol{x}(n) = [x(n) \quad x(n-1) \quad \cdots \quad x(n-M+1)]^T,$$

where $(\cdot)^T$ denotes transposition. At each instant, we have $\hat{y}(n) = \boldsymbol{w}^T(n)\boldsymbol{x}(n)$, and our cost function becomes

$$\widehat{P}(n) = \frac{1}{N}\sum_{k=0}^{N-1}\left[d(n-k) - \boldsymbol{w}^T(n-k)\boldsymbol{x}(n-k)\right]^2, \tag{12.10}$$

which depends on $\boldsymbol{w}(n), \ldots, \boldsymbol{w}(n-N+1)$. In order to apply the steepest descent algorithm, let us keep the coefficient vector $\boldsymbol{w}(n)$ constant during the evaluation of $\widehat{P}$, as follows. Starting from an initial condition $\boldsymbol{w}(0)$, compute for $n = 0, N, 2N, \ldots$ (The notation is explained in Boxes 2 and 6.)

$$\boldsymbol{w}(n+N) = \boldsymbol{w}(n) - \alpha \left.\frac{\partial \widehat{P}(n+N-1)}{\partial \boldsymbol{w}^T}\right|_{\boldsymbol{w}=\boldsymbol{w}(n)}, \tag{12.11}$$

where α is a positive constant, and $\boldsymbol{w}(n+N-1) = \boldsymbol{w}(n+N-2) = \cdots = \boldsymbol{w}(n)$. Our cost function now depends on only one $\boldsymbol{w}(n)$ (compare with (12.10)):

$$\begin{aligned}\widehat{P}(n+N-1) &= \frac{1}{N}\sum_{k=0}^{N-1}\left[d(n+N-1-k) - \boldsymbol{w}^T(n)\boldsymbol{x}(n+N-1-k)\right]^2 \\ &= \frac{1}{N}\sum_{k=0}^{N-1}\left[d(n+k) - \boldsymbol{w}^T(n)\boldsymbol{x}(n+k)\right]^2, n = 0, N, 2N, \ldots \end{aligned} \tag{12.12}$$

We now need to evaluate the gradient of $\widehat{P}$. Expanding the expression above, we obtain

$$\widehat{P}(n+N-1) = \frac{1}{N}\sum_{k=0}^{N-1}\left[d(n+k) - \sum_{\ell=0}^{M-1} w_\ell(n)x(n+k-\ell)\right]^2,$$

so

$$\frac{\partial \widehat{P}(n-N+1)}{\partial w_m(n)} = -\frac{2}{N}\sum_{k=0}^{N-1}\left[d(n+k) - \boldsymbol{w}^T(n)\boldsymbol{x}(n+k)\right]x(n+k-m)$$
$$= -\frac{2}{N}\sum_{k=0}^{N-1} e(n+k)x(n+k-m),$$

and

$$\frac{\partial \widehat{P}(n-N+1)}{\partial \boldsymbol{w}^T} = -\frac{2}{N}\sum_{k=0}^{N-1} e(n+k)\boldsymbol{x}(n+k), n = 0, N, 2N, \ldots \tag{12.13}$$

As we needed, this gradient depends only on measurable quantities, $e(n+k)$ and $\boldsymbol{x}(n+k)$, for $k = 0, \ldots, N-1$. Our algorithm for updating the filter coefficients then becomes

$$\boldsymbol{w}(n+N) = \boldsymbol{w}(n) + \mu\frac{1}{N}\sum_{k=0}^{N-1} e(n+k)\boldsymbol{x}(n+k), n = 0, N, 2N, \ldots, \tag{12.14}$$

where we introduced the overall step-size $\mu = 2\alpha$.

We still must choose μ and N. The choice of μ is more complicated and will be treated in Section 1.12.1.2.5. The value usually employed for N may be a surprise: in almost all cases, one uses $N = 1$, resulting in the so-called *least-mean squares* (LMS) algorithm

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e(n)\boldsymbol{x}(n), \tag{12.15}$$

proposed initially by Widrow and Hoff in 1960 [7] (Widrow [8] describes the history of the creation of the LMS algorithm). The question is, how can this work, if no average is being used for estimating the error power? An intuitive answer is not complicated: assume that μ is a very small number so that $\mu = \mu_0/N$ for a large N. In this case, we can approximate $e(n+k)$ from (12.15) as follows.

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{\mu_0}{N}e(n)\boldsymbol{x}(n) \approx \boldsymbol{w}(n), \quad \text{if } N \text{ is large.}$$

Since $\boldsymbol{w}(n+1) \approx \boldsymbol{w}(n)$, we have $e(n+1) = d(n+1) - \boldsymbol{w}^T(n+1)\boldsymbol{x}(n+1) \approx d(n+1) - \boldsymbol{w}^T(n)\boldsymbol{x}(n+1)$. Therefore, we could approximate

$$e(n+k) \approx d(n+k) - \boldsymbol{w}^T(n)\boldsymbol{x}(n+k), \quad k = 0, \ldots, N-1,$$

so N steps of the LMS recursion (12.15) would result

$$\boldsymbol{w}(n+N) \approx \boldsymbol{w}(n) + \frac{\mu_0}{N}\sum_{k=0}^{N-1}\left[d(n+k) - \boldsymbol{w}^T(n)\boldsymbol{x}(n+k)\right]\boldsymbol{x}(n+k),$$

just what would be obtained from (12.14). The conclusion is that, although there is no explicit average being taken in the LMS algorithm (12.15), the algorithm in fact computes an implicit, approximate average if the step-size is small. This is exactly what happens, as can be seen in the animations in

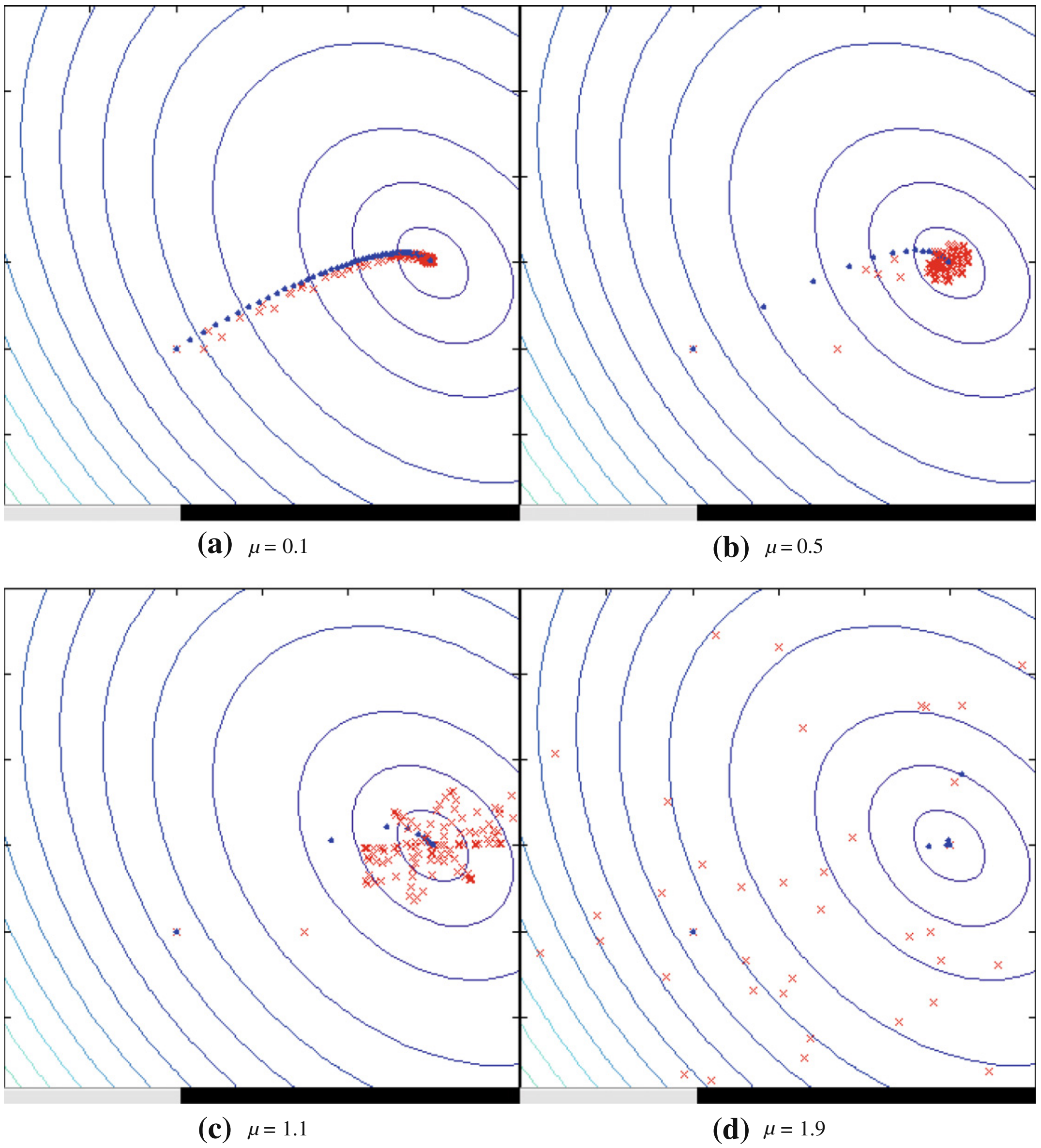


FIGURE 12.9

LMS algorithm for echo cancelation with sinusoidal input. Click LMS video files to see animations on your browser.

Figure 12.9. These simulations were prepared with

$$x(n) = \cos(0.4\pi n + \varphi_1), \quad v(n) = 0.2\cos(0.2\pi n + \theta_1),$$

where φ_1 and θ_1 were chosen randomly in the interval $[0, 2\pi)$. Several step-sizes were used. The estimates computed with the LMS algorithm are marked by the crosses (red in the web version). The initial condition is at the left, and the theoretical optimum is at the right end of each figure. In the simulations, the true echo path was modeled by the fixed filter

$$H(z) = 1.5 + 0.5z^{-1}.$$

For comparison, we also plotted the estimates that would be obtained by the gradient algorithm using the exact value of the error power at each instant. These estimates are marked with small dots (blue in the web version).

Note that when a small step-size is used, such as $\mu = 0.1$ in Figure 12.9a, the LMS filter stays close to the dots obtained assuming exact knowledge of the error power. However, as the step-size increases, the LMS estimates move farther away from the dots. Although convergence is faster, the LMS estimates do not reach the optimum and stay there: instead, they hover around the optimum. For larger step-sizes, the LMS estimates can go quite far from the optimum (Figure 12.9b and c). Finally, if the step-size is too large, the algorithm will diverge, that is, the filter coefficients grow without bounds (Figure 12.9d).

1.12.1.2.5 *Tradeoff between speed and precision*

The animations in Figure 12.9 illustrate an important problem in adaptive filters: even if you could magically chose as initial conditions the exact optimum solutions, the adaptive filter coefficients would not stay there! This happens because the filter does not know the exact value of the cost function it is trying to minimize (in this case, the error power): since $\widehat{P}(n)$ is an approximation, noise (and the near-end signal in our echo cancellation example) would make the estimated gradient non-zero, and the filter would wander away from the optimum solution. This effect keeps the minimum error power obtainable using the adaptive filter always a little higher than the optimum value. The difference between the (theoretical, unattainable without perfect information) optimum and the actual error power is known as *excess mean-square error* (EMSE), and the ratio between the EMSE and the optimum error is known as *misadjustment* (see Eqs. (12.171) and (12.174)).

For small step-sizes the misadjustment is small, since the LMS estimates stay close to the estimates that would be obtained with exact knowledge of the error power. However, a small step-size also means slow convergence. This trade-off exists in all adaptive filter algorithms, and has been an intense topic of research: many algorithms have been proposed to allow faster convergence without increasing the misadjustment. We will see some of them in the next sections.

The misadjustment is central to evaluate the performance of an adaptive filter. In fact, if we want to eliminate the echo from the near-end signal, we would like that after convergence, $e(n) \approx v(n)$. This is indeed what happens when the step-size is small (see Figure 12.10a). However, when the step-size is increased, although the algorithm converges more quickly, the performance after convergence is not very good, because of the wandering of the filter weights around the optimum (Figure 12.10b–d—in the last case, for $\mu = 1.9$, the algorithm is diverging.).

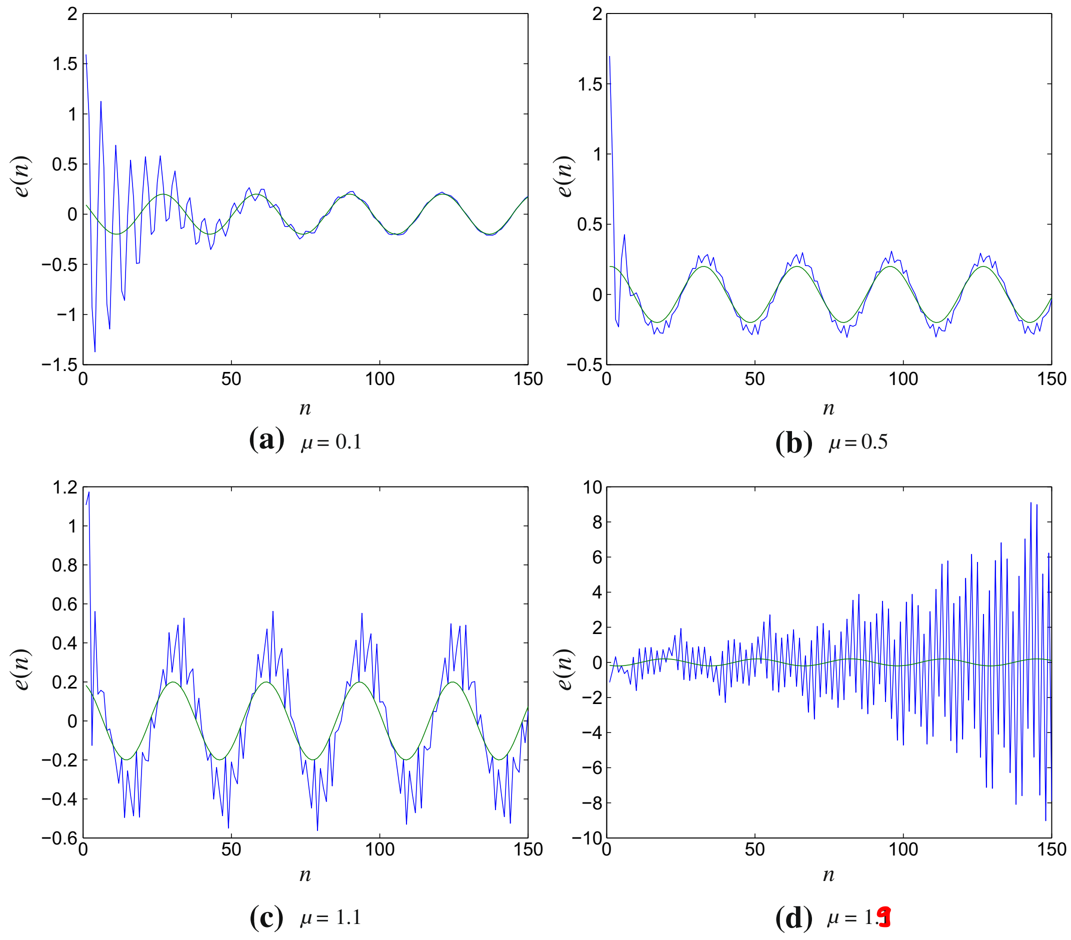


FIGURE 12.10

Compromise between convergence rate and misadjustment in LMS. Plots of $e(n)x_n$ for several values of μ.

We will stop this example here for the moment, and move forward to a description of adaptive filters using probability theory and stochastic processes. We will return to it during discussions of stability and model order selection.

The important message from this section is that an adaptive filter is able to separate two signals in a mixture by minimizing a well-chosen cost function. The minimization must be made iteratively, since the true value of the cost function is not known: only an approximation to it may be computed at each step. When the cost function is quadratic, as in the example shown in this section, the components of the mixture are separated such that they are in some sense orthogonal to each other.

Box 1: Steepest descent algorithm

Given a cost function

$$J(\boldsymbol{w}) = J(w_0, \ldots, w_{M-1}),$$

with $\boldsymbol{w} = [w_0, \ldots, w_{M-1}]^T$ ($(\cdot)^T$ denotes transposition), we want to find the minimum of J, that is, we want to find $\boldsymbol{w}_\text{o}$ such that

$$\boldsymbol{w}_\text{o} = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w}).$$

The solution can be found using the gradient of $J(\boldsymbol{w})$, which is defined by

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{\partial J}{\partial \boldsymbol{w}^T} = \begin{bmatrix} \frac{\partial J}{\partial w_0} \\ \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_{M-1}} \end{bmatrix}.$$

See Box 2 for a brief explanation about gradients and Hessians, that is, derivatives of functions of several variables.
The notation $\partial J / \partial \boldsymbol{w}^T$ is most convenient when we deal with complex variables, as in Box 6. We use it for real variables for consistency.

Since the gradient always points to the direction in which $J(\boldsymbol{w})$ increases most quickly, the steepest descent algorithm searches iteratively for the minimum of $J(\boldsymbol{w})$ taking at each iteration a small step in the opposite direction, i.e., towards $-\nabla_{\boldsymbol{w}} J(\boldsymbol{w})$:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \frac{\mu}{2} \nabla_{\boldsymbol{w}} J(\boldsymbol{w}(n)), \tag{12.16}$$

where μ is a step-size, i.e., a constant controlling the speed of the algorithm. As we shall see, this constant should not be too small (or the algorithm will converge too slowly) neither too large (or the recursion will diverge).

As an example, consider the quadratic cost function with $M = 2$

$$J(\boldsymbol{w}) = 1 - 2\boldsymbol{w}^T \boldsymbol{r} + \boldsymbol{w}^T \boldsymbol{R} \boldsymbol{w}, \tag{12.17}$$

with

$$\boldsymbol{r} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{R} = \begin{bmatrix} \frac{4}{3} & -\frac{2}{3} \\ -\frac{2}{3} & \frac{4}{3} \end{bmatrix}. \tag{12.18}$$

The gradient of $J(\boldsymbol{w})$ is

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{\partial J}{\partial \boldsymbol{w}^T} = -2\boldsymbol{r} + 2\boldsymbol{R}\boldsymbol{w}. \tag{12.19}$$

Of course, for this example we can find the optimum $\boldsymbol{w}_\text{o}$ by equating the gradient to zero:

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}_\text{o}) = \boldsymbol{0} \Longrightarrow \boldsymbol{R}\boldsymbol{w}_\text{o} = \boldsymbol{r} \Longrightarrow \boldsymbol{w}_\text{o} = \boldsymbol{R}^{-1}\boldsymbol{r} = \begin{bmatrix} 0.9877 \\ 0.4902 \end{bmatrix}. \tag{12.20}$$

Note that in this case the *Hessian* of $J(\boldsymbol{w})$ (the matrix of second derivatives, see Box 2) is simply $\nabla^2_{\boldsymbol{w}} J = 2\boldsymbol{R}$. As $\boldsymbol{R}$ is symmetric with positive eigenvalues ($\lambda_1 = 2$ and $\lambda_2 = \frac{2}{3}$), it is positive-definite, and we can conclude that $\boldsymbol{w}_o$ is indeed a minimum of $J(\boldsymbol{w})$ (See Boxes 2 and 3. Box 3 lists several useful properties of matrices.)

Even though in this case we could compute the optimum solution through (12.20), we will apply the gradient algorithm with the intention of understanding how it works and which are its main limitations. From (12.16) and (12.17), we obtain the recursion

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu(\boldsymbol{r} - \boldsymbol{R}\boldsymbol{w}(n)). \tag{12.21}$$

The user must choose an initial condition $\boldsymbol{w}(0)$.

In Figure 12.11 we plot the evolution of the approximations computed through (12.21) against the level sets (i.e., curves of constant value) of the cost function (12.17). Different choices of step-size μ are shown. In Figure 12.11a, a small step-size is used. The algorithm converges to the correct solution, but slowly. In Figure 12.11b, we used a larger step-size—convergence is now faster, but the algorithm still needs several iterations to reach the solution. If we try to increase the step-size even further, convergence at first becomes slower again, with the algorithm oscillating towards the solution (Figure 12.11c). For even larger step-sizes, such as that in Figure 12.11d, the algorithm diverges: the filter coefficients get farther and farther away from the solution.

It is important to find the maximum step-size for which the algorithm (12.21) remains stable. For this, we need concepts from linear systems and linear algebra, in particular eigenvalues, their relation to stability of linear systems, and the fact that symmetric matrices always have real eigenvalues (see Box 3).

The range of allowed step-sizes is found as follows. Rewrite (12.21) as

$$\boldsymbol{w}(n+1) = \left[\boldsymbol{I} - \mu\boldsymbol{R}\right]\boldsymbol{w}(n) + \mu\boldsymbol{r},$$

which is a linear recursion in state-space form ($\boldsymbol{I}$ is the identity matrix).

Linear systems theory [9] tells us that this recursion converges as long as the largest eigenvalue of $\boldsymbol{A} = \boldsymbol{I} - \mu\boldsymbol{R}$ has absolute value less than one. The eigenvalues of $\boldsymbol{A}$ are the roots of

$$\det(\beta\boldsymbol{I} - (\boldsymbol{I} - \mu\boldsymbol{R})) = \det((\beta - 1)\boldsymbol{I} + \mu\boldsymbol{R}) = -\det((1-\beta)\boldsymbol{I} - \mu\boldsymbol{R}) = 0.$$

Let $1 - \beta = \nu$. Then β is an eigenvalue of $\boldsymbol{A}$ if and only if $\nu = 1 - \beta$ is an eigenvalue of $\mu\boldsymbol{R}$. Therefore, if we denote by λ_i the eigenvalues of $\boldsymbol{R}$, the stability condition is

$$-1 < 1 - \mu\lambda_i < 1 \;\forall i \Leftrightarrow 0 < \mu < \frac{2}{\lambda_i} \;\forall i \Leftrightarrow 0 < \mu < \frac{2}{\max\{\lambda_i\}}.$$

The stability condition for our example is thus $0 < \mu < 1$, in agreement with what we saw in Figure 12.11.

The gradient algorithm leads to relatively simple adaptive filtering algorithms; however, it has an important drawback. As you can see in Figure 12.11a, the gradient does not point directly to the direction of the optimum. This effect is heightened when the level sets of the cost function are very elongated, as in Figure 12.12. This case corresponds to a quadratic function in which $\boldsymbol{R}$ has one eigenvalue much smaller than the other. In this example, we replaced the matrix $\boldsymbol{R}$ in (12.18) by

$$\boldsymbol{R}' = \frac{1}{9}\begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix}.$$

This matrix has eigenvalues $\lambda'_1 = 5$ and $\lambda'_2 = \frac{5}{9}$.

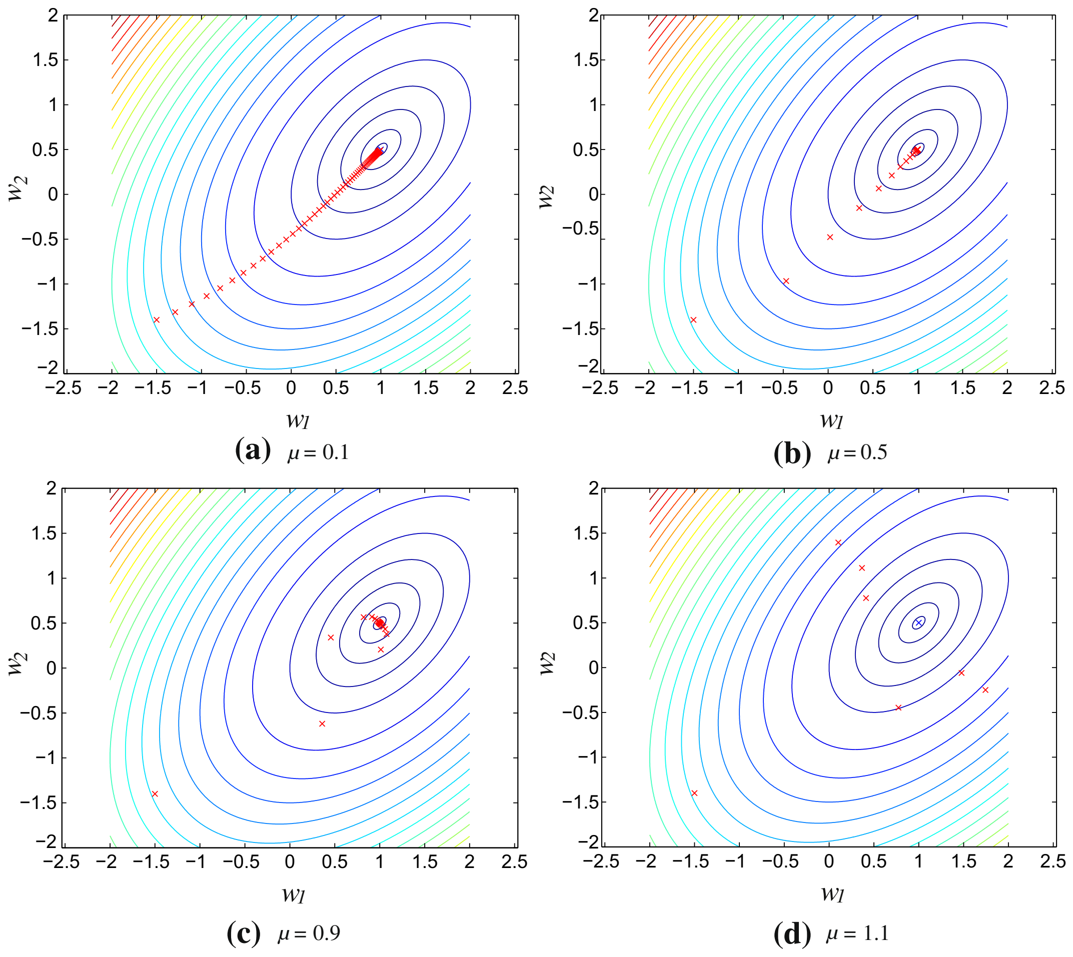


FIGURE 12.11

Performance of the steepest descent algorithm for different step-sizes. The crosses (x) represent the successive approximations $\boldsymbol{w}(n)$, plotted against the level curves of (12.17). The initial condition is the point in the lower left.

Let us see why this is so. Recall from Box 3 that for every symmetric matrix there exists an orthogonal matrix $\boldsymbol{U}$ (that is, $\boldsymbol{U}^{-1} = \boldsymbol{U}^T$) such that

$$\boldsymbol{U}^T\boldsymbol{R}\boldsymbol{U} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \triangleq \boldsymbol{\Lambda}, \tag{12.22}$$

where we defined the diagonal eigenvalue matrix $\boldsymbol{\Lambda}$.

Let us apply a change of coordinates to (12.19). The optimum solution to (12.17) is

$$\boldsymbol{w}_{\mathrm{o}} = \boldsymbol{R}^{-1}\boldsymbol{r}.$$

Our first change of coordinates is to replace $\boldsymbol{w}(n)$ by $\tilde{\boldsymbol{w}}(n) = \boldsymbol{w}_{\mathrm{o}} - \boldsymbol{w}(n)$ in (12.19). Subtract (12.19) from $\boldsymbol{w}_{\mathrm{o}}$ and replace $\boldsymbol{r}$ in (12.19) by $\boldsymbol{r} = \boldsymbol{R}\boldsymbol{w}_{\mathrm{o}}$ to obtain

$$\tilde{\boldsymbol{w}}(n+1) = \tilde{\boldsymbol{w}}(n) - \mu\boldsymbol{R}\tilde{\boldsymbol{w}}(n) = \left[\boldsymbol{I} - \mu\boldsymbol{R}\right]\tilde{\boldsymbol{w}}(n). \tag{12.23}$$

Next, multiply this recursion from both sides by $\boldsymbol{U}^T = \boldsymbol{U}^{-1}$

$$\boldsymbol{U}^T\tilde{\boldsymbol{w}}(n+1) = \boldsymbol{U}^T\left[\boldsymbol{I} - \mu\boldsymbol{R}\right]\underbrace{\boldsymbol{U}\boldsymbol{U}^T}_{=\boldsymbol{I}}\tilde{\boldsymbol{w}}(n) = \left[\boldsymbol{I} - \mu\boldsymbol{\Lambda}\right]\boldsymbol{U}^T\tilde{\boldsymbol{w}}(n).$$

Defining $\bar{\boldsymbol{w}}(n) = \boldsymbol{U}^T\tilde{\boldsymbol{w}}(n)$, we have rewritten the gradient equation in a new set of coordinates, such that now the equations are uncoupled. Given that $\boldsymbol{\Lambda}$ is diagonal, the recursion is simply

$$\begin{bmatrix} \bar{w}_1(n+1) \\ \bar{w}_2(n+1) \end{bmatrix} = \begin{bmatrix} (1-\mu\lambda_1)\bar{w}_1(n) \\ (1-\mu\lambda_2)\bar{w}_2(n) \end{bmatrix}. \tag{12.24}$$

Note that, as long as $|1-\mu\lambda_i| < 1$ for $i = 1, 2$, both entries of $\bar{\boldsymbol{w}}(n)$ will converge to zero, and consequently $\boldsymbol{w}(n)$ will converge to $\boldsymbol{w}_{\mathrm{o}}$.

The stability condition for this recursion is

$$|1-\mu\lambda_i| < 1, i = 1, 2 \Rightarrow \mu < \frac{2}{\max\{\lambda_1, \lambda_2\}}.$$

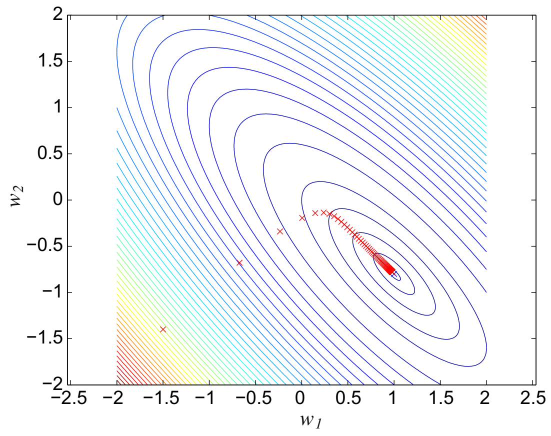


FIGURE 12.12

Performance of the steepest descent algorithm for a problem with large ratio of eigenvalues. The crosses (×) represent the successive approximations $\boldsymbol{w}(n)$, plotted against the level curves of (12.17). The initial condition is the point in the lower left. Step-size $\mu = 0.1$.

When one of the eigenvalues is much larger than the other, say, when $\lambda_1 \gg \lambda_2$, the rate of convergence for the direction relative to the smaller eigenvalue becomes

$$1 > 1 - \mu\lambda_2 > 1 - 2\frac{\lambda_{\min}}{\lambda_{\max}} \approx 1,$$

and even if one of the coordinates in $\bar{\boldsymbol{w}}(n)$ converges quickly to zero, the other will converge very slowly. This is what we saw in Figure 12.12. The ratio $\lambda_{\max}/\lambda_{\min}$ is known as the *eigenvalue spread* of a matrix.

In general, the gradient algorithm converges slowly when the eigenvalue spread of the Hessian is large. One way of solving this problem is to use a different optimization algorithm, such as the *Newton* or *quasi-Newton* algorithms, which use the inverse of the Hessian matrix, or an approximation to it, to improve the search direction used by the gradient algorithm.

Although these algorithms converge very quickly, they require more computational power, compared to the gradient algorithm. We will see more about them when we describe the RLS (*recursive least-squares*) algorithm in Section 1.12.3.3.

For example, for the quadratic cost-function (12.17), the Hessian is equal to $2\boldsymbol{R}$. If we had a good approximation $\hat{\boldsymbol{R}}$ to $\boldsymbol{R}$, we could use the recursion

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\hat{\boldsymbol{R}}^{-1}[\boldsymbol{r} - \boldsymbol{R}\boldsymbol{w}(n)]. \tag{12.25}$$

This class of algorithms is known as *quasi-Newton* (if $\hat{\boldsymbol{R}}^{-1}$ is an approximation) or *Newton* (if $\hat{\boldsymbol{R}}^{-1}$ is exact). In the Newton case, we would have

$$\boldsymbol{w}(n+1) = (1-\mu)\boldsymbol{w}(n) + \boldsymbol{w}_{\rm o},$$

that is, the algorithm moves at each step precisely in the direction of the optimum solution. Of course, this is not a very interesting algorithm for a quadratic cost function as in this example (the optimum solution is used in the recursion!) However, this method is very useful when the cost function is not quadratic and in adaptive filtering, when the cost function is not known exactly.

Box 2: Gradients

Consider a function of several variables $J(\boldsymbol{w})$, where

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix}.$$

Its gradient is defined by

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{\partial J}{\partial \boldsymbol{w}^T} = \begin{bmatrix} \frac{\partial J}{\partial w_0} \\ \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_{M-1}} \end{bmatrix}.$$

The real value of the notation $\partial J/\partial \boldsymbol{w}^T$ will only become apparent when working with functions of complex variables, as shown in Box 6. We also define, for consistency,

$$\frac{\partial J}{\partial \boldsymbol{w}} = \begin{bmatrix} \frac{\partial J}{\partial w_0} & \frac{\partial J}{\partial w_1} & \cdots & \frac{\partial J}{\partial w_{M-1}} \end{bmatrix}.$$

As an example, consider the quadratic cost function

$$J(\boldsymbol{w}) = b - 2\boldsymbol{w}^T\boldsymbol{r} + \boldsymbol{w}^T\boldsymbol{R}\boldsymbol{w}. \tag{12.26}$$

The gradient of $J(\boldsymbol{w})$ is (verify!)

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{\partial J}{\partial \boldsymbol{w}^T} = -2\boldsymbol{r} + 2\boldsymbol{R}\boldsymbol{w}. \tag{12.27}$$

If we use the gradient to find the minimum of $J(\boldsymbol{w})$, it would be necessary also to check the second-order derivative, to make sure the solution is not a maximum or a saddle point. The second order derivative of a function of several variables is a matrix, the *Hessian*. It is defined by

$$\nabla^2_{\boldsymbol{w}} J = \frac{\partial^2 J}{\partial \boldsymbol{w}\, \partial \boldsymbol{w}^T} = \begin{bmatrix} \frac{\partial}{\partial \boldsymbol{w}}\left\{\frac{\partial J}{\partial w_0}\right\} \\ \vdots \\ \frac{\partial}{\partial \boldsymbol{w}}\left\{\frac{\partial J}{\partial w_{M-1}}\right\} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 J}{\partial w_0^2} & \cdots & \frac{\partial^2 J}{\partial w_0\, \partial w_{M-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial w_{M-1}\, \partial w_0} & \cdots & \frac{\partial^2 J}{\partial w_{M-1}^2} \end{bmatrix}. \tag{12.28}$$

Note that for the quadratic cost-function (12.26), the Hessian is equal to $\boldsymbol{R} + \boldsymbol{R}^T$. It is equal to $2\boldsymbol{R}$ if $\boldsymbol{R}$ is symmetric, that is, if $\boldsymbol{R}^T = \boldsymbol{R}$. This will usually be the case here. Note that $\boldsymbol{R} + \boldsymbol{R}^T$ is always symmetric: in fact, since for well-behaved functions J it holds that

$$\frac{\partial^2 J}{\partial w_i\, \partial w_j} = \frac{\partial^2 J}{\partial w_j\, \partial w_i},$$

the Hessian is usually symmetric.

Assume that we want to find the minimum of $J(\boldsymbol{w})$. The first-order conditions are then

$$\left.\frac{\partial J}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}_o} = \boldsymbol{0}.$$

The solution $\boldsymbol{w}_o$ is a minimum of $J(\boldsymbol{w})$ if the Hessian at $\boldsymbol{w}_o$ is a positive semi-definite matrix, that is, if

$$\boldsymbol{x}^T \nabla^2_{\boldsymbol{w}} J(\boldsymbol{w}_o)\boldsymbol{x} \geq 0$$

for all directions $\boldsymbol{x}$.

Box 3: Useful results from Matrix Analysis

Here we list without proof a few useful results from matrix analysis. Detailed explanations can be found, for example, in [10–13].

Fact 1 (Traces). The *trace* of a matrix $\boldsymbol{A} \in \mathcal{C}^{M\times M}$ is the sum of its diagonal elements:

$$\operatorname{Tr}(\boldsymbol{A}) = \sum_{i=1}^{M} a_{ii}, \tag{12.29}$$

in which the a_{ii} are the entries of $\boldsymbol{A}$. For any two matrices $\boldsymbol{B} \in \mathcal{C}^{M\times N}$ and $\boldsymbol{C} \in \mathcal{C}^{N\times M}$, it holds that

$$\operatorname{Tr}(\boldsymbol{BC}) = \operatorname{Tr}(\boldsymbol{CB}). \tag{12.30}$$

In addition, if λ_i, $i = 1, \ldots, M$ are the eigenvalues of a square matrix $\boldsymbol{A} \in \mathcal{C}^{M\times M}$, then

$$\operatorname{Tr}(\boldsymbol{A}) = \sum_{i=1}^{M} \lambda_i. \tag{12.31}$$

Fact 2 (Singular matrices). A matrix $\boldsymbol{A} \in \mathcal{C}^{M\times M}$ is singular if and only if there exists a nonzero vector $\boldsymbol{u}$ such that $\boldsymbol{Au} = \boldsymbol{0}$.

The *null space*, or *kernel* $N(\boldsymbol{A})$ of a matrix $\boldsymbol{A} \in \mathcal{C}^{M\times N}$ is the set of vectors $\boldsymbol{u} \in \mathcal{C}^N$ such that $\boldsymbol{Au} = \boldsymbol{0}$. A square matrix $\boldsymbol{A} \in \mathcal{C}^{M\times M}$ is nonsingular if, and only if, $N(\boldsymbol{A}) = \{\boldsymbol{0}\}$, that is, if its null space contains only the null vector.

Note that, if $\boldsymbol{A} = \sum_{k=1}^{K} \boldsymbol{u}_k \boldsymbol{v}_k^H$, with $\boldsymbol{u}_k, \boldsymbol{v}_k \in \mathcal{C}^M$, then $\boldsymbol{A}$ cannot be invertible if $K < M$. This is because when $K < M$, there always exists a nonzero vector $\boldsymbol{x} \in \mathcal{C}^M$ such that $\boldsymbol{v}_k^H \boldsymbol{x} = 0$ for $1 \le k \le K$, and thus $\boldsymbol{Ax} = \boldsymbol{0}$ with $\boldsymbol{x} \neq \boldsymbol{0}$.

Fact 3 (Inverses of block matrices). Assume the square matrix $\boldsymbol{A}$ is partitioned such that

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix},$$

with $\boldsymbol{A}_{11}$ and $\boldsymbol{A}_{22}$ square. If $\boldsymbol{A}$ and $\boldsymbol{A}_{11}$ are invertible (nonsingular), then

$$\boldsymbol{A}^{-1} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_{11}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Delta}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1} & \boldsymbol{I} \end{bmatrix},$$

where $\boldsymbol{\Delta} = \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$ is the *Schur complement* of $\boldsymbol{A}_{11}$ in $\boldsymbol{A}$. If $\boldsymbol{A}_{22}$ and $\boldsymbol{A}$ are invertible, then

$$\boldsymbol{A}^{-1} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{A}_{22}^{-1}\boldsymbol{A}_{21} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}'^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{22}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{A}_{12}\boldsymbol{A}_{22}^{-1} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix},$$

where $\boldsymbol{\Delta}' = \boldsymbol{A}_{11} - \boldsymbol{A}_{12}\boldsymbol{A}_{22}^{-1}\boldsymbol{A}_{21}$.

Fact 4 (Matrix inversion Lemma). Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two nonsingular $M \times M$ matrices, $\boldsymbol{D} \in \mathcal{C}^{K\times K}$ be also nonsingular, and $\boldsymbol{C}, \boldsymbol{E} \in \mathcal{C}^{M\times K}$ be such that

$$\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{C}\boldsymbol{D}\boldsymbol{E}^H. \qquad (12.32)$$

The inverse of $\boldsymbol{A}$ is then given by

$$\boldsymbol{A}^{-1} = \boldsymbol{B}^{-1} - \boldsymbol{B}^{-1}\boldsymbol{C}(\boldsymbol{D}^{-1} + \boldsymbol{E}^H\boldsymbol{B}^{-1}\boldsymbol{C})^{-1}\boldsymbol{E}^H\boldsymbol{B}^{-1}. \qquad (12.33)$$

The lemma is most useful when $K \ll M$. In particular when $K = 1$, $\boldsymbol{D} + \boldsymbol{E}^H\boldsymbol{B}^{-1}\boldsymbol{C}$ is a scalar, and we have

$$\boldsymbol{A}^{-1} = \boldsymbol{B}^{-1} - \frac{\boldsymbol{B}^{-1}\boldsymbol{C}\boldsymbol{E}^H\boldsymbol{B}^{-1}}{\boldsymbol{D}^{-1} + \boldsymbol{E}^H\boldsymbol{B}^{-1}\boldsymbol{C}}.$$

Fact 5 (Symmetric matrices). Let $\boldsymbol{A} \in \mathcal{C}^{M\times M}$ be *Hermitian Symmetric* (or simply *Hermitian*), that is, $\boldsymbol{A}^H = \left(\boldsymbol{A}^T\right)^* = \boldsymbol{A}$. Then, it holds that

1. All eigenvalues λ_i, $i = 1, \ldots, M$ of $\boldsymbol{A}$ are real.
2. $\boldsymbol{A}$ has a complete set of orthonormal eigenvectors $\boldsymbol{u}_i$, $i = 1, \ldots, M$. Since the eigenvectors are orthonormal, $\boldsymbol{u}_i^H\boldsymbol{u}_j = \delta_{ij}$, where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.
3. Arranging the eigenvectors in a matrix $\boldsymbol{U} = \left[\boldsymbol{u}_1 \ \ldots \ \boldsymbol{u}_M\right]$, we find that $\boldsymbol{U}$ is *unitary*, that is, $\boldsymbol{U}^H\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^H = \boldsymbol{I}$. In addition,
$$\boldsymbol{U}^H\boldsymbol{A}\boldsymbol{U} = \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_M \end{bmatrix}.$$
4. If $\boldsymbol{A} = \boldsymbol{A}^T \in \mathcal{R}^{M\times M}$, it is called simply *symmetric*. In this case, not only the eigenvalues, but also the eigenvectors are real.
5. If $\boldsymbol{A}$ is symmetric and invertible, then $\boldsymbol{A}^{-1}$ is also symmetric.

Fact 6 (Positive-definite matrices). If $\boldsymbol{A} \in \mathcal{C}^{M\times M}$ is Hermitian symmetric and moreover for all $\boldsymbol{u} \in \mathcal{C}^M$ it holds that

$$\boldsymbol{u}^H\boldsymbol{A}\boldsymbol{u} \geq 0, \qquad (12.34)$$

then $\boldsymbol{A}$ is *positive semi-definite*. Positive semi-definite matrices have non-negative eigenvalues: $\lambda_i \geq 0$, $i = 1, \ldots, M$. They may be singular, if one or more eigenvalue is zero.

If the inequality (12.34) is strict, that is, if

$$\boldsymbol{u}^H\boldsymbol{A}\boldsymbol{u} > 0 \qquad (12.35)$$

for all $\boldsymbol{u} \neq \boldsymbol{0}$, then $\boldsymbol{A}$ is *positive-definite*. All eigenvalues λ_i of positive-definite matrices satisfy $\lambda_i > 0$. Positive-definite matrices are always nonsingular.

Finally, all positive-definite matrices admit a Cholesky factorization, i.e., if $\boldsymbol{A}$ is positive-definite, there is a lower-triangular matrix $\boldsymbol{L}$ such that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^H$ [14].

Fact 7 (Norms and spectral radius). The *spectral radius* $\rho(\boldsymbol{A})$ of a matrix $\boldsymbol{A} \in \mathcal{C}^{M \times M}$ is

$$\rho(\boldsymbol{A}) = \max_{1 \le i \le M} |\lambda_i|, \tag{12.36}$$

in which λ_i are the eigenvalues of $\boldsymbol{A}$. The spectral radius is important, because a linear system

$$\boldsymbol{x}(n+1) = \boldsymbol{A}\boldsymbol{x}(n)$$

is stable if and only if $\rho(\boldsymbol{A}) \le 1$, as is well known. A useful inequality is that, for any matrix norm $\|\cdot\|$ such that for any $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{C}^{M \times M}$

$$\|\boldsymbol{A}\boldsymbol{B}\| \le \|\boldsymbol{A}\|\|\boldsymbol{B}\|,$$

it holds that

$$\rho(\boldsymbol{A}) \le \|\boldsymbol{A}\|. \tag{12.37}$$

This property is most useful when used with norms that are easy to evaluate, such as the 1-norm,

$$\|\boldsymbol{A}\|_1 = \max_{1 \le i \le M} \sum_{j=1}^{M} |a_{ij}|,$$

where a_{ij} are the entries of $\boldsymbol{A}$. We could also use the infinity norm,

$$\|\boldsymbol{A}\|_\infty = \max_{1 \le j \le M} \sum_{i=1}^{M} |a_{ij}|.$$

1.12.1.3 Applications

Due to the ability to adjust themselves to different environments, adaptive filters can be used in different signal processing and control applications. Thus, they have been used as a powerful device in several fields, such as communications, radar, sonar, biomedical engineering, active noise control, modeling, etc. It is common to divide these applications into four groups:

1. interference cancellation;
2. system identification;
3. prediction; and
4. inverse system identification.

In the first three cases, the goal of the adaptive filter is to find an approximation $\hat{y}(n)$ for the signal $y(n)$, which is contained in the signal $d(n) = y(n) + v(n)$. Thus, as $\hat{y}(n)$ approaches $y(n)$, the signal $e(n) = d(n) - \hat{y}(n)$ approaches $v(n)$. The difference between these applications is in what we are interested. In interference cancellation, we are interested in the signal $v(n)$, as is the case of acoustic echo cancellation, where $v(n)$ is the speech of the person using the hands-free telephone (go back to Figures 12.4 and 12.7). In system identification, we are interested in the filter parameters, and in

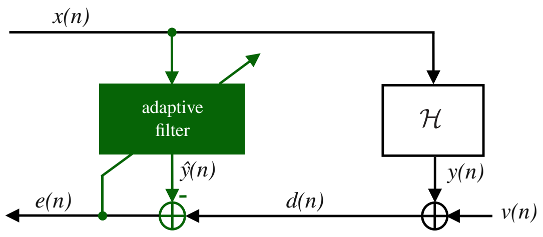


FIGURE 12.13

Scheme for interference cancellation and system identification. $x(n)$ and $y(n)$ are correlated to each other.

prediction, we may be interested in the signal $v(n)$ and/or in the filter parameters. Note that in all these applications (interference cancellation, system identification and prediction), the signal $e(n)$ is an approximation for $v(n)$ and should not converge to zero, except when $v(n) \equiv 0$. In inverse system identification, differently from the other applications, the signal at the output of the adaptive filter must be as close as possible to the signal $d(n)$ and thus, ideally, the signal $e(n)$ should be zeroed. In the sequel, we give an overview of these four groups in order to arrive at a common formulation that will simplify our analysis of adaptive filtering algorithms in further sections.

1.12.1.3.1 *Interference cancellation*

In a general interference cancellation problem, we have access to a signal $d(n)$, which is a mixture of two other signals, $v(n)$ and $y(n)$. We are interested in one of these signals, and want to separate it from the other (the interference) (see Figure 12.13). Even though we do not know $v(n)$ or $y(n)$, we have some information about $y(n)$, usually in the form of a reference signal $x(n)$ that is related to $y(n)$ through a filtering operation $\mathcal{H}$. We may know the general form of this operation, for example, we may know that the relation is linear and well approximated by an FIR filter with 200 coefficients. However, we do not know the parameters (the filter coefficients) necessary to reproduce it. The goal of the adaptive filter is to find an approximation $\hat{y}(n)$ to $y(n)$, given only $x(n)$ and $d(n)$. In the process of finding this $\hat{y}$, the adaptive filter will construct an approximation for the relation $\mathcal{H}$. A typical example of an interference cancellation problem is the echo cancellation example we gave in Section 1.12.1.1. In that example, $x(n)$ is the far-end voice signal, $v(n)$ is the near-end voice signal, and $y(n)$ is the echo. The relation $\mathcal{H}$ is usually well approximated by a linear FIR filter with a few hundred taps.

The approximation for $\mathcal{H}$ is a by-product, that does not need to be very accurate as long as it leads to a $\hat{y}(n)$ close to $y(n)$. This configuration is shown in Figure 12.13 and is called *interference cancellation*, since the interference $y(n)$ should be canceled. Note that we do not want to make $e(n) \equiv 0$, otherwise we would not only be killing the interference $y(n)$, but also the signal of interest $v(n)$.

There are many applications of interference cancellation. In addition to acoustic and line echo cancellation, we can mention, for example, adaptive notch filters for cancellation of sinusoidal interference (a common application is removal of 50 or 60 Hz interference from the mains line), cancellation of the maternal electrocardiography in fetal electrocardiography, cancellation of echoes in long distance telephone circuits, active noise control (as in noise-canceling headphones), and active vibration control.

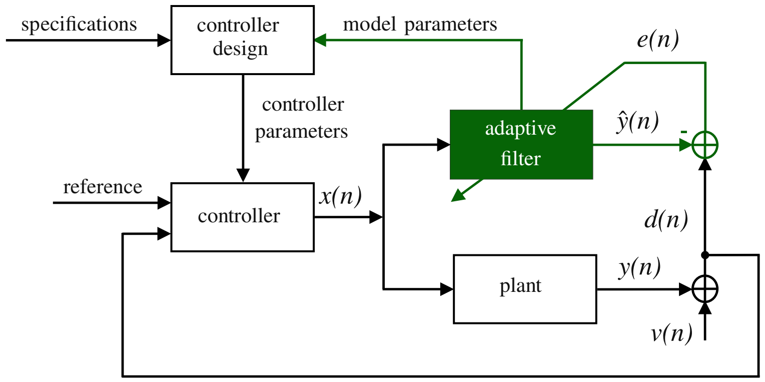


FIGURE 12.14

Scheme for plant identification in a control system.

1.12.1.3.2 ***System identification***

In interference cancellation, the coefficients of the adaptive filter converge to an approximation for the true relation $\mathcal{H}$, but as mentioned before, this approximation is a by-product that does not need to be very accurate. However, there are some applications in which the goal is to construct an approximation as accurate as possible for the unknown relation $\mathcal{H}$ between $x(n)$ and $y(n)$, thus obtaining a model for the unknown system. This is called *system identification* or *modeling* (the diagram in Figure 12.13 applies also for this case). The signal $d(n)$ is now composed of the output $y(n)$ of an unknown system $\mathcal{H}$, plus noise $v(n)$. The reference signal $x(n)$ is the input to the system which, when possible, is chosen to be white noise. In general, this problem is harder than interference cancellation, as we shall see in Section 1.12.2.1.5, due to some conditions that the reference signal $x(n)$ must satisfy. However, one point does not change: in the ideal case, the signal $e(n)$ will be equal to the noise $v(n)$. Again, we do not want to make $e(n) \equiv 0$, otherwise we would be trying to model the noise (however, the smaller the noise the easier the task, as one would expect).

In many control systems, an unknown dynamic system (also called as *plant* in control system terminology) is identified online and the result is used in a self-tuning controller, as depicted in Figure 12.14. Both the plant and the adaptive filter have the same input $x(n)$. In practical situations, the plant to be modeled is noisy, which is represented by the signal $v(n)$ added to the plant output. The noise $v(n)$ is generally uncorrelated with the plant input. The task of the adaptive filter is to minimize the error model $e(n)$ and track the time variations in the dynamics of the plant. The model parameters are continually fed back to the controller to obtain the controller parameters used in the self-tuning regulator loop [15].

There are many applications that require the identification of an unknown system. In addition to control systems, we can mention, for example, the identification of the room impulse response, used to study the sound quality in concert rooms, and the estimation of the communication channel impulse response, required by maximum likelihood detectors and blind equalization techniques based on second-order statistics.

1.12.1.3.3 Prediction

In *prediction*, the goal is to find a relation between the current sample $d(n)$ and previous samples $d(n-L),\ldots,d(n-L-M+1)$, as shown in Figure 12.15. Therefore, we want to model $d(n)$ as a part $y(n)$ that depends only on $d(n-L),\ldots,d(n-L-M+1)$, and a part $v(n)$ that represents new information. For example, for a linear model, we would have

$$d(n)=\underbrace{\sum_{k=0}^{M-1} h_k d(n-L-k)}_{y(n)}+v(n),$$

Thus, we in fact have again the same problem as in Figure 12.13. The difference is that now the reference signal is a delayed version of $d(n)$: $x(n)=d(n-L)$, and the adaptive filter will try to find an approximation $\hat{y}(n)$ for $y(n)$, thereby separating the "predictable" part $y(n)$ of $d(n)$ from the "new information" $v(n)$, to which the signal $e(n)=d(n)-\hat{y}(n)$ should converge. The system $\mathcal{H}$ in Figure 12.13 now represents the relation between $d(n-L)$ and the predictable part of $d(n)$.

Prediction finds application in many fields. We can mention, for example, linear predictive coding (LPC) and adaptive differential pulse code modulation (ADPCM) used in speech coding, adaptive line enhancement, autoregressive spectral analysis, etc.

For example, adaptive line enhancement (ALE) seeks the solution for a classical detection problem, whose objective is to separate a narrowband signal $y(n)$ from a wideband signal $v(n)$. This is the case, for example, of finding a low-level sine wave (predictable narrowband signal) in noise (non-predictable wideband signal). The signal $d(n)$ is constituted by the sum of these two signals. Using $d(n)$ as the reference signal and a delayed replica of it, i.e., $d(n-L)$, as input of the adaptive filter as in Figure 12.15, the output $\hat{y}(n)$ provides an estimate of the (predictable) narrowband signal $y(n)$, while the error signal $e(n)$ provides an estimate of the wideband signal $v(n)$. The delay L is also known as *decorrelation parameter* of the ALE, since its main function is to remove the correlation between the wideband signal $v(n)$ present in the reference $d(n)$ and in the delayed predictor input $d(n-L)$.

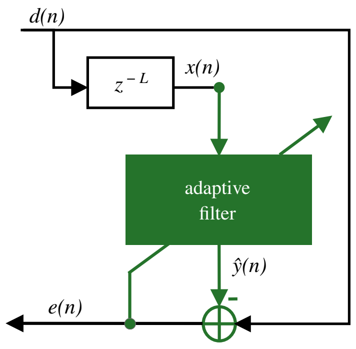


FIGURE 12.15

Scheme for prediction. $x(n)=d(n-L)$ is a delayed version of $d(n)$.

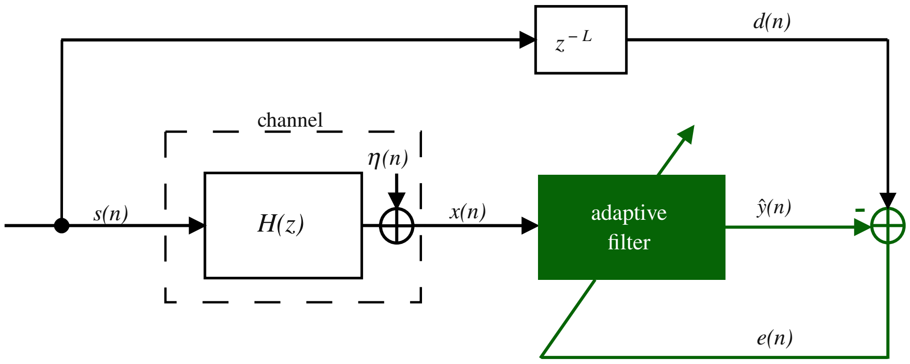


FIGURE 12.16

Simplified communications system with an adaptive equalizer in the training mode.

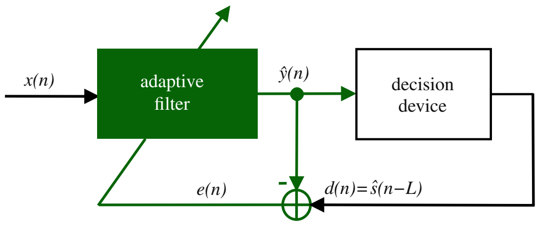


FIGURE 12.17

Adaptive equalizer in the decision-directed mode.

1.12.1.3.4 *Inverse system identification*

Inverse system identification, also known as *deconvolution*, has been widely used in different fields as communications, acoustics, optics, image processing, control, among others. In communications, it is also known as *channel equalization* and the adaptive filter is commonly called as *equalizer*. Adaptive equalizers play an important role in digital communications systems, since they are used to mitigate the inter-symbol interference (ISI) introduced by dispersive channels. Due to its importance, we focus on the equalization application, which is explained in the sequel.

A simplified baseband communications system is depicted in Figure 12.16. The signal $s(n)$ is transmitted through an unknown channel, whose model is constituted by an FIR filter with transfer function $H(z) = h_0 + h_1 z^{-1} + \cdots + h_{K-1} z^{K-1}$ and additive noise $\eta(n)$. Due to the channel memory, the signal at the receiver contains contributions not only from $s(n)$, but also from the previous symbols $s(n-1), s(n-2), \ldots, s(n-K+1)$, i.e.,

$$x(n) = \underbrace{\sum_{k=0}^{L-1} h_k s(n-k)}_{\text{pre-ISI}} + h_L s(n-L) + \underbrace{\sum_{k=L+1}^{K-1} h_k s(n-k)}_{\text{post-ISI}} + \eta(n). \tag{12.38}$$

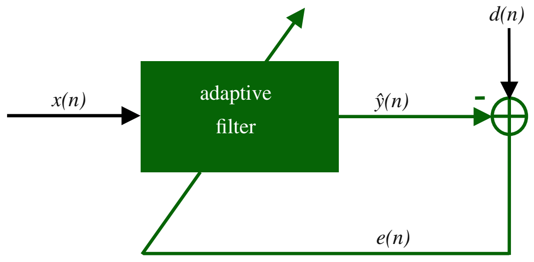


FIGURE 12.18

Inputs and output of an adaptive filter.

Assuming that the overall channel-equalizer system imposes a delay of L samples, the adaptive filter will try to find an approximation $\hat{y}(n)$ for $d(n) = s(n - L)$ and for this purpose, the two summations in (12.38), which constitute the inter-symbol interference, must be mitigated. Of course, when you are transmitting information the receiver will not have access to $d(n - L)$. The filter will adapt during a *training* phase, in which the transmitter sends a pre-agreed signal. You can see that in this case, the role of $x(n)$ and $d(n)$ is the reverse of that in system identification. In this case, you would indeed like to have $e(n) \equiv 0$. The problem is that this is not possible, given the presence of noise in $x(n)$. The role of the adaptive filter is to approximately invert the effect of the channel, at the same time trying to suppress the noise (or at least, not to amplify it too much). In one sense, we are back to the problem of separating two signals, but now the mixture is in the reference signal $x(n)$, so what can and what cannot be done is considerably different from the other cases.

The scheme of Figure 12.16 is also called training mode, since the delayed version of the transmitted sequence $d(n) = s(n - L)$ (training sequence) is known at the receiver. After the convergence of the filter, the signal $d(n)$ is changed to the estimate $\hat{s}(n - L)$ obtained at the output of a decision device, as shown in Figure 12.17. In this case, the equalizer works in the so-called decision-directed mode. The decision device depends on the signal constellation—for example, if $s(n) = \pm 1$, the decision device returns $+1$ for $\hat{y}(n) \geq 0$, and -1 for $\hat{y}(n) < 0$.

1.12.1.3.5 A common formulation

In all the four groups of applications, the inputs of the adaptive filter are given by the signals $x(n)$ and $d(n)$ and the output by $\hat{y}(n)$. Note that the input $d(n)$ appears effectively in the signal $e(n) = d(n) - \hat{y}(n)$, which is computed and fed back at each time instant n, as shown in Figure 12.18. In the literature, the signal $d(n)$ is referred to as desired signal, $x(n)$ as reference signal and $e(n)$ as error signal. These names unfortunately are somewhat misleading, since they give the impression that our goal is to recover exactly $d(n)$ by filtering (possibly in a nonlinear way) the reference $x(n)$. In almost all applications, this is far from the truth, as previously discussed. In fact, except in the case of channel equalization, exactly zeroing the error would result in very poor performance.

The only application in which $d(n)$ is indeed a “desired signal” is channel equalization, in which $d(n) = s(n - L)$. Despite this particularity in channel equalization, the common feature in all adaptive filtering applications is that the filter must learn a relation between the reference $x(n)$ and the desired signal $d(n)$ (we will use this name to be consistent with the literature.) In the process of building this relation, the adaptive filter is able to perform useful tasks, such as separating two mixed signals; or recovering a distorted signal. Section 1.12.2 explains this idea in more detail.

1.12.2 Optimum filtering

We now need to extend the ideas of Section 1.12.1.2, that applied only to periodic (deterministic) signals, to more general classes of signals. For this, we will use tools from the theory of stochastic processes. The main ideas are very similar: as before, we need to choose a structure for our filter and a means of measuring how far or how near we are to the solution. This is done by choosing a convenient cost function, whose minimum we must search iteratively, based only on measurable signals.

In the case of periodic signals, we saw that minimizing the error power (12.3), repeated below,

$$P = \sum_{k=1}^{K_0} \frac{\widetilde{C}_k^2}{2} + \sum_{\ell=1}^{K_1} \frac{B_\ell^2}{2},$$

would be equivalent to zeroing the echo. However, we were not able to compute the error power exactly—we used an approximation through a time-average, as in (12.5), repeated here:

$$\widehat{P}(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^2(n-k),$$

where N is a convenient window length (in Section 1.12.1.2, we saw that choosing a large window length is approximately equivalent to choosing $N = 1$ and a small step-size). Since we used an approximated estimate of the cost, our solution was also an approximation: the estimated coefficients did not converge exactly to their optimum values, but instead hovered around the optimum (Figures 12.9 and 12.10).

The minimization of the time-average $\widehat{P}(n)$ also works in the more general case in which the echo is modeled as a random signal—but what is the corresponding exact cost function? The answer is given by the property of *ergodicity* that some random signals possess. If a random signal $s(n)$ is such that its mean $\mathrm{E}\{s(n)\}$ does not depend on n and its autocorrelation $\mathrm{E}\{s(n)s(k)\}$ depends only on $n-k$, it is called wide-sense stationary (WSS) [16]. If $s(n)$ is also (mean-square) ergodic, then

$$\text{Average power of } s(n) = \lim_{N\to\infty} \frac{1}{N+1} \sum_{n=0}^{N} s^2(n) = \mathrm{E}\{s^2(n)\}. \tag{12.39}$$

Note that $\mathrm{E}\{s^2(n)\}$ is an ensemble average, that is, the expected value is computed over all possible realizations of the random signal. Relation (12.39) means that for an ergodic signal, the time average of a single realization of the process is equal to the ensemble-average of all possible realizations of the process. This ensemble average is the exact cost function that we would like to minimize.

In the case of adaptive filters, (12.39) holds approximately for $e(n)$ for a finite value of N if the environment does not change too fast, and if the filter adapts slowly. Therefore, for random variables we will still use the average error power as a measure of how well the adaptive filter is doing. The difference is that in the case of periodic signals, we could understand the effect of minimizing the average error power in terms of the amplitudes of each harmonic in the signal, but now the interpretation will be in terms of ensemble averages (variances).

Although the error power is not the only possible choice for cost function, it is useful to study this choice in detail. Quadratic cost functions such as the error power have a number of properties that make them popular. For example, they are differentiable, and so it is relatively easy to find a closed-form solution for the optimum, and in the important case where all signals are Gaussian, the optimum filters are linear. Of course, quadratic cost functions are not the best option in all situations: the use of other cost functions, or even of adaptive filters not based on the minimization of a cost function, is becoming more common. We will talk about some of the most important alternatives in Sections 1.12.5.2 and 1.12.5.4.

Given the importance of quadratic cost functions, in this section we study them in detail. Our focus is on what could and what could not be done if the filter were able to measure perfectly $\mathrm{E}\{e^2(n)\}$ at each instant. This discussion has two main goals: the first is to learn what is feasible, so we do not expect more from a filter than it can deliver. The second is to enable us to make more knowledgeable choices when designing a filter—for example, which filter order should be used, and for system identification, which input signal would result in better performance. In Section 1.12.4.4.3 we will study the performance of adaptive filters taking into consideration their imperfect knowledge of the environment.

1.12.2.1 Linear least-mean squares estimation

In the examples we gave in previous sections, a generic adaptive filtering problem resumed to this: we are given two sequences, $\{x(n)\}$ and $\{d(n)\}$, as depicted in Figure 12.18. It is known, from physical arguments, that $d(n)$ has two parts, one that is in some sense related to $x(n)$, $x(n-1)$, ..., and another that is not, and that both parts are combined additively, that is,

$$d(n) = \underbrace{\mathcal{H}(x(n), x(n-1), \dots)}_{y(n)} + v(n), \tag{12.40}$$

where $\mathcal{H}(\cdot)$ is an unknown relation, and $v(n)$ is the part "unrelated" to $\{x(n)\}$. In our examples so far, $\mathcal{H}(\cdot)$ was linear, but we do not need to restrict ourselves to this case. Our objective is to extract $y(n)$ and $v(n)$ from $d(n)$, based only on observations of $\{x(n)\}$ and $\{d(n)\}$. We saw in Section 1.12.1.2 that the average power of the difference $e(n)$ between $d(n)$ and our current approximation $\hat{y}(n)$ could be used as a measure of how close we are to the solution. In general, $\mathcal{H}(\cdot)$ is time-variant, i.e., depends directly on n. We will not write this dependence explicitly to simplify the notation.

In this section we ask in some sense the inverse problem, that is: given two sequences $\{x(n)\}$ and $\{d(n)\}$, what sort of relation will be found between them if we use $\mathrm{E}\{e^2(n)\}$ as a standard? The difference is that we now do not assume a model such as (12.40); instead, we want to know what kind of model results from the exact minimization of $\mathrm{E}\{e^2(n)\}$, where $e(n)$ is defined as the difference between $d(n)$ and a function of $x(n)$, $x(n-1)$, ...

We can answer this question in an entirely general way, without specifying beforehand the form of $\mathcal{H}$. This is done in Box 4. However, if we have information about the physics of a problem and know that the relation $\mathcal{H}(x(n), x(n-1), \ldots)$ is of a certain kind, we can restrict the search for the solution $\hat{\mathcal{H}}$ to this class. This usually reduces the complexity of the filter and increases its convergence rate (because less data is necessary to estimate a model with less unknowns, as we will see in Section 1.12.3). In this section we focus on this second option.

The first task is to describe the class $\mathcal{F}$ of allowed functions $\hat{\mathcal{H}}$. This may be done by choosing a relation that depends on a few parameters, such as (recall that the adaptive filter output is $\hat{y}(n) = \hat{\mathcal{H}}(x(n), x(n-1), \ldots)$)

$$\begin{aligned}
\mathcal{F}_{\text{FIR}} \text{ (FIR filter):} \quad & \hat{y}(n) = w_0 x(n) + \cdots + w_{M-1} x(n-M+1), && (12.41)\\
\mathcal{F}_{\text{IIR}} \text{ (IIR filter):} \quad & \hat{y}(n) = -a_1 \hat{y}(n-1) + b_0 x(n), && (12.42)\\
\mathcal{F}_{\text{V}} \text{ (Volterra filter):} \quad & \hat{y}(n) = w_0 x(n) + w_1 x(n-1) + w_{0,0} x^2(n)\\
& \quad + w_{0,1} x(n) x(n-1) + w_{1,1} x(n-1)^2, && (12.43)\\
\mathcal{F}_{\text{S}} \text{ (Saturation):} \quad & \hat{y}(n) = \arctan(a x(n)). && (12.44)
\end{aligned}$$

In each of these cases, the relation between the input sequence $\{x(n)\}$ and $d(n)$ is constrained to a certain class, for example, linear length-M FIR filters in (12.41), first-order IIR filters in (12.42), and second-order Volterra filters in (12.43). Each class is described by a certain number of parameters: the filter coefficients $w_0, \ldots, w_{M-1}$ in the case of FIR filters, a_1 and b_0 in the case of IIR filters, and so on. The task of the adaptive filter will then be to choose the values of the parameters that best fit the data.

For several practical reasons, it is convenient if we make the filter output depend linearly on the parameters, as happens in (12.41) and in (12.43). It is important to distinguish linear in the parameters from input-output linear: (12.43) is linear in the parameters, but the relation between the input sequence $\{x(n)\}$ and the output $\hat{y}(n)$ is nonlinear. What may come as a surprise is that the IIR filter of Eq. (12.42) is *not* linear in the parameters: in fact, $\hat{y}(n) = -a_1\hat{y}(n-1) + b_0 x(n) = a_1^2 \hat{y}(n-2) - a_1 b_0 x(n-1) + b_0 x(n) = \cdots$— you can see that $\hat{y}(n)$ depends nonlinearly on a_1 and b_0.

Linearly parametrized classes $\mathcal{F}$, such as $\mathcal{F}_{\text{FIR}}$ (12.41) and $\mathcal{F}_{\text{V}}$ (12.43) are popular because in general it is easier to find the optimum parameters, both theoretically and in real time. In fact, when the filter output depends linearly on the parameters and the cost function is a *convex* function of the error, it can be shown that the optimal solution is unique (see Box 5). This is a very desirable property, since it simplifies the search for the optimal solution.

In the remainder of this section we will concentrate on classes of relations that are linear in the parameters. As $\mathcal{F}_{\text{V}}$ shows, this does not imply that we are restricting ourselves to linear models. There are, however, adaptive filtering algorithms that use classes of relations that are not linear in the parameters, such as IIR adaptive filters. On the other hand, blind equalization algorithms are based on non-convex cost functions.

Assume then that we have chosen a convenient class of relations $\mathcal{F}$ that depends linearly on its parameters. That is, we want to solve the problem

$$\min_{\hat{\mathcal{H}} \in \mathcal{F}} \mathrm{E}\left\{\left[d(n) - \hat{\mathcal{H}}(x(n), x(n-1), \ldots)\right]^2\right\}. \tag{12.45}$$

We want to know which properties the solution to this problem will have when $\mathcal{F}$ depends linearly on a finite number of parameters. In this case, $\mathcal{F}$ is a linear combinations of certain functions ϕ_i of $x(n), x(n-1), \dots$

$$\hat{y}(n) = w_0\phi_0 + w_1\phi_1 + \cdots + w_{M-1}\phi_{M-1} \overset{\Delta}{=} \boldsymbol{w}^T\boldsymbol{\phi}, \tag{12.46}$$

where in general $\phi_i = \phi_i(x(n), x(n-1), \dots)$, $0 \le i \le M-1$. The vector $\boldsymbol{\phi}$ is known as *regressor*. In the case of length-M FIR filters, we would have

$$\phi_0 = x(n),\ \phi_1 = x(n-1),\ \dots,\ \phi_{M-1} = x(n-M+1),$$

whereas in the case of second-order Volterra filters with memory 1, we would have

$$\phi_0 = x(n),\ \phi_1 = x(n-1),\ \phi_2 = x^2(n),\ \phi_3 = x(n)x(n-1),\ \phi_4 = x^2(n-1).$$

Our problem can then be written in general terms as: find $\boldsymbol{w}_o$ such that

$$\boldsymbol{w}_o = \arg\min_{\boldsymbol{w}} \mathrm{E}\{(d - \boldsymbol{w}^T\boldsymbol{\phi})^2\}. \tag{12.47}$$

We omitted the dependence of the variables on n to lighten the notation. Note that, in general, $\boldsymbol{w}_o$ will also depend on n.

To solve this problem, we use the facts that $\boldsymbol{w}^T\boldsymbol{\phi} = \boldsymbol{\phi}^T\boldsymbol{w}$ to expand the expected value

$$J(\boldsymbol{w}) \overset{\Delta}{=} \mathrm{E}\left\{\left(d - \boldsymbol{w}^T\boldsymbol{\phi}\right)^2\right\} = \mathrm{E}\{d^2\} - 2\boldsymbol{w}^T\mathrm{E}\{d\boldsymbol{\phi}\} + \boldsymbol{w}^T\mathrm{E}\{\boldsymbol{\phi}\boldsymbol{\phi}^T\}\boldsymbol{w}.$$

Recall that the weight vector $\boldsymbol{w}$ is *not* random, so we can take it out of the expectations.

Define the autocorrelation of d, and also the cross-correlation vector and autocorrelation matrix

$$r_d = \mathrm{E}\{d^2\}, \quad \boldsymbol{r}_{d\boldsymbol{\phi}} = \mathrm{E}\{d\boldsymbol{\phi}\}, \quad \boldsymbol{R}_{\boldsymbol{\phi}} = \mathrm{E}\{\boldsymbol{\phi}\boldsymbol{\phi}^T\}. \tag{12.48}$$

The cost function then becomes

$$J(\boldsymbol{w}) = r_d - 2\boldsymbol{w}^T\boldsymbol{r}_{d\boldsymbol{\phi}} + \boldsymbol{w}^T\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{w}. \tag{12.49}$$

Differentiating $J(\boldsymbol{w})$ with respect to $\boldsymbol{w}$, we obtain

$$\frac{\partial J}{\partial \boldsymbol{w}^T} = -2\boldsymbol{r}_{d\boldsymbol{\phi}} + 2\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{w},$$

and equating the result to zero, we see that the optimal solution must satisfy

$$\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{w}_o = \boldsymbol{r}_{d\boldsymbol{\phi}}. \tag{12.50}$$

These are known as the *normal*, or *Wiener-Hopf*, equations. The solution, $\boldsymbol{w}_o$, is known as the *Wiener solution*. A note of caution: the Wiener solution is not the same thing as the Wiener filter. The Wiener filter is the linear filter that minimizes the mean-square error, without restriction of filter order [17]. The difference is that the Wiener solution has the filter order pre-specified (and is not restricted to linear filters, as we saw).

When the autocorrelation matrix is non-singular (which is usually the case), the Wiener solution is

$$\boldsymbol{w}_{\mathrm{o}} = \boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}. \tag{12.51}$$

Given $\boldsymbol{w}_{\mathrm{o}}$, the optimum error will be

$$v_{\mathrm{o}} = d - \boldsymbol{w}_{\mathrm{o}}^T\boldsymbol{\phi}. \tag{12.52}$$

Note that the expected value of v_{o} is not necessarily zero:

$$\mathrm{E}\{v_{\mathrm{o}}\} = \mathrm{E}\{d\} - \boldsymbol{w}_{\mathrm{o}}^T\mathrm{E}\{\boldsymbol{\phi}\}. \tag{12.53}$$

If, for example, $\mathrm{E}\{\boldsymbol{\phi}\} = \boldsymbol{0}$ and $\mathrm{E}\{d\} \neq 0$, then $\mathrm{E}\{v_{\mathrm{o}}\} = \mathrm{E}\{d\} \neq 0$. In practice, it is good to keep $\mathrm{E}\{v_{\mathrm{o}}\} = 0$, because we usually know that v_{o} should approximate a zero-mean signal, such as speech or noise. We will show shortly, in Section 1.12.2.1.4, how to guarantee that v_{o} has zero mean.

1.12.2.1.1 *Orthogonality condition*

A key property of the Wiener solution is that the optimum error is orthogonal to the regressor $\boldsymbol{\phi}$, that is,

$$\mathrm{E}\{v_{\mathrm{o}}\boldsymbol{\phi}\} = \mathrm{E}\{\boldsymbol{\phi}(d - \underbrace{\boldsymbol{w}_{\mathrm{o}}^T\boldsymbol{\phi}}_{=\boldsymbol{\phi}^T\boldsymbol{w}_{\mathrm{o}}})\} = \mathrm{E}\{d\boldsymbol{\phi}\} - \mathrm{E}\{\boldsymbol{\phi}\boldsymbol{\phi}^T\}\boldsymbol{w}_{\mathrm{o}} = \boldsymbol{r}_{d\boldsymbol{\phi}} - \boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{w}_{\mathrm{o}} = \boldsymbol{0}. \tag{12.54}$$

We saw a similar condition in Section 1.12.1.2. It is very useful to remember this result, known as the *orthogonality condition*: from it, we will find when to apply the cost function (12.47) to design an adaptive filter. Note that when v_{o} has zero mean, (12.54) also implies that v_{o} and $\boldsymbol{\phi}$ are uncorrelated.

Remark 1. You should not confuse orthogonal with uncorrelated. Two random variables x and y are orthogonal if $\mathrm{E}\{xy\} = 0$, whereas they are uncorrelated if $\mathrm{E}\{xy\} = \mathrm{E}\{x\}\,\mathrm{E}\{y\}$. The two concepts coincide only if either x or y have zero mean.

Remark 2. The orthogonality condition is an intuitive result, if we remember that we can think of the space of random variables with finite variance as a vector space. In fact, define the cross-correlation $r_{xy} = \mathrm{E}\{xy\}$ between two random variables x and y as the inner product. Then the autocorrelation of a random variable x, $\mathrm{E}\{x^2\}$, would be interpreted as the square of the "length" of x. In this case, our problem (12.47) is equivalent to finding the vector in the subspace spanned by $\phi_0, \dots, \phi_{M-1}$ that is closest to d. As we know, the solution is such that the error is orthogonal to the subspace. So, the orthogonality condition results from the vector space structure of our problem and the quadratic nature of our cost function. See Figure 12.19.

Remark 3. The difference between (12.54) and the corresponding result obtained in Box 4, Eq. (12.105), is that here the optimum error is orthogonal only to the functions ϕ_i of $x(n), x(n-1), \dots$ that were included in the regressor (and their linear combinations), not to any function, as in (12.105). This is not surprising: in this section the optimization was made only over the functions in class $\mathcal{F}$, and in Box 4 we allow any function. Both solutions, (12.106) and (12.51), will be equal if the general solution according to Box 4 is indeed in $\mathcal{F}$.

Remark 4. The optimal mean-square error is (recall that $\mathrm{E}\{v_o\boldsymbol{\phi}\} = \mathbf{0}$)

$$\begin{aligned} J_{\min} = r_{v,o} &= \mathrm{E}\{v_o^2\} = \mathrm{E}\{v_o(d - \boldsymbol{w}_o^T\boldsymbol{\phi})\} = \mathrm{E}\{v_o d\} \\ &= \mathrm{E}\{(d - \boldsymbol{w}_o^T\boldsymbol{\phi})d\} = \mathrm{E}\{d^2\} - \boldsymbol{w}_o^T\boldsymbol{r}_{d\boldsymbol{\phi}} = r_d - \boldsymbol{r}_{d\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}. \end{aligned} \tag{12.55}$$

Now that we have the general solution to (12.47), we turn to the question: for which class of problems is the quadratic cost function adequate?

1.12.2.1.2 *Implicit vs. physical models*

From (12.54), we see that the fact that we are minimizing the mean-square error between d and a linear combination of the regressor $\boldsymbol{\phi}$ induces a model

$$d = \boldsymbol{w}_o^T\boldsymbol{\phi} + v_o, \tag{12.56}$$

in which v_o is orthogonal to $\boldsymbol{\phi}$. This is not an assumption, as we sometimes see in the literature, but a consequence of the quadratic cost function we chose. In other words, there is always a relation such as (12.56) between any pair $(d, \boldsymbol{\phi})$, as long as both have finite second-order moments. This relation may make sense from a physical analysis of the problem (as we saw in the echo cancellation example), or be simply a consequence of solving (12.47) (see Section 1.12.2.1.3 for examples).

On the other hand, the orthogonality condition allows us to find out when the solution of (12.47) will be able to successfully solve a problem: assume now that we know beforehand, by physical arguments

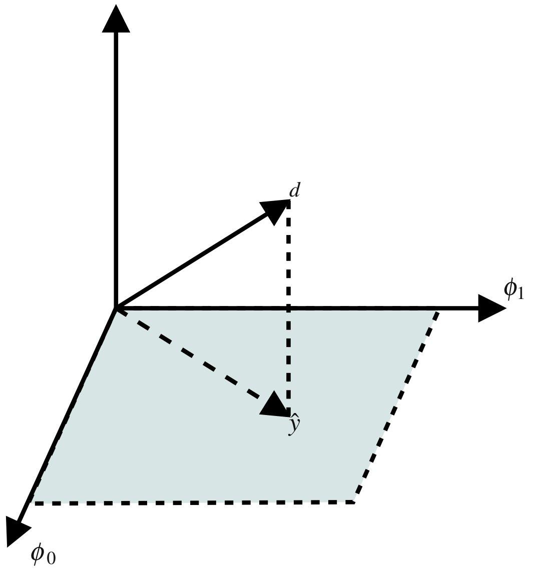


FIGURE 12.19

Orthogonality in vector spaces with inner products.

about the problem, that d and $\boldsymbol{\phi}$ must be related through

$$d = \boldsymbol{w}_*^T \boldsymbol{\phi} + v, \tag{12.57}$$

where $\boldsymbol{w}_*$ is a certain coefficient vector and v is orthogonal to $\boldsymbol{\phi}$. Will the solution to (12.47) be such that $\boldsymbol{w}_\mathrm{o} = \boldsymbol{w}_*$ and $v_\mathrm{o} = v$?

To check if this is the case, we only need to evaluate the cross-correlation vector $\boldsymbol{r}_{d\boldsymbol{\phi}}$ assuming the model (12.57):

$$\boldsymbol{r}_{d\boldsymbol{\phi}} = \mathrm{E}\{d\boldsymbol{\phi}\} = \mathrm{E}\{\boldsymbol{\phi}(\underbrace{\boldsymbol{w}_*^T\boldsymbol{\phi}}_{=\boldsymbol{\phi}^T\boldsymbol{w}_*} + v)\} = \mathrm{E}\{\boldsymbol{\phi}\boldsymbol{\phi}^T\}\boldsymbol{w}_* + \underbrace{\mathrm{E}\{\boldsymbol{\phi}v\}}_{=\boldsymbol{0}} = \boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{w}_*. \tag{12.58}$$

Therefore, $\boldsymbol{w}_*$ also obeys the normal equations, and we can conclude that $\boldsymbol{w}_\mathrm{o} = \boldsymbol{w}_*$, $v_\mathrm{o} = v$, as long as $\boldsymbol{R}_{\boldsymbol{\phi}}$ is nonsingular.

Even if $\boldsymbol{R}_{\boldsymbol{\phi}}$ is singular, the optimal error v_o will always be equal to v (in a certain sense). In fact, if $\boldsymbol{R}_{\boldsymbol{\phi}}$ is singular, then there exists a vector $\boldsymbol{a}$ such that $\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{a} = \boldsymbol{0}$ (see Fact 2 in Box 3), and thus any solution $\boldsymbol{w}_\mathrm{o}$ of the normal equations (we know from (12.58) that there is at least one, $\boldsymbol{w}_*$) will have the form

$$\boldsymbol{w}_\mathrm{o} = \boldsymbol{w}_* + \boldsymbol{a}, \ \ \boldsymbol{a} \in N(\boldsymbol{R}_{\boldsymbol{\phi}}), \tag{12.59}$$

where $N(\boldsymbol{R}_{\boldsymbol{\phi}})$ is the null-space of $\boldsymbol{R}_{\boldsymbol{\phi}}$. In addition,

$$0 = \boldsymbol{a}^T\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{a} = \boldsymbol{a}^T\mathrm{E}\{\boldsymbol{\phi}\boldsymbol{\phi}^T\}\boldsymbol{a} = \mathrm{E}\{\boldsymbol{a}^T\boldsymbol{\phi}\boldsymbol{\phi}^T\boldsymbol{a}\} = \mathrm{E}\left\{(\boldsymbol{a}^T\boldsymbol{\phi})^2\right\}.$$

Therefore, $\boldsymbol{a}^T\boldsymbol{\phi} = 0$ with probability one. Then,

$$v_\mathrm{o} = d - \boldsymbol{w}_\mathrm{o}^T\boldsymbol{\phi} = d - (\boldsymbol{w}_* + \boldsymbol{a})^T\boldsymbol{\phi} = d - \boldsymbol{w}_*^T\boldsymbol{\phi},$$

with probability one. Therefore, although we cannot say that $v = v_\mathrm{o}$ always, we can say that they are equal with probability one. In other words, when $\boldsymbol{R}_{\boldsymbol{\phi}}$ is singular we may not be able to identify $\boldsymbol{w}_*$, but (with probability one) we are still able to separate d into its two components. More about this in Section 1.12.2.1.5.

1.12.2.1.3 Undermodeling

We just saw that the solution to the quadratic cost function (12.47) is indeed able to separate the two components of d when the regressor that appears in physical model (12.57) is the same $\boldsymbol{\phi}$ that we used to describe our class $\mathcal{F}$, that is, when our choice of $\mathcal{F}$ includes the general solution from Box 4. Let us check now what happens when this is not the case.

Assume there is a relation

$$d = \boldsymbol{w}_*^T\boldsymbol{\phi}_\mathrm{e} + v, \tag{12.60}$$

in which v is orthogonal to $\boldsymbol{\phi}_\mathrm{e}$, but our class $\mathcal{F}$ is defined through a regressor $\boldsymbol{\phi}$ that is a subset of the "correct" one, $\boldsymbol{\phi}_\mathrm{e}$. This situation is called *undermodeling*: our class is not rich enough to describe the true relation between $\{x(n)\}$ and $\{d(n)\}$.

Assume then that

$$\boldsymbol{\phi}_{\mathrm{e}} = \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{\theta} \end{bmatrix}, \tag{12.61}$$

where $\boldsymbol{\phi} \in \mathcal{R}^M$, $\boldsymbol{\theta} \in \mathcal{R}^{K-M}$, with $K > M$. The autocorrelation of $\boldsymbol{\phi}_{\mathrm{e}}$ is

$$\boldsymbol{R}_{\boldsymbol{\phi}_{\mathrm{e}}} = \mathrm{E}\{\boldsymbol{\phi}_{\mathrm{e}}\boldsymbol{\phi}_{\mathrm{e}}^T\} = \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{\phi}} & \boldsymbol{R}_{\boldsymbol{\phi\theta}} \\ \boldsymbol{R}_{\boldsymbol{\phi\theta}}^T & \boldsymbol{R}_{\boldsymbol{\theta}} \end{bmatrix},$$

where $\boldsymbol{R}_{\boldsymbol{\phi\theta}} = \mathrm{E}\{\boldsymbol{\phi}\boldsymbol{\theta}^T\}$. From our hypothesis that $\boldsymbol{\phi}_{\mathrm{e}}$ is orthogonal to v, that is,

$$\mathbf{0} = \mathrm{E}\{v\boldsymbol{\phi}_{\mathrm{e}}\} = \begin{bmatrix} \mathrm{E}\{v\boldsymbol{\phi}\} \\ \mathrm{E}\{v\boldsymbol{\theta}\} \end{bmatrix},$$

we conclude that $\boldsymbol{\phi}$ is also orthogonal to v, so

$$\boldsymbol{r}_{d\boldsymbol{\phi}} = \mathrm{E}\left\{\boldsymbol{\phi}(\boldsymbol{\phi}_{\mathrm{e}}^T\boldsymbol{w}_* + v)\right\} = \mathrm{E}\{\boldsymbol{\phi}\boldsymbol{\phi}_{\mathrm{e}}^T\}\boldsymbol{w}_* + \underbrace{\mathrm{E}\{\boldsymbol{\phi}v\}}_{=\mathbf{0}} = \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{\phi}} & \boldsymbol{R}_{\boldsymbol{\phi\theta}} \end{bmatrix}\boldsymbol{w}_*.$$

Assuming that $\boldsymbol{R}_{\boldsymbol{\phi}}$ is nonsingular, solving the normal Eq. (12.50) we obtain

$$\boldsymbol{w}_{\mathrm{o}} = \boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\begin{bmatrix} \boldsymbol{R}_{\boldsymbol{\phi}} & \boldsymbol{R}_{\boldsymbol{\phi\theta}} \end{bmatrix}\boldsymbol{w}_* = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{R}_{\boldsymbol{\phi\theta}} \end{bmatrix}\boldsymbol{w}_*. \tag{12.62}$$

Now we have two interesting special cases: first, if $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are orthogonal, i.e., if $\boldsymbol{R}_{\boldsymbol{\phi\theta}} = \mathbf{0}$, then $\boldsymbol{w}_{\mathrm{o}}$ contains the first M elements of $\boldsymbol{w}_*$. Let us consider a specific example: assume that

$$\boldsymbol{\phi} = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-M+1) \end{bmatrix}, \quad \text{but} \quad \boldsymbol{\phi}_{\mathrm{e}} = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-K+1) \end{bmatrix}, \tag{12.63}$$

with $M < K$. Then $\boldsymbol{\theta} = \begin{bmatrix} x(n-M) & \dots & x(n-K+1) \end{bmatrix}^T$. This situation is very common in practice.

In this case, $\boldsymbol{R}_{\boldsymbol{\phi\theta}} = \mathbf{0}$ when $\{x(n)\}$ is zero-mean white noise. This is one reason why white noise is the preferred input to be used in system identification: even in the (very likely in practice) case in which $M < K$, at least the first elements of $\boldsymbol{w}_*$ are identified without bias.

On the other hand, if $\boldsymbol{R}_{\boldsymbol{\phi\theta}} \neq \mathbf{0}$, then $\boldsymbol{w}_{\mathrm{o}}$ is a mixture of the elements of $\boldsymbol{w}_*$. This happens in (12.63) when the input sequence $\{x(n)\}$ is not white. In this case, the optimum filter $\boldsymbol{w}_{\mathrm{o}}$ takes advantage of the correlation between the entries of $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ to estimate $\boldsymbol{\theta}$ given $\boldsymbol{\phi}$, and uses these estimated values to obtain a better approximation for d. Sure enough, if you write down the problem of approximating $\boldsymbol{\theta}$ linearly from $\boldsymbol{\phi}$, namely,

$$\boldsymbol{W}_{\mathrm{o}} = \arg\min_{\boldsymbol{W}} \mathrm{E}\left\{\left\|\boldsymbol{\theta} - \boldsymbol{W}^T\boldsymbol{\phi}\right\|^2\right\},$$

you will see that the solution is exactly $\boldsymbol{W}_{\mathrm{o}} = \boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{R}_{\boldsymbol{\phi\theta}}$.

What is important to notice is that in both cases the optimum error v_o is not equal to v:

$$v_o = d - \boldsymbol{w}_o^T \boldsymbol{\phi} = \boldsymbol{w}_*^T \boldsymbol{\phi}_e + v - \boldsymbol{w}_o^T \boldsymbol{\phi}.$$

Partitioning $\boldsymbol{w}_* = \left[\boldsymbol{w}_{*,1}^T \ \boldsymbol{w}_{*,2}^T \right]^T$, with $\boldsymbol{w}_{*,1} \in \mathcal{R}^M$, $\boldsymbol{w}_{*,2} \in \mathcal{R}^{K-M}$, we have

$$\begin{aligned} v_o &= \left(\boldsymbol{w}_{*,1} - \boldsymbol{w}_o\right)^T \boldsymbol{\phi} + \boldsymbol{w}_{*,2}^T \boldsymbol{\theta} + v = \boldsymbol{w}_{*,2}^T \boldsymbol{R}_{\boldsymbol{\phi\theta}}^T \boldsymbol{R}_{\boldsymbol{\phi}}^{-1} \boldsymbol{\phi} + \boldsymbol{w}_{*,2}^T \boldsymbol{\theta} + v \\ &= \boldsymbol{w}_{*,2}^T \left(\boldsymbol{\theta} - \boldsymbol{R}_{\boldsymbol{\phi\theta}}^T \boldsymbol{R}_{\boldsymbol{\phi}}^{-1} \boldsymbol{\phi} \right) + v. \end{aligned}$$

Although $v_o \neq v$, you may check that indeed $\mathrm{E}\{v_o \boldsymbol{\phi}\} = \boldsymbol{0}$.

As another example, assume that we have two variables such that $d = ax + bx^3 + v$, in which v has zero mean and is independent of x, and a and b are constants. Assume that we choose as regressor simply $\boldsymbol{\phi} = x$, so we try to solve

$$w_o = \arg\min_w \mathrm{E}\left\{ (d - wx)^2 \right\}.$$

In this case we have

$$R_\phi = \mathrm{E}\{x^2\}, \quad r_{d\phi} = \mathrm{E}\{xd\} = \mathrm{E}\{ax^2 + bx^4\},$$

since $\mathrm{E}\{x^k v\} = \mathrm{E}\{x^k\}\mathrm{E}\{v\} = 0$. The optimum solution is thus

$$w_o = \frac{a\mathrm{E}\{x^2\} + b\mathrm{E}\{x^4\}}{\mathrm{E}\{x^2\}},$$

and the optimum error is

$$v_o = d - w_o x = ax + bx^3 + v - \frac{a\mathrm{E}\{x^2\} + b\mathrm{E}\{x^4\}}{\mathrm{E}\{x^2\}} x = v + b\left(x^2 - \frac{\mathrm{E}\{x^4\}}{\mathrm{E}\{x^2\}} \right) x \neq v.$$

However, v_o is again orthogonal to x (but not to x^3, which we did not include in the regressor).

What happens in the opposite situation, when we include more entries in $\boldsymbol{\phi}$ than necessary? As one would expect, in this case the parameters related to these unnecessary entries will be zeroed in the optimum solution $\boldsymbol{w}_o$. This does not mean, however, that we should hasten to increase $\boldsymbol{\phi}$ to make $\mathcal{F}$ as large as possible, and leave to the filter the task of zeroing whatever was not necessary. This is because increasing the number of parameters to be estimated has a cost. The first and more obvious cost is in terms of memory and number of computations. However, we saw in Section 1.12.1.2 that an actual adaptive filter usually does not converge to the exact optimum solution, but rather hovers around it. Because of this, increasing the number of parameters may in fact decrease the quality of the solution. We explain this in more detail in Section 1.12.4.

1.12.2.1.4 *Zero and non-zero mean variables*

Up to now we did not assume anything about the means of d and $\boldsymbol{\phi}$. If both d and $\boldsymbol{\phi}$ have zero mean, we can interpret all autocorrelations as variances or covariance matrices, according to the case. To see this, assume that

$$\mathrm{E}\{d\} = 0, \quad \mathrm{E}\{\boldsymbol{\phi}\} = \boldsymbol{0}. \tag{12.64}$$

Then $\mathrm{E}\{\hat{y}\} = \mathrm{E}\{\boldsymbol{w}_\mathrm{o}^T\boldsymbol{\phi}\} = 0$, and thus

$$\mathrm{E}\{v_\mathrm{o}\} = \mathrm{E}\{d\} - \boldsymbol{w}_\mathrm{o}^T\mathrm{E}\{\boldsymbol{\phi}\} = 0 + \boldsymbol{w}_\mathrm{o}^T\boldsymbol{0} = 0,$$

so the optimum error has zero mean. In this case, $r_d = \mathrm{E}\{d^2\} = \sigma_d^2$ is the variance of d, and $\mathrm{E}\{v_\mathrm{o}^2\} = \sigma_{v,\mathrm{o}}^2$ is the variance of v_o, so (12.55) becomes

$$J_{\min} = \sigma_{v,\mathrm{o}}^2 = \mathrm{E}\{v_\mathrm{o}^2\} = \sigma_d^2 - \boldsymbol{r}_{d\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}. \tag{12.65}$$

Since $\boldsymbol{R}_{\boldsymbol{\phi}}$ is positive-definite, we conclude that the uncertainty in v_o (its variance) is never larger than the uncertainty in d (Box 3).

However, v_o may have a nonzero mean if either d or $\boldsymbol{\phi}$ has nonzero mean. This is verified as follows. Define

$$\bar{d} = \mathrm{E}\{d\}, \quad \bar{\boldsymbol{\phi}} = \mathrm{E}\{\boldsymbol{\phi}\}, \tag{12.66}$$

then

$$\mathrm{E}\{v_\mathrm{o}\} = \mathrm{E}\{d - \boldsymbol{r}_{d\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi}\} = \bar{d} - \boldsymbol{r}_{d\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}} \neq 0$$

in general (for example, if $\bar{\boldsymbol{\phi}} = \boldsymbol{0}$, but $\bar{d} \neq 0$).

In many applications, v_o should approximate some signal that is constrained to have zero mean, such as speech or noise, or because v_o will be passed through a system with a high DC gain (such as an integrator), so it may be useful to guarantee that v_o has zero mean. How can this be done if the means of d and $\boldsymbol{\phi}$ are not zero? This problem is quite common, particularly when $\boldsymbol{\phi}$ includes nonlinear functions of $x(n)$. Fortunately, there are two easy solutions: First, if one knew the means of d and $\boldsymbol{\phi}$, one could define zero-mean variables

$$d_\mathrm{c} = d - \mathrm{E}\{d\}, \quad \boldsymbol{\phi}_\mathrm{c} = \boldsymbol{\phi} - \mathrm{E}\{\boldsymbol{\phi}\}, \tag{12.67}$$

and use them instead of the original variables in (12.47). This is the simplest solution, but we often do not know the means $\bar{d}$ and $\bar{\boldsymbol{\phi}}$.

The second solution is used when the means are not known: simply add an extra term to $\boldsymbol{\phi}$, defining an extended regressor (not to be confused with the extended regressor from Section 1.12.2.1.3)

$$\boldsymbol{\phi}_\mathrm{e} = \begin{bmatrix} 1 \\ \boldsymbol{\phi} \end{bmatrix}, \tag{12.68}$$

and an extended weight vector $\boldsymbol{w}_\mathrm{e} \in \mathcal{R}^{M+1}$. The first coefficient of $\boldsymbol{w}_\mathrm{e}$ will take care of the means, as we show next. The extended cross-correlation vector and autocorrelation matrix are

$$\boldsymbol{r}_{d\boldsymbol{\phi}_\mathrm{e}} = \begin{bmatrix} \bar{d} \\ \boldsymbol{r}_{d\boldsymbol{\phi}} \end{bmatrix}, \quad \boldsymbol{R}_{\boldsymbol{\phi}_\mathrm{e}} = \begin{bmatrix} 1 & \bar{\boldsymbol{\phi}}^T \\ \bar{\boldsymbol{\phi}} & \boldsymbol{R}_{\boldsymbol{\phi}} \end{bmatrix}.$$

Assuming that $\boldsymbol{R}_{\boldsymbol{\phi}_\mathrm{e}}$ and $\boldsymbol{R}_{\boldsymbol{\phi}}$ are nonsingular, we can compute $\boldsymbol{w}_{\mathrm{o,e}}$ if we evaluate the inverse of $\boldsymbol{R}_{\boldsymbol{\phi}_\mathrm{e}}$ using Schur complements (See Fact 3 in Box 3), as follows

$$\begin{aligned}
\boldsymbol{w}_{\mathrm{o,e}} &= \boldsymbol{R}_{\boldsymbol{\phi}_\mathrm{e}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}_\mathrm{e}} \\
&= \begin{bmatrix} 1 & \boldsymbol{0}^T \\ -\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} (1-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}})^{-1} & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{R}_{\boldsymbol{\phi}}^{-1} \end{bmatrix} \begin{bmatrix} 1 & -\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \bar{d} \\ \boldsymbol{r}_{d\boldsymbol{\phi}} \end{bmatrix} \\
&= \begin{bmatrix} 1 & \boldsymbol{0}^T \\ -\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} (1-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}})^{-1} & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{R}_{\boldsymbol{\phi}}^{-1} \end{bmatrix} \begin{bmatrix} \bar{d}-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}} \\ \boldsymbol{r}_{d\boldsymbol{\phi}} \end{bmatrix} \\
&= \begin{bmatrix} 1 & \boldsymbol{0}^T \\ -\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \frac{\bar{d}-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}}{1-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}}} \\ \boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}} \end{bmatrix} = \begin{bmatrix} \frac{\bar{d}-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}}{1-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}}} \\ \boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}} - \frac{\bar{d}-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}}{1-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}}}\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}} \end{bmatrix}.
\end{aligned}$$

Using this result, we can evaluate now $\mathrm{E}\{v_{\mathrm{o,e}}\} = \mathrm{E}\{d\} - \boldsymbol{w}_{\mathrm{o,e}}^T\mathrm{E}\{\boldsymbol{\phi}_\mathrm{e}\}$ as follows (we used the fact that the inverse of a symmetric matrix is also symmetric from Box 3):

$$\begin{aligned}
\mathrm{E}\{v_{\mathrm{o,e}}\} &= \bar{d} - \frac{\bar{d}-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}}{1-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}}} - \boldsymbol{r}_{d\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}} + \frac{\bar{d}-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}}{1-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}}}\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}} \\
&= \bar{d} - \boldsymbol{r}_{d\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}} - \frac{\bar{d}-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}}{1-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}}}\left(1-\bar{\boldsymbol{\phi}}^T\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\bar{\boldsymbol{\phi}}\right) = 0.
\end{aligned}$$

1.12.2.1.5 *Sufficiently rich signals*

We saw in Section 1.12.2.1.2 that even when $\boldsymbol{R}_{\boldsymbol{\phi}}$ is singular, we can recover the optimum v_o (with probability one), but not the optimum $\boldsymbol{w}_*$. Let us study this problem in more detail, since it is important for system identification. So, what does it mean in practice to have a singular autocorrelation function $\boldsymbol{R}_{\boldsymbol{\phi}}$? The answer to this question can be quite complicated when the input signals are nonstationary (i.e., when $\boldsymbol{R}_{\boldsymbol{\phi}}$ is not constant), so in this case we refer the interested reader to texts in adaptive control (such as [18]), which treat this problem in detail (although usually from a deterministic point of view). We consider here only the case in which the signals $\{d(n), x(n)\}$ are jointly wide-sense stationary.

In this case, $\boldsymbol{R}_{\boldsymbol{\phi}}$ may be singular because some entries of $\boldsymbol{\phi}$ are linear combinations of the others. For example, if we choose $\boldsymbol{\phi}_0 = x(n)$ and $\boldsymbol{\phi}_1 = 2x(n)$, then

$$\boldsymbol{R}_{\boldsymbol{\phi}} = \mathrm{E}\{x^2(n)\}\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix},$$

which is singular.

However, there are more subtle ways in which $\boldsymbol{R}_{\boldsymbol{\phi}}$ may become singular. Assume for example that our input sequence is given by

$$x(n) = \cos(\omega_0 n + \varphi), \tag{12.69}$$

where $0 < \omega_0 \le \pi$ is the frequency, and φ is a random phase, uniformly distributed in the interval $[0, 2\pi)$. This kind of model (sinusoidal signals with random phases), by the way, is the bridge between the periodic signals from Section 1.12.1.2 and the stochastic models discussed in this section. In this case, consider a vector $\boldsymbol{\phi}$ formed from a delay line with $x(n)$ as input,

$$\boldsymbol{\phi} = [\, x(n)\ x(n-1)\ \dots\ x(n-M+1)\,]^T.$$

If $M = 2$, we have

$$\boldsymbol{R_\phi} = \begin{bmatrix} \mathrm{E}\{x^2(n)\} & \mathrm{E}\{x(n)x(n-1)\} \\ \mathrm{E}\{x(n)x(n-1)\} & \mathrm{E}\{x^2(n-1)\} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & \cos(\omega_0) \\ \cos(\omega_0) & 1 \end{bmatrix}.$$

The determinant of $\boldsymbol{R_\phi}$ is $1 - \cos^2(\omega_0) = \sin^2(\omega_0) \neq 0$ if $\omega_0 \neq 0$ and $\omega_0 \neq \pi$.

However, for any $M \ge 3$, the resulting $\boldsymbol{R_\phi}$ will be singular: we can write

$$\boldsymbol{\phi} = \frac{1}{2}e^{j(\omega_0 n+\varphi)}\begin{bmatrix} 1 \\ e^{-j\omega_0} \\ \vdots \\ e^{-j(M-1)\omega_0} \end{bmatrix} + \frac{1}{2}e^{-j(\omega_0 n+\varphi)}\begin{bmatrix} 1 \\ e^{j\omega_0} \\ \vdots \\ e^{j(M-1)\omega_0} \end{bmatrix},$$

and since $\mathrm{E}\{(e^{j\omega_0 n+j\varphi})^2\} = \mathrm{E}\{e^{j2(\omega_0 n+\varphi)}\} = 0$, $\mathrm{E}\{e^{j(\omega_0 n+\varphi)}e^{-j(\omega_0 n+\varphi)}\} = 1$, we have

$$\boldsymbol{R_\phi} = \frac{1}{4}\begin{bmatrix} 1 \\ e^{-j\omega_0} \\ \vdots \\ e^{-j(M-1)\omega_0} \end{bmatrix}\begin{bmatrix} 1 & e^{j\omega_0} & \dots & e^{j(M-1)\omega_0} \end{bmatrix} + \frac{1}{4}\begin{bmatrix} 1 \\ e^{j\omega_0} \\ \vdots \\ e^{j(M-1)\omega_0} \end{bmatrix}\begin{bmatrix} 1 & e^{-j\omega_0} & \dots & e^{-j(M-1)\omega_0} \end{bmatrix}.$$

Since $\boldsymbol{R_\phi}$ is the sum of two rank-one matrices, its rank is at most two. It will be two when $0 < \omega_0 < \pi$, since in this case the vectors are linearly independent. We conclude that when the input signal is a sinusoid as in (12.69), the optimum solution to (12.47) will never be able to identify more than two coefficients. This argument can be generalized: if $x(n)$ is composed of K sinusoids of different frequencies, then $\boldsymbol{R_\phi}$ will be nonsingular if and only if $M \le 2K$.

The general conclusion is that $\boldsymbol{R_\phi}$ will be singular or not, depending on which functions are chosen as inputs, and on the input signal. We say that the input is *sufficiently rich* for a given problem if it is such that $\boldsymbol{R_\phi}$ is nonsingular. We show a few examples next.

1.12.2.1.6 Examples

The concepts we saw in Sections 1.12.2.1.2–1.12.2.1.5 can be understood intuitively if we return to our example in Section 1.12.1.2. In that example, the true echo was obtained by filtering the input $x(n)$ by an unknown $H(z)$, so we had

$$d(n) = \sum_{k=0}^{K} h_k x(n-k) + v(n),$$

where K is the true order of the echo path, $v(n) = B\cos(\omega_1 n + \theta)$ and $x(n) = A\cos(\omega_0 n + \varphi)$. We considered only one harmonic for simplicity.

We are trying to cancel the echo, by constructing an approximated transfer function $\widehat{H}(z)$, which is modeled as an FIR filter with M coefficients. As we saw in Section 1.12.1.2, if $x(n)$ is a simple sinusoid, the echo will be completely canceled if we satisfy (12.8), reproduced below

$$\text{Re}\{\widehat{H}(e^{j\omega_0})\} = \text{Re}\{H(e^{j\omega_0})\}, \quad \text{Im}\{\widehat{H}(e^{j\omega_0})\} = \text{Im}\{H(e^{j\omega_0})\}.$$

If $K = M = 1$, so $H(z) = h_0 + h_1 z^{-1}$ and $\widehat{H}(z) = w_0 + w_1 z^{-1}$, this condition becomes

$$\begin{aligned} w_0 + w_1\cos(-\omega_0) &= h_0 + h_1\cos(-\omega_0), \\ w_1\sin(-\omega_0) &= h_1\sin(-\omega_0), \end{aligned}$$

with a single solution $w_0 = h_0$, $w_1 = h_1$.

On the other hand, if the input were as before (a single sinusoid), but $K = 2$, so $H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2}$, the equations would be

$$\begin{aligned} w_0 + w_1\cos(-\omega_0) &= h_0 + h_1\cos(-\omega_0) + h_2\cos(-2\omega_0), \\ w_1\sin(-\omega_0) &= h_1\sin(-\omega_0) + h_2\sin(-2\omega_0), \end{aligned}$$

and the solution would be unique (since $\boldsymbol{R_\phi}$ is nonsingular), but would not approximate h_0 and h_1 in general.

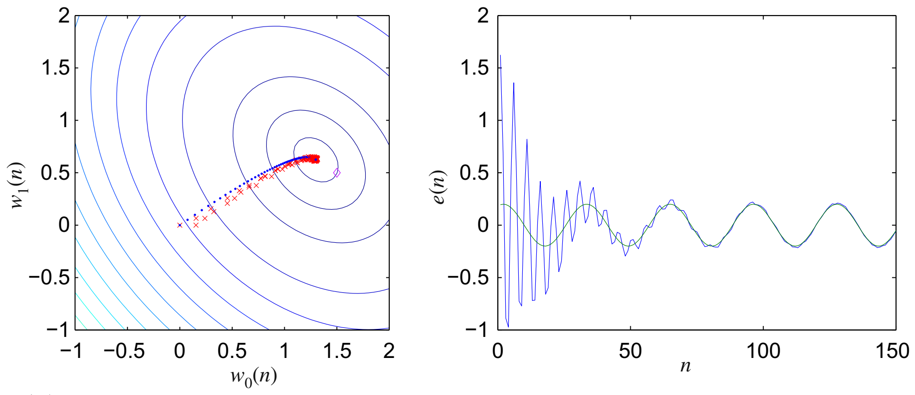
This is an example of an undermodeled system, as we saw in Section 1.12.2.1.3. Figure 12.20 shows the LMS algorithm derived in Section 1.12.1.2 applied to this problem. Note that, even though $w_0(n)$, $w_1(n)$ do not converge to h_0, h_1, in this example the error $e(n)$ does approximate well $v(n)$. This happens because in this case $x(n)$ is periodic, and the term $x(n-2)$ can be obtained exactly as a linear combination of $x(n)$ and $x(n-1)$ (Check that using the expressions for v_0 from Section 1.12.2.1.3, we obtain $v_0 = v$ in this case.)

If we tried to identify $H(z)$ by adding an extra coefficient to $\widehat{H}(z)$, the equations would be

$$\begin{aligned} w_0 + w_1\cos(-\omega_0) + w_2\cos(-2\omega_0) &= h_0 + h_1\cos(-\omega_0) + h_2\cos(-2\omega_0), \\ w_1\sin(-\omega_0) + w_2\sin(-2\omega_0) &= h_1\sin(-\omega_0) + h_2\sin(-2\omega_0). \end{aligned} \tag{12.70}$$

Now $w_0 = h_0$, $w_1 = h_1$, and $w_2 = h_2$ is a solution, but not the only one. Depending on the initial condition, the filter would converge to a different solution, as can be seen in Figure 12.21 (the level sets are not shown, because they are not closed curves in this case). The input is not rich enough (in harmonics) to excite the unknown system $H(z)$ so that the adaptive filter might identify it, as we saw in Section 1.12.2.1.5. However, for all solutions the error converges to $v(n)$.

If the input had a second harmonic ($K_0 = 2$), we would add two more equations to (12.70), corresponding to $\widehat{H}(e^{2j\omega_0}) = H(e^{2j\omega_0})$, and the filter would be able to identify systems with up to four coefficients.

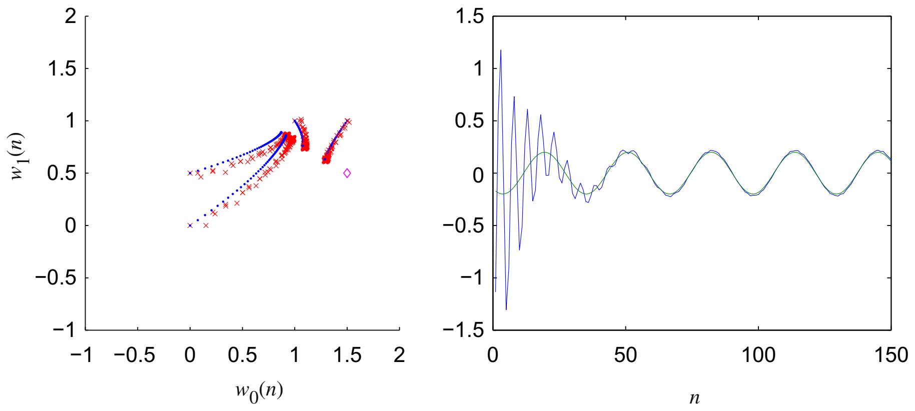


(a) LMS trajectory in (w0,w1)-space.The diamond marks the true value of the first two terms in $H(z)$.

(b) Output error

FIGURE 12.20

LMS in undermodeled case.



(a) LMS trajectory in ($w0,w1$)-space.The diamond marks the true value of the first two terms in $H(z)$.

(b) Output error

FIGURE 12.21

LMS under non-sufficiently rich input.

The important message is that the adaptive filter only "sees" the error $d(n) - \hat{y}(n)$. If a mismatch between the true echo path $H(z)$ and the estimated one $\widehat{H}(z)$ is not observable through looking only at the error signal, the filter will not converge to $H(z)$. Whether a mismatch is observable or not through the error depends on both $H(z)$ and the input signal being used: the input must excite a large enough number of frequencies so that $H(z)$ can be estimated correctly. For this reason, in system identification a good input would be white noise.

1.12.2.2 Complex variables and multi-channel filtering

Adaptive filters used in important applications such as communications and radar have complex variables as input signals. In this section, we extend the results of Section 1.12.2.1 to consider complex signals. We start with the general solution, then restrict the result to the more usual case of circular signals.

We first show that the general solution for complex signals is equivalent to an adaptive filter with two inputs and two outputs: Indeed, if all input variables are complex, then the error $e(n) = d(n) - \hat{y}(n)$ will also be complex, that is, $e(n) = e_{\rm r}(n) + je_{\rm i}(n)$, where $e_{\rm r}(n), e_{\rm i}(n) \in \mathcal{R}$ are the real and imaginary parts of $e(n)$. We must then minimize $\mathrm{E}\{|e(n)|^2\} = \mathrm{E}\{e_{\rm r}^2(n)\} + \mathrm{E}\{e_{\rm i}^2(n)\}$, the total power of the complex signal $e(n)$. The quadratic cost function (12.45) must be changed to

$$J(\boldsymbol{w}) = \mathrm{E}\{|e(n)|^2\} = \mathrm{E}\{e(n)e^*(n)\} = \mathrm{E}\{e_{\rm r}^2(n)\} + \mathrm{E}\{e_{\rm i}^2(n)\},$$

where $e^*(n)$ is the complex conjugate of $e(n)$. Now, we need to be careful about what exactly is the coefficient vector $\boldsymbol{w}$ in this case: If the regressor is $\boldsymbol{\phi} = \boldsymbol{\phi}_{\rm r} + j\boldsymbol{\phi_i}$ and $d = d_{\rm r} + jd_{\rm i}$, in principle $\hat{y} = \hat{y}_{\rm r} + j\hat{y}_{\rm i}$ should be such that both its real and imaginary parts depend on both the real and imaginary parts of $\boldsymbol{\phi}$, that is,

$$\begin{aligned} \hat{y}_{\rm r} &= \boldsymbol{w}_{\rm rr}^T\boldsymbol{\phi}_{\rm r} + \boldsymbol{w}_{\rm ri}^T\boldsymbol{\phi}_{\rm i}, \\ \hat{y}_{\rm i} &= \boldsymbol{w}_{\rm ir}^T\boldsymbol{\phi}_{\rm r} + \boldsymbol{w}_{\rm ii}^T\boldsymbol{\phi}_{\rm i}. \end{aligned} \tag{12.71}$$

Note that there is no *a priori* reason for us to imagine that there should be a special relation between the pairs $(\boldsymbol{w}_{\rm rr}, \boldsymbol{w}_{\rm ri})$ and $(\boldsymbol{w}_{\rm ir}, \boldsymbol{w}_{\rm ii})$. That is, we are dealing with two different coefficient vectors and one extended regressor:

$$\boldsymbol{w}_{\rm r} = \begin{bmatrix} \boldsymbol{w}_{\rm rr} \\ \boldsymbol{w}_{\rm ri} \end{bmatrix}, \quad \boldsymbol{w}_{\rm i} = \begin{bmatrix} \boldsymbol{w}_{\rm ir} \\ \boldsymbol{w}_{\rm ii} \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\phi}_{\rm r} \\ \boldsymbol{\phi}_{\rm i} \end{bmatrix}. \tag{12.72}$$

Let us proceed for the time being separating the real and imaginary channels of our filter. Later, in Section 1.12.2.2.1, we return to complex algebra. With these definitions, we can write

$$\hat{y}_{\rm r} = \boldsymbol{w}_{\rm r}^T\boldsymbol{\theta}, \quad \hat{y}_{\rm i} = \boldsymbol{w}_{\rm i}^T\boldsymbol{\theta}. \tag{12.73}$$

The cost function then becomes

$$\begin{aligned} J(\boldsymbol{w}_{\rm r}, \boldsymbol{w}_{\rm i}) &= \mathrm{E}\{(d_{\rm r} - \boldsymbol{w}_{\rm r}^T\boldsymbol{\theta})^2\} + \mathrm{E}\{(d_{\rm i} - \boldsymbol{w}_{\rm i}^T\boldsymbol{\theta})^2\} \\ &= \mathrm{E}\{d_{\rm r}^2 - 2\boldsymbol{w}_{\rm r}^T\boldsymbol{\theta}d_{\rm r} + \boldsymbol{w}_{\rm r}^T\boldsymbol{\theta}\boldsymbol{\theta}^T\boldsymbol{w}_{\rm r}\} + \mathrm{E}\{d_{\rm i}^2 - 2\boldsymbol{w}_{\rm i}^T\boldsymbol{\theta}d_{\rm i} + \boldsymbol{w}_{\rm i}^T\boldsymbol{\theta}\boldsymbol{\theta}^T\boldsymbol{w}_{\rm i}\} \\ &= r_{d_{\rm r}} - 2\boldsymbol{w}_{\rm r}^T\boldsymbol{r}_{d_{\rm r}\boldsymbol{\theta}} + \boldsymbol{w}_{\rm r}^T\boldsymbol{R}_{\boldsymbol{\theta}}\boldsymbol{w}_{\rm r} + r_{d_{\rm i}} - 2\boldsymbol{w}_{\rm i}^T\boldsymbol{r}_{d_{\rm i}\boldsymbol{\theta}} + \boldsymbol{w}_{\rm i}^T\boldsymbol{R}_{\boldsymbol{\theta}}\boldsymbol{w}_{\rm i}, \end{aligned} \tag{12.74}$$

where the autocorrelation matrix and cross-correlation vectors are given by

$$\boldsymbol{R}_{\boldsymbol{\theta}} = \begin{bmatrix} \boldsymbol{R}_{\rm r} & \boldsymbol{R}_{\rm ri} \\ \boldsymbol{R}_{\rm ir} & \boldsymbol{R}_{\rm i} \end{bmatrix}, \quad \boldsymbol{r}_{d_{\rm r}\boldsymbol{\theta}} = \begin{bmatrix} \boldsymbol{r}_{\rm rr} \\ \boldsymbol{r}_{\rm ri} \end{bmatrix}, \quad \boldsymbol{r}_{d_{\rm i}\boldsymbol{\theta}} = \begin{bmatrix} \boldsymbol{r}_{\rm ir} \\ \boldsymbol{r}_{\rm ii} \end{bmatrix}, \tag{12.75}$$

where we simplified the notation, writing $\boldsymbol{R}_{\rm r}$ instead of $\boldsymbol{R}_{\boldsymbol{\phi}_{\rm r}}$, $\boldsymbol{R}_{\rm ri}$ instead of $\boldsymbol{R}_{\boldsymbol{\phi}_{\rm r}\boldsymbol{\phi}_{\rm i}}$, $\boldsymbol{r}_{\rm rr}$ instead of $\boldsymbol{r}_{d_{\rm r}\boldsymbol{\phi}_{\rm r}}$, and so on.

Differentiating (12.74) to find the minimum, we obtain

$$\frac{\partial J}{\partial \boldsymbol{w}_{\rm r}^T} = -2\boldsymbol{r}_{d_{\rm r}\boldsymbol{\theta}} + 2\boldsymbol{R}_{\boldsymbol{\theta}}\boldsymbol{w}_{\rm r}, \quad \frac{\partial J}{\partial \boldsymbol{w}_{\rm i}^T} = -2\boldsymbol{r}_{d_{\rm i}\boldsymbol{\theta}} + 2\boldsymbol{R}_{\boldsymbol{\theta}}\boldsymbol{w}_{\rm i}. \tag{12.76}$$

Therefore, the optimum filters satisfy

$$\begin{bmatrix} \boldsymbol{R}_{\rm r} & \boldsymbol{R}_{\rm ri} \\ \boldsymbol{R}_{\rm ir} & \boldsymbol{R}_{\rm i} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_{\rm rr,o} \\ \boldsymbol{w}_{\rm ri,o} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{\rm rr} \\ \boldsymbol{r}_{\rm ri} \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{R}_{\rm r} & \boldsymbol{R}_{\rm ri} \\ \boldsymbol{R}_{\rm ir} & \boldsymbol{R}_{\rm i} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_{\rm ir,o} \\ \boldsymbol{w}_{\rm ii,o} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{\rm ir} \\ \boldsymbol{r}_{\rm ii} \end{bmatrix}. \tag{12.77}$$

The optimum error is then

$$v_{r,{\rm o}} = d_r - \left[\, \boldsymbol{w}_{\rm rr,o}^T \; \boldsymbol{w}_{\rm ri,o}^T \,\right]\boldsymbol{\theta}, \quad v_{i,{\rm o}} = d_i - \left[\, \boldsymbol{w}_{\rm ir,o}^T \; \boldsymbol{w}_{\rm ii,o}^T \,\right]\boldsymbol{\theta}. \tag{12.78}$$

There is also an orthogonality condition for this case: using (12.77), we obtain

$$\mathrm{E}\{v_{r,{\rm o}}\boldsymbol{\theta}\} = \mathrm{E}\{d_r\boldsymbol{\theta}\} - \mathrm{E}\{\boldsymbol{\theta}\boldsymbol{\theta}^T\}\begin{bmatrix} \boldsymbol{w}_{\rm rr,o} \\ \boldsymbol{w}_{\rm ri,o} \end{bmatrix} = \mathbf{0}, \tag{12.79a}$$

$$\mathrm{E}\{v_{i,{\rm o}}\boldsymbol{\theta}\} = \mathrm{E}\{d_i\boldsymbol{\theta}\} - \mathrm{E}\{\boldsymbol{\theta}\boldsymbol{\theta}^T\}\begin{bmatrix} \boldsymbol{w}_{\rm ir,o} \\ \boldsymbol{w}_{\rm ii,o} \end{bmatrix} = \mathbf{0}. \tag{12.79b}$$

The value of the cost function at the minimum can be obtained using this result. As in the case of real variables,

$$\begin{aligned} \mathrm{E}\{v_{r,{\rm o}}^2\} &= \mathrm{E}\left\{v_{r,{\rm o}}\left(d_r - \left[\, \boldsymbol{w}_{\rm rr,o}^T \; \boldsymbol{w}_{\rm ri,o}^T \,\right]\boldsymbol{\theta}\right)\right\} \\ &= \mathrm{E}\{v_{r,{\rm o}}d_r\} = \mathrm{E}\left\{d_r^2\right\} - \left[\, \boldsymbol{w}_{\rm rr,o}^T \; \boldsymbol{w}_{\rm ri,o}^T \,\right]\mathrm{E}\{d_r\boldsymbol{\theta}\}, \end{aligned}$$

where we used (12.79a) in the second equality. Doing similarly for $v_{\rm i,o}$, we obtain

$$r_{v_{\rm r,o}} = \mathrm{E}\{v_{r,{\rm o}}^2\} = r_{d_{\rm r}} - \left[\, \boldsymbol{w}_{\rm rr,o}^T \; \boldsymbol{w}_{\rm ri,o}^T \,\right]\boldsymbol{r}_{d_{\rm r}\boldsymbol{\theta}}, \tag{12.80a}$$

$$r_{v_{\rm i,o}} = \mathrm{E}\{v_{\rm i,o}^2\} = r_{d_{\rm i}} - \left[\, \boldsymbol{w}_{\rm ir,o}^T \; \boldsymbol{w}_{\rm ii,o}^T \,\right]\boldsymbol{r}_{d_i\boldsymbol{\theta}}, \tag{12.80b}$$

$$J_{\min} = r_{v_{\rm r,o}} + r_{v_{\rm i,o}}. \tag{12.80c}$$

What we have just described is a two-input/two-output optimal filter. In the next section, we re-derive it using complex algebra. Note that multi-channel filters have applications that are unrelated to complex variables, particularly in control [18], stereo echo cancellation [19,20], and active noise control [21–24].

1.12.2.2.1 *Widely-linear complex least-mean squares*

This general solution can be obtained in a more straightforward way if we use the complex gradients described in Box 6. Keeping the regressor $\boldsymbol{\theta}$ as in (12.72) and defining the extended complex weight vector as $\boldsymbol{\omega} = \boldsymbol{w}_{\rm r} + j\boldsymbol{w}_{\rm i}$, we can write the cost function as

$$J(\boldsymbol{\omega}) = \mathrm{E}\left\{(d - \boldsymbol{\omega}^H\boldsymbol{\theta})(d - \boldsymbol{\omega}^H\boldsymbol{\theta})^*\right\} = r_d - \boldsymbol{\omega}^H\boldsymbol{r}_{d\boldsymbol{\theta}} - \boldsymbol{r}_{d\boldsymbol{\theta}}^H\boldsymbol{\omega} + \boldsymbol{\omega}^H\boldsymbol{R}_{\boldsymbol{\theta}}\boldsymbol{\omega},$$

where we defined $r_d = \mathrm{E}\{|d|^2\}$ and $\boldsymbol{r}_{d\boldsymbol{\theta}} = \mathrm{E}\{d^*\boldsymbol{\theta}\}$. Note that $(\cdot)^H$ represents the Hermitian transpose, that is, transpose followed by conjugation. Differentiating $J(\boldsymbol{\omega})$ with respect to ω^H, as described in Box 6, we obtain

$$\frac{\partial J}{\partial \boldsymbol{\omega}^H} = -\boldsymbol{r}_{d\boldsymbol{\theta}} + \boldsymbol{R}_{\boldsymbol{\theta}}\boldsymbol{\omega}. \tag{12.81}$$

The normal equations then become

$$\boldsymbol{R}_{\boldsymbol{\theta}}\boldsymbol{\omega}_{\mathrm{o}} = \boldsymbol{r}_{d\boldsymbol{\theta}}. \tag{12.82}$$

Expanding the real and imaginary parts of (12.82), we recover (12.77). Similarly, the optimum error and the minimum value of the cost are

$$\nu_{\mathrm{o}} = d - \boldsymbol{\omega}_{\mathrm{o}}^H\boldsymbol{\theta}, \quad J_{\min} = \mathrm{E}\{|\nu_{\mathrm{o}}|^2\} = \mathrm{E}\{|d|^2\} - \boldsymbol{\omega}_{\mathrm{o}}^H\boldsymbol{r}_{d\boldsymbol{\theta}}. \tag{12.83}$$

Comparing with (12.80), we see that $\nu_{\mathrm{o}} = v_{\mathrm{r,o}} + jv_{\mathrm{i,o}}$, and the value of $J_{\min}$ is the same as before. However, as you noticed, using a complex weight vector $\boldsymbol{\omega}$ and the complex derivatives of Box 6, all expressions are much simpler to derive.

The filter using a complex weight vector $\boldsymbol{\omega}$ and the real regressor $\boldsymbol{\theta}$ constitutes a version of what is known as *widely-linear* complex filter. This version was recently proposed in [25] as a reduced-complexity alternative to the widely-linear complex filter originally proposed in [26,27], which uses as regressor the extended vector

$$\boldsymbol{\phi}_{\mathrm{e}} = \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{\phi}^* \end{bmatrix}, \tag{12.84}$$

composed of the original regressor $\boldsymbol{\phi}$ and its complex conjugate $\boldsymbol{\phi}^*$ (computational complexity is reduced because the number of operations required to compute $\boldsymbol{\omega}^H\boldsymbol{\theta}$ is about half of what is necessary to evaluate $\boldsymbol{\omega}^H\boldsymbol{\phi}_{\mathrm{e}}$). Widely-linear estimation has been receiving much attention lately, due to new applications that have appeared [28–31].

But why the name *widely* linear? The reason for this name is that the complex regressor $\boldsymbol{\phi}$ does not appear linearly in the expressions, but only by separating its real and imaginary parts (as in $\boldsymbol{\theta}$) or by including its complex conjugate (as in $\boldsymbol{\phi}_{\mathrm{e}}$). Both these expressions, from the point of view of a complex variable, are nonlinear.

What would be a *linear* complex filter, then? This is the subject of the next section.

1.12.2.2.2 *Linear complex least-mean squares*

There are many applications in which the data and regressor are such that their real and imaginary parts are similarly distributed, so that

$$\boldsymbol{R}_{\mathrm{r}} = \boldsymbol{R}_{\mathrm{i}}, \quad r_{d_{\mathrm{r}}} = r_{d_{\mathrm{i}}}, \quad \boldsymbol{R}_{\mathrm{ri}} = -\boldsymbol{R}_{\mathrm{ir}}, \quad \boldsymbol{r}_{\mathrm{rr}} = \boldsymbol{r}_{\mathrm{ii}}, \quad \boldsymbol{r}_{\mathrm{ri}} = -\boldsymbol{r}_{\mathrm{ir}}. \tag{12.85}$$

Under these conditions, we say that the pair $(d, \boldsymbol{\phi})$ is *complex circular*, or simply *circular*.

In this case, we can rewrite (12.77) as

$$\begin{bmatrix} \boldsymbol{R}_{\mathrm{r}} & \boldsymbol{R}_{\mathrm{ri}} \\ -\boldsymbol{R}_{\mathrm{ri}} & \boldsymbol{R}_{\mathrm{r}} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_{\mathrm{rr}} \\ \boldsymbol{w}_{\mathrm{ri}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{\mathrm{rr}} \\ \boldsymbol{r}_{\mathrm{ri}} \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}} & \boldsymbol{R}_{\mathrm{ri}} \\ -\boldsymbol{R}_{\mathrm{ri}} & \boldsymbol{R}_{\mathrm{r}} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_{\mathrm{ir}} \\ \boldsymbol{w}_{\mathrm{ii}} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{r}_{\mathrm{ri}} \\ \boldsymbol{r}_{\mathrm{rr}} \end{bmatrix}.$$

Expand these equations as follows (notice the change in order in (12.86b))

$$\boldsymbol{R}_{\mathrm{r}}\boldsymbol{w}_{\mathrm{rr}} + \boldsymbol{R}_{\mathrm{ri}}\boldsymbol{w}_{\mathrm{ri}} = \boldsymbol{r}_{\mathrm{rr}}, \quad -\boldsymbol{R}_{\mathrm{ri}}\boldsymbol{w}_{\mathrm{rr}} + \boldsymbol{R}_{\mathrm{r}}\boldsymbol{w}_{\mathrm{ri}} = \boldsymbol{r}_{\mathrm{ri}}, \tag{12.86a}$$

$$\boldsymbol{R}_{\mathrm{r}}\boldsymbol{w}_{\mathrm{ii}} - \boldsymbol{R}_{\mathrm{ri}}\boldsymbol{w}_{\mathrm{ir}} = \boldsymbol{r}_{\mathrm{rr}}, \quad \boldsymbol{R}_{\mathrm{ri}}\boldsymbol{w}_{\mathrm{ii}} + \boldsymbol{R}_{\mathrm{r}}\boldsymbol{w}_{\mathrm{ir}} = -\boldsymbol{r}_{\mathrm{ri}}. \tag{12.86b}$$

Now we can see that, if $(\boldsymbol{w}_{\mathrm{rr}}, \boldsymbol{w}_{\mathrm{ri}})$ is a solution to (12.86a), then

$$(\boldsymbol{w}_{\mathrm{ii}} = \boldsymbol{w}_{\mathrm{rr}}, \boldsymbol{w}_{\mathrm{ir}} = -\boldsymbol{w}_{\mathrm{ri}}) \tag{12.87}$$

is a solution to (12.86b). Therefore, when the input signals are circular, the number of degrees of freedom in the problem is actually smaller. In this case, there is a very nice way of recovering the solutions, working only with complex algebra. Indeed, defining $\boldsymbol{w} = \boldsymbol{w}_{\mathrm{rr}} + j\boldsymbol{w}_{\mathrm{ri}}$, we have

$$\begin{aligned}\boldsymbol{w}^H\boldsymbol{\phi} &= (\boldsymbol{w}_{\mathrm{rr}} - j\boldsymbol{w}_{\mathrm{ri}})^T(\boldsymbol{\phi}_{\mathrm{r}} + j\boldsymbol{\phi}_{\mathrm{i}}) \\ &= \left(\boldsymbol{w}_{\mathrm{rr}}^T\boldsymbol{\phi}_{\mathrm{r}} + \boldsymbol{w}_{\mathrm{ri}}^T\boldsymbol{\phi}_{\mathrm{i}}\right) + j\left(\boldsymbol{w}_{\mathrm{rr}}^T\boldsymbol{\phi}_{\mathrm{i}} - \boldsymbol{w}_{\mathrm{ri}}^T\boldsymbol{\phi}_{\mathrm{r}}\right),\end{aligned} \tag{12.88}$$

which is equivalent to (12.71) with the identities (12.87). Using this definition, it is possible to obtain the optimum solution (12.86a), (12.86b), and (12.87) using only complex algebra, following exactly the same steps as our derivation for the real case in the previous section.

This path is very useful, we can derive almost all results for real or complex circular adaptive filters using the same arguments. Let us see how it goes. First, define $\hat{y} = \boldsymbol{w}^H\boldsymbol{\phi}$, so our problem becomes

$$\min_{\boldsymbol{w}\in\mathcal{C}^M} \mathrm{E}\left\{\left|d - \boldsymbol{w}^H\boldsymbol{\phi}\right|^2\right\}. \tag{12.89}$$

Expanding the cost, recalling that now $\left|d - \boldsymbol{w}^H\boldsymbol{\phi}\right|^2 = (d - \boldsymbol{w}^H\boldsymbol{\phi})(d - \boldsymbol{w}^H\boldsymbol{\phi})^* = (d - \boldsymbol{w}^H\boldsymbol{\phi})(d^* - \boldsymbol{\phi}^H\boldsymbol{w})$, we obtain

$$\begin{aligned}J(\boldsymbol{w}) &= \mathrm{E}\left\{|d|^2 - \boldsymbol{w}^H\boldsymbol{\phi}d^* - d\boldsymbol{\phi}^H\boldsymbol{w} + \boldsymbol{w}^H\boldsymbol{\phi}\boldsymbol{\phi}^H\boldsymbol{w}\right\} \\ &= r_d - \boldsymbol{w}^H\boldsymbol{r}_{d\boldsymbol{\phi}} - \boldsymbol{r}_{d\boldsymbol{\phi}}^H\boldsymbol{w} + \boldsymbol{w}^H\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{w},\end{aligned} \tag{12.90}$$

where we defined $r_d = \mathrm{E}\{|d|^2\}$, $\boldsymbol{r}_{d\boldsymbol{\phi}} = \mathrm{E}\{d^*\boldsymbol{\phi}\}$, $\boldsymbol{R}_{\boldsymbol{\phi}} = \mathrm{E}\{\boldsymbol{\phi}\boldsymbol{\phi}^H\}$.

We need to find the minimum of this cost.One easy solution is to use the rules for differentiation of real functions of complex variables, as described in Box 6. These rules are very useful, and surprisingly easy to use: basically, we can treat $\boldsymbol{w}$ and its conjugate $\boldsymbol{w}^H$ as if they were independent variables, that is

$$\frac{\partial J}{\partial \boldsymbol{w}^H} = -\boldsymbol{r}_{d\boldsymbol{\phi}} + \boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{w}. \tag{12.91}$$

Equating the gradient to zero, the solution is (assuming that $\boldsymbol{R}_{\boldsymbol{\phi}}$ is nonsingular)

$$\boldsymbol{w}_{\mathrm{o}} = \boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}, \tag{12.92}$$

which is exactly equal to (12.51). This is the advantage of this approach: the expressions for complex and real problems are *almost* equal. The important differences are the conjugates that appear in the definitions of $\boldsymbol{R}_{\boldsymbol{\phi}}$ and $\boldsymbol{r}_{d\boldsymbol{\phi}}$, and the absence of a factor of 2 in the gradient (12.91). This follows from the definition of complex derivative we used (see Box 6). Here this difference is canceled out in the solution (12.92), but further on we will find situations in which there will be a different factor for the case of real and for the case of complex variables.

Let us check that (12.92) is really equivalent to (12.86a). Expanding $\boldsymbol{R}_{\boldsymbol{\phi}}$ and $\boldsymbol{r}_{d\boldsymbol{\phi}}$ and using the circularity conditions (12.85), we obtain

$$\boldsymbol{R}_{\boldsymbol{\phi}} = \mathrm{E}\{\boldsymbol{\phi}\boldsymbol{\phi}^H\} = \mathrm{E}\{(\boldsymbol{\phi}_\mathrm{r} + j\boldsymbol{\phi}_\mathrm{i})(\boldsymbol{\phi}_\mathrm{r} - j\boldsymbol{\phi}_\mathrm{i})^T\} = \boldsymbol{R}_\mathrm{r} + \boldsymbol{R}_\mathrm{i} + j\left(\boldsymbol{R}_\mathrm{ir} - \boldsymbol{R}_\mathrm{ri}\right), \tag{12.93a}$$
$$= 2\boldsymbol{R}_\mathrm{r} - 2j\boldsymbol{R}_\mathrm{ri}, \tag{12.93b}$$

and

$$\boldsymbol{r}_{d\boldsymbol{\phi}} = \mathrm{E}\{d^*\boldsymbol{\phi}\} = \mathrm{E}\{(d_\mathrm{r} - jd_\mathrm{i})(\boldsymbol{\phi}_\mathrm{r} + j\boldsymbol{\phi}_\mathrm{i})\} = \boldsymbol{r}_\mathrm{rr} + \boldsymbol{r}_\mathrm{ii} + j\left(\boldsymbol{r}_\mathrm{ri} - \boldsymbol{r}_\mathrm{ir}\right)$$
$$= 2\boldsymbol{r}_\mathrm{rr} + 2j\boldsymbol{r}_\mathrm{ri}, \tag{12.94}$$

so, equating the gradient (12.91) to zero, $\boldsymbol{w}_\mathrm{o} = \boldsymbol{w}_{\mathrm{r,o}} + j\boldsymbol{w}_{\mathrm{i,o}}$ must satisfy

$$2\left(\boldsymbol{R}_\mathrm{r} - j\boldsymbol{R}_\mathrm{ri}\right)\left(\boldsymbol{w}_{\mathrm{r,o}} + j\boldsymbol{w}_{\mathrm{i,o}}\right) = 2\left(\boldsymbol{r}_\mathrm{rr} + j\boldsymbol{r}_\mathrm{ri}\right),$$
$$\boldsymbol{R}_\mathrm{r}\boldsymbol{w}_{\mathrm{r,o}} + \boldsymbol{R}_\mathrm{ri}\boldsymbol{w}_{\mathrm{i,o}} + j\left(\boldsymbol{R}_\mathrm{r}\boldsymbol{w}_{\mathrm{i,o}} - \boldsymbol{R}_\mathrm{ri}\boldsymbol{w}_{\mathrm{r,o}}\right) = \boldsymbol{r}_\mathrm{rr} + j\boldsymbol{r}_\mathrm{ri}. \tag{12.95}$$

Comparing the real and imaginary parts of (12.95) with (12.86a), we see that they indeed correspond to the same set of equations.

A few important remarks:

Remark 5. The orthogonality condition still holds in the complex circular case, but with a small difference: defining the optimum error as before,

$$v_\mathrm{o} = d - \boldsymbol{w}_\mathrm{o}^H\boldsymbol{\phi},$$

we obtain, noting that $(\boldsymbol{w}_\mathrm{o}^H\boldsymbol{\phi})^* = \boldsymbol{\phi}^H\boldsymbol{w}_\mathrm{o}$,

$$\mathrm{E}\{v_\mathrm{o}^*\boldsymbol{\phi}\} = \mathrm{E}\{d^*\boldsymbol{\phi} - \boldsymbol{\phi}\boldsymbol{\phi}^H\boldsymbol{w}_\mathrm{o}\} = \boldsymbol{r}_{d\boldsymbol{\phi}} - \boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{w}_\mathrm{o} = \boldsymbol{0}. \tag{12.96}$$

Remark 6. The complex gradient (12.91) seems strange because it does not include the factors of 2 that would appear in the real case. This difference will appear in many expressions in the following sections, whenever we need to differentiate between the real and complex cases. However, the difference is illusory: if you go back to (12.93b) and (12.94), you will see that the factor of two is actually there, just hidden by the definition of autocorrelation and cross-correlation under the circularity conditions.

Remark 7. The optimum error power can be obtained as in the real case, and is equal to

$$\mathrm{E}\{|v_\mathrm{o}|^2\} = r_d - \boldsymbol{w}_\mathrm{o}^H\boldsymbol{r}_{d\boldsymbol{\phi}} = r_d - \boldsymbol{r}_{d\boldsymbol{\phi}}^H\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}}. \tag{12.97}$$

Box 4: Least mean-squares estimation

Here we consider the general problem of finding the optimum relation $\hat{\mathcal{H}}_\text{o}$ between $d(n)$ and $x(n)$, such that

$$\hat{\mathcal{H}}_\text{o} = \arg\min_{\hat{\mathcal{H}}} \mathrm{E}\left\{d(n) - \hat{\mathcal{H}}(x(n), x(n-1), \ldots)\right\}, \qquad (12.98)$$

without restricting $\mathcal{H}$ to a particular class.

In order to understand well this problem, let us consider first the simpler case where the relation between $x(n)$ and $d(n)$ is memoryless, that is, our function depends only on the current value of $x(n)$. Dropping the time index to simplify the notation, our problem then becomes how to find a function $\hat{\mathcal{H}}$ that, applied to a certain random variable x, approximates a certain random variable d so that the mean-square error is minimized, that is, we want to solve

$$\min_{\hat{\mathcal{H}}} \mathrm{E}\{[d - \hat{\mathcal{H}}(x)]^2\}. \qquad (12.99)$$

Assume that the joint probability density function $p_{dx}(d, x)$ of d and x is known (we are assuming that d and x are continuous random variables, but the final result in (12.100) also holds if they are discrete). Then,

$$\begin{aligned}\mathrm{E}\{[d - \hat{\mathcal{H}}(x)]^2\} &= \int\int [d - \hat{\mathcal{H}}(x)]^2 p_{dx}(d, x)\mathrm{d}\,d\,\mathrm{d}x \\ &= \int \underbrace{\left\{\int [d - \hat{\mathcal{H}}(x)]^2 p_{d|x}(d|x)\mathrm{d}\,d\right\}}_{\mathrm{E}_d\{[d-\hat{\mathcal{H}}(x)]^2|x\}} p_x(x)\mathrm{d}x \\ &= \mathrm{E}_x\left\{\mathrm{E}_d\left\{[d - \hat{\mathcal{H}}(x)]^2|x\right\}\right\}, \qquad (12.100)\end{aligned}$$

where we use the subscripts E_d and E_x to highlight that the expectations are taken with respect to d or x only.

Note that the inner integrand $\mathrm{E}_d\{[d-\hat{\mathcal{H}}(x)]^2|x\}$ in (12.100) is nonnegative. Therefore, if we minimize it for each value of x, we will obtain the minimum of the full expression. That is, the optimum $\hat{\mathcal{H}}_\text{o}$ is the function defined by

$$\hat{\mathcal{H}}_\text{o}(x) = \arg\min_{\hat{\mathcal{H}}(x)} \mathrm{E}_d\left\{[d - \hat{\mathcal{H}}(x)]^2|x\right\}. \qquad (12.101)$$

It is important to remember that, fixed x, $\hat{\mathcal{H}}(x)$ is simply a number. This can be seen more easily expanding the expectations in (12.101)

$$\begin{aligned}\mathrm{E}_d\left\{[d - \hat{\mathcal{H}}(x)]^2|x\right\} &= \int \left(d^2 - 2d\hat{\mathcal{H}}(x) + \hat{\mathcal{H}}^2(x)\right) p_{d|x}(d|x)\mathrm{d}d \\ &= \int d^2 p_{d|x}(d|x)\mathrm{d}d - 2\hat{\mathcal{H}}(x)\int d p_{d|x}(d|x)\mathrm{d}d + \hat{\mathcal{H}}^2(x)\int p_{d|x}(d|x)\mathrm{d}d \\ &= \mathrm{E}_d\{d^2|x\} - 2\mathrm{E}\{d|x\}\hat{\mathcal{H}}(x) + \hat{\mathcal{H}}^2(x).\end{aligned}$$

Differentiating with respect to $\hat{\mathcal{H}}(x)$ and equating to zero, we obtain the solution

$$\hat{\mathcal{H}}_o(x) = \mathrm{E}_d\{d|x\} \triangleq \hat{y}(n), \tag{12.102}$$

which is indeed a function of x, as desired.

Let us study the properties of this solution. First, define

$$v_o = d - \hat{\mathcal{H}}_o(x).$$

Note that v_o and $\hat{\mathcal{H}}_o$ are not necessarily equal to v and $\mathcal{H}$ from (12.40). Evaluating the average of $\hat{\mathcal{H}}_o(x)$, we obtain

$$\mathrm{E}_x\{\hat{\mathcal{H}}_o(x)\} = \mathrm{E}_x\left\{\mathrm{E}_d\{d|x\}\right\} = \mathrm{E}\{d\},$$

by an argument similar to that used to obtain (12.100). We conclude that v_o has zero mean. In addition, for any function $f(x)$ of x for which the expected values exist, we have

$$\mathrm{E}\{v_o f(x)\} = \mathrm{E}_x\left\{\mathrm{E}_d\{df(x)\}\right\} - \mathrm{E}\left\{\mathrm{E}_d\{d|x\}f(x)\right\} = 0 = \mathrm{E}\{v_o\}\mathrm{E}\{f(x)\}, \tag{12.103}$$

since $\mathrm{E}_d\{df(x)|x\} = f(x)\mathrm{E}_d\{d|x\}$. This result means that the error v_o is uncorrelated to any function of the reference data x, which is a very strong condition. It basically means that we have extracted all second-order information available in x about d.

We remark that (12.103) does not imply that x and v_o are independent. A simple counter-example is the following: consider two discrete-valued random variables r and s with joint probabilities given by Table 12.1. Then $\mathrm{E}\{sf(r)\} = 1/4\cdot(-1)\cdot f(+1) + 1/4\cdot(+1)\cdot f(+1) + 1/2\cdot 0\cdot f(-1) = 0, \forall f(\cdot)$, but r and s are not independent.

In the more general case in which $\hat{\mathcal{H}}$ is allowed to depend on previous samples of $x(n)$, the arguments and the final conclusion are similar:

$$\arg\min_{\hat{\mathcal{H}}} \mathrm{E}\left\{\left[d - \hat{\mathcal{H}}(x(n), x(n-1), \ldots)\right]^2\right\} = \mathrm{E}\left\{d|\, x(n), x(n-1), \ldots\right\}, \tag{12.104}$$

and defining

$$v_o(n) = d(n) - \mathrm{E}\left\{d \mid x(n), x(n-1), \ldots\right\},$$

Table 12.1 Counter-Example: (12.103) does not Imply Independence

$r \setminus s$	−1	0	1
−1	0	1/2	0
1	1/4	0	1/4

it holds that $v_o(n)$ has zero mean and is uncorrelated with any function of $x(n), x(n-1), \dots$:

$$\mathrm{E}\{v_o(n)\} = 0, \quad \mathrm{E}\{v_o(n) f(x(n), x(n-1), \dots\} = 0. \tag{12.105}$$

Note that, in general, the optimal solution of (12.104) depends on n, so we should write

$$\hat{\mathcal{H}}_{o,n}(x(n), x(n-1), \dots) = \mathrm{E}\{d | x(n), x(n-1), \dots\}, \tag{12.106}$$

to make the time dependence explicit.

We can now return to the question at the start of this section: From our results thus far, we see that minimizing $\mathrm{E}\{e^2(n)\}$ leads to a model which relates $d(n)$ and the sequence $\{x(n)\}$ such that

$$d(n) = \hat{\mathcal{H}}_{o,n}(x(n), x(n-1), \dots) + v_o(n),$$

in which $v_o(n)$ is uncorrelated with any function $f(\cdot)$ of $x(n), x(n-1), \dots$ In other words, the solution of (12.104) imposes such a model on our data, even if this model makes no physical sense at all. This is an important point: such a model will always result from the minimization of the mean-square error, at least as long as the statistics of $d(n)$ and $\{x(n)\}$ are such that the expected values in (12.104) are finite.

Now, what if we knew beforehand, by physical arguments about the data, that a model such as (12.40) holds between $d(n)$ and $\{x(n)\}$, such that $v(n)$ has zero mean and is uncorrelated to any function of $x(n), x(n-1), \dots$? Will it result that the solution of (12.104) will be such that

$$\hat{\mathcal{H}}_o(x(n), x(n-1), \dots) = \mathcal{H}(x(n), x(n-1), \dots), \quad v_o(n) = v(n)?$$

To answer this question, let us go back to (12.99), and use (12.40) to find the solution. We thus have (we omit the arguments of $d(n)$, $v(n)$ and $\mathcal{H}(\cdot)$ for simplicity)

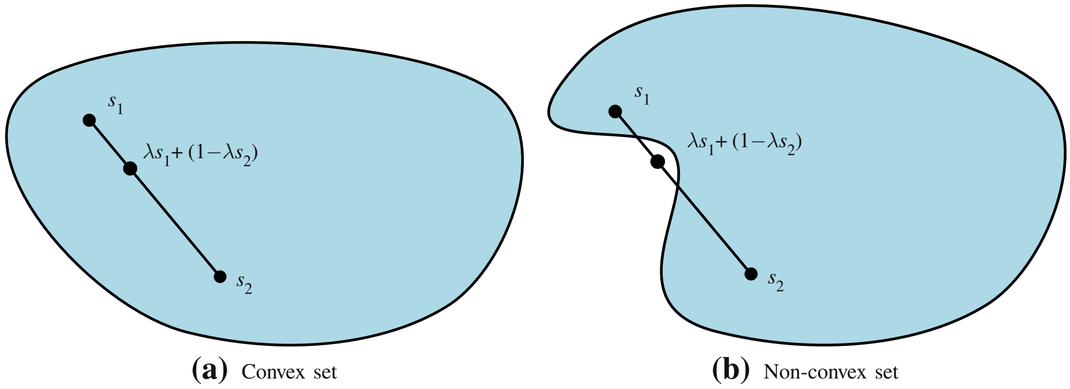
$$\min_{\hat{\mathcal{H}}} \mathrm{E}\{[d - \hat{\mathcal{H}}(x)]^2\} = \min_{\hat{\mathcal{H}}} \mathrm{E}\left\{[\mathcal{H}_n + v - \hat{\mathcal{H}}]^2\right\}.$$

Since v is orthogonal to any function of $x(n), x(n-1), \dots$ by assumption, it will be orthogonal to $\mathcal{H}_n$ and $\hat{\mathcal{H}}$ in particular. Therefore, we can simplify the cost as

$$\min_{\hat{\mathcal{H}}} \mathrm{E}\{[d - \hat{\mathcal{H}}(x)]^2\} = \min_{\hat{\mathcal{H}}} \left[\mathrm{E}\left\{(\mathcal{H} - \hat{\mathcal{H}})^2\right\} + \mathrm{E}\{v^2\}\right]. \tag{12.107}$$

Both terms in (12.107) are positive, and only the first depends on $\hat{\mathcal{H}}$. Therefore, the solution is indeed $\hat{\mathcal{H}} = \mathcal{H}$ and $v_o = v$, as one would wish (to be precise, $\mathcal{H}$ and $\hat{\mathcal{H}}$ could differ on a set of probability zero).

We conclude that we can use (12.102) as a criterion for separating the two parts $y(n)$ and $v(n)$ of our model for $d(n)$ if and only if $v(n)$ satisfies (12.105). This holds, for example, if $v(n)$ has zero mean and is independent of $x(n-k)$ for all $k \geq 0$.



s_1
$\lambda s_1 + (1 - \lambda s_2)$
s_2
(a) Convex set
s_1
$\lambda s_1 + (1 - \lambda s_2)$
s_2
(b) Non-convex set

FIGURE 12.22

Convexity.

Note that since $\hat{y}(n) = \hat{\mathcal{H}}_{o,n}(x(n), x(n-1), \ldots)$ is itself a function of $x(n), x(n-1), \ldots$, (12.105) implies that

$$\mathrm{E}\{v_o(n)\hat{y}(n)\} = 0,$$

and we have

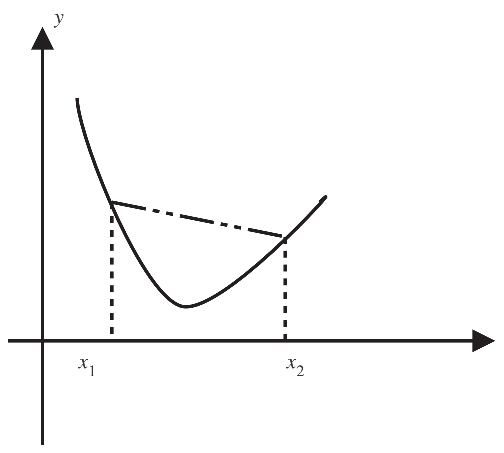
$$\begin{aligned}
\mathrm{E}\{v_o^2(n)\} &= \mathrm{E}\{v_o[d(n) - \hat{y}(n)]\} = \mathrm{E}\{v_o d(n)\} \\
&= \mathrm{E}\{[d(n) - \hat{y}(n)]d(n)\} \\
&= \mathrm{E}\{d^2(n)\} - \mathrm{E}\{d(n)\hat{y}(n)\} \\
&= \mathrm{E}\{d^2(n)\} - \mathrm{E}\{[\hat{y}(n) + v_o(n)]\hat{y}(n)\} \\
&= \mathrm{E}\{d^2(n)\} - \mathrm{E}\{\hat{y}^2(n)\}.
\end{aligned}$$

Recalling that $\mathrm{E}\{\hat{y}(n)\} = \mathrm{E}\{d(n)\}$, if we add and subtract $\mathrm{E}^2\{d(n)\}$ to the last result, we conclude that

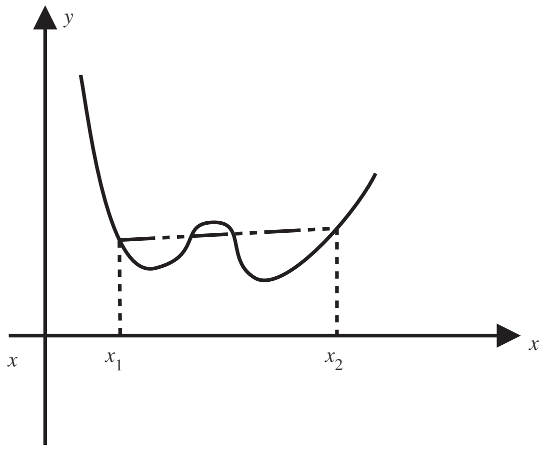
$$\mathrm{E}\{v_o^2(n)\} = \sigma_d^2 - \sigma_{\hat{y}}^2 \le \sigma_d^2, \tag{12.108}$$

where σ_d^2 and $\sigma_{\hat{y}}^2$ are the variances of $d(n)$ and of $\hat{y}(n)$.

The solution we just found is very general (it assumes little about $\mathcal{H}$), and there are ways of computing it adaptively from the data, using for example algorithms known as particle filters [32]. The solution requires, however, that one builds approximations for the joint probability distribution of $d(n)$ and $x(n), x(n-1), \ldots$, which in general requires a large amount of data, resulting in relatively high complexity and relatively slow convergence. If you know more about the relation between $\{x(n)\}$ and $d(n)$, this information might be used to constrain the search to a smaller class of problems, potentially leading to simpler algorithms with faster convergence. That is the goal of adaptive filters, and the reason why we restricted ourselves to this case in the main text.



(a) Convex function



(b) Non-convex function

FIGURE 12.23

Convex functions.

Box 5: Convex functions

Convex sets are such that lines between any two points in the set always stay entirely in the set (see Figure 12.22). More formally, a set $\mathcal{S} \in \mathcal{R}^n$ is convex if

$$s_1, s_2 \in \mathcal{S} \Rightarrow \lambda s_1 + (1-\lambda)s_2 \in \mathcal{S}, \quad \text{for all } \lambda \in [0,1].$$

Convex *functions*, on the other hand, are functions $f(\boldsymbol{x}) : \mathcal{R}^n \to \mathcal{R}$ such that for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{R}^n$ and $0 < \lambda < 1$,

$$f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) \leq \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2).$$

The function is called *strictly convex* if the inequality is strict ($<$ instead of $\leq$). For example, $f(x) = x^2$ is strictly convex. Figure 12.23 shows examples of convex and non-convex functions. An important property of strictly convex functions is that they always have a unique minimum, that is, they never have sub-optimal local minima.

Box 6: Wirtinger derivatives and minima of functions of a complex variable

Suppose you want to minimize a function

$$\begin{aligned} f &: \mathcal{C} \to \mathcal{R}, \\ w \in \mathcal{C} &\mapsto f(w) \in \mathcal{R}, \end{aligned} \qquad (12.109)$$

that is, a function that takes a complex argument to a real value. The minimization problem is well defined, since the image of f is real. However, we cannot use standard arguments about derivatives and stationary points to find the minimum, because a function like this is never analytic, that is, the standard derivative used in complex analysis does not exist.

To see why, we need to think about f as a function of two variables, x and y, the real and imaginary parts of w. Take $f(w) = |w|^2 = ww^*$ for example. We can write it as a function of x and y as

$$\hat{f}(x, y) = f(x + jy) = x^2 + y^2. \tag{12.110}$$

Since it is a function of two variables, the derivatives in principle depend on the direction we are moving along. Consider the derivative at a point w along the x axis, that is, keeping y constant:

$$\lim_{\epsilon\to 0} \frac{(x+\epsilon)^2 + y^2 - (x^2+y^2)}{\epsilon} = 2x.$$

On the other hand, along the y axis, we have

$$\lim_{\epsilon\to 0} \frac{x^2 + (y+\epsilon)^2 - (x^2+y^2)}{\epsilon} = 2y,$$

that is, the derivative at point w is different according to the direction we are moving. This is essentially what is meant by saying that the function is not analytic at w.

For contrast, consider now the analytic function $g = w^2$. Its derivative is the same, no matter which direction we take:

$$\begin{aligned}&\lim_{\epsilon\to 0} \frac{(x+\epsilon+jy)^2 - (x+jy)^2}{\epsilon}\\ &\quad = \lim_{\epsilon\to 0} \frac{(x+\epsilon)^2 - y^2 + 2j(xy+\epsilon y) - (x^2 - y^2 + 2jxy)}{\epsilon} = 2w.\end{aligned}$$

Similarly, differentiating along the y axis we obtain again the same result

$$\lim_{\epsilon\to 0} \frac{\big(x + j(y+\epsilon)\big)^2 - (x+jy)^2}{\epsilon} = 2w.$$

When the derivative at a certain point w_0 is the same along every direction, we can define derivatives of complex functions writing simply

$$\frac{\mathrm{d}g(w_0)}{\mathrm{d}w} = \lim_{\Delta w\to 0} \frac{g(w_0+\Delta w) - g(w_0)}{\Delta w}, \tag{12.111}$$

since we know the limit does not depend on the direction along which Δw goes to zero.

It can be shown that a function is analytic at a given point only if the *Cauchy-Riemann conditions* hold at that point. Separating a generic complex function g into its real and imaginary parts $g(w) = u(x, y) + jv(x, y)$, the Cauchy-Riemann conditions are [33]

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}.$$

In the case of a *real* function of a complex variable, such as f defined as in (12.109), $v \equiv 0$, and of course the conditions could hold only for points in which the real part has zero partial derivatives.

This means that we cannot search for the stationary points of a function such as (12.109) using the derivative defined as in (12.111). We could of course always go back and interpret the function as a function of two variables, as we did in (12.110). In order to find the minimum of $f(w)$, we would need to find the stationary points of $\hat{f}(x, y)$

$$\frac{\partial \hat{f}}{\partial x} = 0, \quad \frac{\partial \hat{f}}{\partial y} = 0.$$

This works just fine, of course, but there is a nice trick, which allows us to keep working with complex variables in a simple way. The idea is to define derivatives differently, such that optimization problems become easy to solve, using what is known as *Wirtinger calculus*.

Real functions of complex variables are usually defined in terms of variables and its conjugates, since $w + w^*$ and ww^* are both real. The idea is to define derivatives such that w and w^* can be treated as if they were independent variables, as follows. Despite the motivation, the new derivatives are defined for general functions $g : \mathcal{C} \to \mathcal{C}$.

Define partial derivatives of a function $g : \mathcal{C} \to \mathcal{C}$ with respect to w and w^* as

$$\frac{\partial g}{\partial w} = \frac{1}{2}\left\{\frac{\partial g}{\partial x} - j\frac{\partial g}{\partial y}\right\}, \quad \frac{\partial g}{\partial w^*} = \frac{1}{2}\left\{\frac{\partial g}{\partial x} + j\frac{\partial g}{\partial y}\right\}. \tag{12.112}$$

You can check that if g is analytic at a given point (so g satisfies the Cauchy-Riemann conditions at that point), then $\partial g/\partial w^* = 0$ at that point.

Consider, for example, $g(w) = w^m (w^*)^n = (x + jy)^m (x - jy)^n$. Then,

$$\begin{aligned}\frac{\partial g}{\partial w} &= \frac{1}{2}\Big\{m(x + jy)^{m-1}(x - jy)^n + n(x + jy)^m (x - jy)^{n-1} \\ &\quad - j(jm(x + jy)^{m-1}(x - jy)^n - jn(x + jy)^m (x - jy)^{n-1})\Big\} \\ &= \frac{1}{2}\Big\{mw^{m-1}(w^*)^n + nw^m (w^*)^{n-1} + mw^{m-1}(w^*)^n - nw^m (w^*)^{n-1}\Big\} \\ &= mw^{m-1}(w^*)^n.\end{aligned}$$

As advertised, the final result is what we would obtain if we had treated w and w^* as two independent variables. Similarly, you can check that

$$\frac{\partial g}{\partial w^*} = nw^m (w^*)^{n-1}.$$

An important special case is the quadratic function $f(w) = d - w^* r - r^* w + w^* w R$, which assumes only real values if d and R are real. Using the previous result, we see that

$$\frac{\partial f}{\partial w} = -r^* + w^* R, \quad \frac{\partial f}{\partial w^*} = -r + wR.$$

Note that there is no factor of 2 in this case. This difference between Wirtinger derivatives and standard real derivatives will propagate to many expressions in this text.

Now, the definitions (12.112) are useful for minimizing real functions of complex variables as in (12.109), since from (12.112)

$$\frac{\partial f}{\partial w^*} = 0 \Leftrightarrow \frac{\partial \hat{f}}{\partial x} = 0 \quad \text{and} \quad \frac{\partial \hat{f}}{\partial y} = 0.$$

Applying this idea to our quadratic function, we see that

$$\frac{\partial f}{\partial w^*} = 0 \Rightarrow R w_o = r \Rightarrow w_o = r/R,$$

if $R \neq 0$. In addition,

$$\frac{\partial^2 f}{\partial w\, \partial w^*} = R,$$

and we can conclude that w_o is a minimum as long as $R > 0$.

The same conclusions could easily be obtained working directly with $\hat{f}$. The Wirtinger derivatives are merely a shortcut, that makes working with complex variables very similar to working with real variables. A good detailed description of Wirtinger derivatives can be found in [34].

In the vector case, the definitions and results are very similar. Given a function

$$g : \mathcal{C}^M \to \mathcal{R},$$
$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix} \in \mathcal{C}^M \mapsto g(\boldsymbol{w}),$$

with $w_i = x_i + j y_i$, $i = 0, \ldots, M-1$, we define

$$\frac{\partial g}{\partial \boldsymbol{w}} = \begin{bmatrix} \frac{\partial g}{\partial w_0} & \cdots & \frac{\partial g}{\partial w_{M-1}} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \frac{\partial g}{\partial x_0} - j \frac{\partial g}{\partial y_0} & \cdots & \frac{\partial g}{\partial x_{M-1}} - j \frac{\partial g}{\partial y_{M-1}} \end{bmatrix},$$
$$\frac{\partial g}{\partial \boldsymbol{w}^H} = \begin{bmatrix} \frac{\partial g}{\partial w_0^*} \\ \vdots \\ \frac{\partial g}{\partial w_{M-1}^*} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \frac{\partial g}{\partial x_0} - j \frac{\partial g}{\partial y_0} \\ \vdots \\ \frac{\partial g}{\partial x_{M-1}} - j \frac{\partial g}{\partial y_{M-1}} \end{bmatrix}.$$

Using our definition of gradients, the second derivative of f (its *Hessian*) is

$$\frac{\partial^2 f}{\partial \boldsymbol{w}\, \partial \boldsymbol{w}^H} = \begin{bmatrix} \frac{\partial}{\partial \boldsymbol{w}} \left\{ \frac{\partial f}{\partial w_0^*} \right\} \\ \vdots \\ \frac{\partial}{\partial \boldsymbol{w}} \left\{ \frac{\partial f}{\partial w_{M-1}^*} \right\} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 f}{\partial w_0\, \partial w_0^*} & \cdots & \frac{\partial^2 f}{\partial w_{M-1}\, \partial w_0^*} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_0\, \partial w_{M-1}^*} & \cdots & \frac{\partial^2 f}{\partial w_{M-1}\, \partial w_{M-1}^*} \end{bmatrix}.$$

The most important example is the quadratic function

$$f(\boldsymbol{w}) = r_d - \boldsymbol{w}^H \boldsymbol{r}_{d\boldsymbol{\phi}} - \boldsymbol{r}_{d\boldsymbol{\phi}}^H \boldsymbol{w} + \boldsymbol{w}^H \boldsymbol{R}_{\boldsymbol{\phi}} \boldsymbol{w},$$

where r_d is a real variable, and $\boldsymbol{R}_{\boldsymbol{\phi}}$ is Hermitian symmetric. Then, we have

$$\frac{\partial f}{\partial w^H} = -\boldsymbol{r}_{d\boldsymbol{\phi}} + \boldsymbol{R}_{\boldsymbol{\phi}} \boldsymbol{w}.$$

We see that a stationary point of f is such that $\boldsymbol{R}_{\boldsymbol{\phi}} \boldsymbol{w}_{\text{o}} = \boldsymbol{r}_{d\boldsymbol{\phi}}$. In addition,

$$\frac{\partial^2 f}{\partial \boldsymbol{w}\, \partial \boldsymbol{w}^H} = \boldsymbol{R}_{\boldsymbol{\phi}},$$

and we conclude that $\boldsymbol{w}_{\text{o}}$ is the single point of minimum of f if and only if $\boldsymbol{R}_{\boldsymbol{\phi}}$ is positive-definite.

Note that in the complex case, the gradient and the Hessian do not have the factors of 2 that we saw in the real case (see Box 1).

1.12.3 Stochastic algorithms

In this section, we focus on three of the most important adaptive algorithms: the least-mean squares (LMS) algorithm, the normalized least-mean squares (NLMS) algorithm, and the recursive least-squares (RLS) algorithm, which are the basis of many other algorithms. These three stochastic algorithms are derived in detail and analyzed using different methods. We present different approaches for analyzing an adaptive filter, because the variety and complexity of their behavior makes it difficult for a single technique to be able to handle all situations. The analyses provided here allow us to predict the behavior of the algorithms in transient and steady-state, and choose values of parameters (such as the step-size) for which the filters operate adequately. In particular, we study conditions under which the filters will remain stable.

We assume linearly parametrized classes, such as FIR ($\mathcal{F}_{\text{FIR}}$) and Volterra ($\mathcal{F}_{\text{V}}$) filters (see Section 1.12.2.1). In these classes, each element of the input regressor vector is given by a certain function ϕ_i $i = 0, 1, \ldots, M-1$ of $x(n), x(n-1), \ldots, x(n-N)$. In the case of length-$M$ FIR filters, we have

$$\begin{aligned} \boldsymbol{\phi}(n) &= \left[\phi_0(n)\ \phi_1(n)\ \cdots\ \phi_{M-1}(n)\right]^T \\ &= \left[x(n)\ x(n-1)\ \cdots\ x(n-M+1)\right]^T, \end{aligned} \tag{12.113}$$

whereas in the case of second-order Volterra filters with memory $N = 1$ and real-valued signals, we have

$$\begin{aligned} \boldsymbol{\phi}(n) &= \left[\phi_0(n)\ \phi_1(n)\ \phi_2(n)\ \phi_3(n)\ \phi_4(n)\right]^T \\ &= \left[x(n)\ x(n-1)\ x^2(n)\ x(n)x(n-1)\ x^2(n-1)\right]^T. \end{aligned} \tag{12.114}$$

As much as possible, our presentation in this section is independent of the choices of $\mathcal{F}$ and $\phi(\cdot)$. However, some algorithms are designed for a specific class $\mathcal{F}$ (the most common case are fast algorithms designed for FIR filters (12.113)). In these cases, we will alert the reader of the restriction.

Using the results of Box 6, the derivation of algorithms for real or circular complex inputs is very similar, the only differences being the need of conjugating a few variables in some expressions, and the appearance of a factor $\beta = 2$ in the expressions of gradients of functions of real variables, and $\beta = 1$ in the complex case.

In general, the output of a length-M adaptive filter is given by

$$\hat{y}(n) = \boldsymbol{w}^H(n)\boldsymbol{\phi}(n) = w_0^*(n)\phi_0(n) + w_1^*(n)\phi_1(n) + \cdots + w_{M-1}^*(n)\phi_{M-1}(n). \tag{12.115}$$

In the case of Volterra filters, it is common to use the notation $w_{k,\ell}$ for the weight related to the input $x(n-k)x(n-\ell)$ (see Eq. (12.43)). However, to obtain a common formulation independent of the class $\mathcal{F}$, we denote the weight vector as

$$\boldsymbol{w}(n) = \begin{bmatrix} w_0(n) & w_1(n) & w_2(n) & \cdots & w_{M-1}(n) \end{bmatrix}^T. \tag{12.116}$$

The task of the adaptive filter will be to choose in a recursive form the values of the parameters $w_0, w_1, \ldots, w_{M-1}$ that best fit the data at each time instant n.

Figure 12.24 shows the scheme of a length-M adaptive filter for linearly parametrized classes, where Φ denotes the set of functions $\phi_i(\cdot)$, $i = 0, 1, \ldots, M-1$ of $x(n), x(n-1), \ldots, x(n-N)$ that generate the elements $\phi_0(n), \phi_1(n), \ldots, \phi_{M-1}(n)$ of the input regressor vector $\boldsymbol{\phi}(n)$. The number of samples of the signal $x(n)$ depends on the class $\mathcal{F}$ and is given by $N+1$. An FIR filter with length M requires $N+1 = M$ samples. On the other hand, a Volterra filter with memory N, requires $N+1$ samples, but the length of the corresponding adaptive filter depends on the order of the Volterra series expansion. At each time instant, the output of the adaptive filter $\hat{y}(n)$ is compared with the desired signal $d(n)$ to compute the error $e(n) = d(n) - \hat{y}(n)$. In order to minimize a cost function of this error, a stochastic algorithm uses $e(n)$ to adjust the filter weights.

1.12.3.1 LMS algorithm

The update equation of the steepest-descent algorithm is given by

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \frac{\mu}{\beta}\nabla_{\boldsymbol{w}} J(n), \tag{12.117}$$

where μ is a step-size and $\beta = 1$ (resp., $\beta = 2$) for complex (resp., real) data (see Box 6). This algorithm requires exact measurements of the gradient vector. For the case of minimization of the mean-square error, that is, if $J(n) = \mathrm{E}\{|e(n)|^2\}$,

$$\nabla_{\boldsymbol{w}} J(n) = -\beta \boldsymbol{r}_{d\boldsymbol{\phi}} + \beta \boldsymbol{R}_{\boldsymbol{\phi}} \boldsymbol{w}(n) \tag{12.118}$$

at each iteration n (see Box 1 for an introduction to gradient algorithms). Note that the exact gradient requires a prior knowledge of the correlation matrix $\boldsymbol{R}_{\boldsymbol{\phi}}$ and the cross-correlation vector $\boldsymbol{r}_{d\boldsymbol{\phi}}$, which is not feasible in real-time applications. For example, in the hands-free telephone application described in Section 1.12.1.1, the input signal of the adaptive filter is the far-end speech (see Figures 12.3 and 12.4). A speech signal is nonstationary and therefore, it is not possible to obtain exact estimates for $\boldsymbol{R}_{\boldsymbol{\phi}}$ and $\boldsymbol{r}_{d\boldsymbol{\phi}}$ at each time instant. Furthermore, good estimates would demand much processing time, which

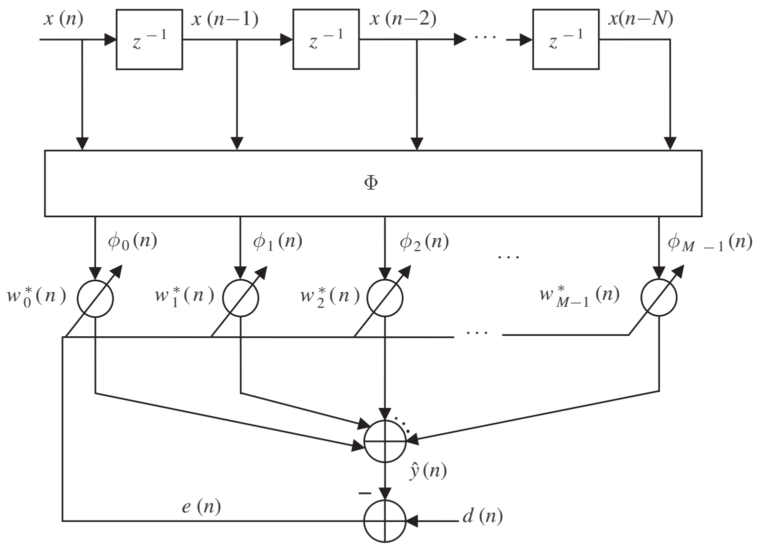


FIGURE 12.24

Scheme for a length-M adaptive filter assuming linearly parametrized classes.

could insert an inadmissible delay in the system, making this approach not feasible, except in off-line applications. Similar restrictions appear in virtually all adaptive filtering applications.

To circumvent this problem, a stochastic version of the steepest descent algorithm was proposed, the least-mean squares (LMS) algorithm. Instead of minimizing the exact mean-square error, the LMS algorithm minimizes a straightforward approximation to it: $|e(n)|^2$. The idea, as we saw in Section 1.12.1.2.4, is that the algorithm will approximately average the cost-function if adaptation is slow (small step-size). The estimation error is

$$e(n) = d(n) - \boldsymbol{w}^H(n)\boldsymbol{\phi}(n), \tag{12.119}$$

so the gradient of the instantaneous cost function is

$$\widehat{\nabla}_{\boldsymbol{w}} J(n) = -\beta\boldsymbol{\phi}(n)d^*(n) + \beta\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{w}(n) = -\beta\boldsymbol{\phi}(n)e^*(n). \tag{12.120}$$

Since $e(n)$ depends on the old estimate of the weight vector, it is called an *a priori* estimation error, by opposition to the *a posteriori* error that will be introduced in (12.129). Instead of minimizing the instantaneous cost function, the stochastic gradient defined in (12.120) can be obtained by replacing the instantaneous estimates

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n) = \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n) \tag{12.121}$$

and

$$\widehat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n) = \boldsymbol{\phi}(n)d^*(n) \tag{12.122}$$

in (12.118).

Table 12.2 Summary of the LMS Algorithm

Initialization:
Set $\mathbf{w}(0) = \mathbf{0}$
For $n = 0, 1, 2, \ldots,$ compute
$\hat{y}(n) = \mathbf{w}^H(n)\boldsymbol{\phi}(n)$
$e(n) = d(n) - \hat{y}(n)$
$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e^*(n)\boldsymbol{\phi}(n)$

Table 12.3 Computational Cost of the LMS Algorithm for Complex and Real-Valued Signals in Terms of Real Multiplications and Real Additions Per Iteration

Term	Real Signals		Complex Signals	
	×	+	×	+
$\hat{y}^*(n) = \boldsymbol{\phi}^H(n)[\mathbf{w}(n)]$	M	$M-1$	$4M$	$4M-2$
$e^*(n) = d^*(n) - [\hat{y}^*(n)]$		1		2
$\mu[e^*(n)]$	1		2	
$[\mu e^*(n)]\boldsymbol{\phi}(n)$	M		$4M$	$2M$
$\mathbf{w}(n+1) = [\mathbf{w}(n)] + [\mu e^*(n)\boldsymbol{\phi}(n)]$		M		$2M$
TOTAL per iteration	$2M+1$	$2M$	$8M+2$	$8M$

Replacing (12.120) in (12.117), we arrive at the update equation for LMS, i.e.,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\boldsymbol{\phi}(n)e^*(n). \tag{12.123}$$

As explained in Section 1.12.1.2.4 and illustrated in the animations of Figure 12.9, although there is no explicit average being taken in the LMS algorithm, it in fact computes an implicit approximate average if the step-size is small.

The LMS algorithm is summarized in Table 12.2. Its computational cost increases linearly with the length of the filter, i.e., the computational cost is $\mathcal{O}(M)$. Table 12.3 shows the number of real multiplications and real additions per iteration for real and complex-valued signals. The quantities inside brackets were computed in previous steps. We assume that a complex multiplication requires 4 real multiplications and 2 real additions (it is possible to use less multiplications and more additions), and a complex addition requires 2 real additions. Different ways of carrying out the calculations may result in a different computational cost. For example, if we first computed $\mu \times \boldsymbol{\phi}(n)$ followed by $\left[\mu\boldsymbol{\phi}(n)\right] \times e^*(n)$, LMS would require $M-1$ more real multiplications per iteration than the ordering of Table 12.3, considering real-valued signals. For complex-valued signals, the cost would be even higher, since LMS would require $4M-2$ and $2M$ more real multiplications and real additions, respectively.

1.12.3.1.1 A deterministic approach for the stability of LMS

Finding the range of step-sizes such that the recursion (12.123) converges is a complicated task. There are two main approaches to finding the maximum step-size: an average approach and a deterministic (worst-case) approach. The average approach will be treated in Section 1.12.4.3. Since the deterministic approach is easy to explain intuitively, we address it in the sequel (although a precise proof is not so simple, see [35,36] for a full analysis).

For simplicity, assume that our filter is such that $d(n) \equiv 0$. Although this is not a very likely situation to encounter in practice, it allows us to avoid technical details that would complicate considerably the discussion, and still arrive at the important conclusions.

Let us rewrite the LMS recursion, expanding the error with $d(n) = 0$ as $e^*(n) = -(\boldsymbol{w}^H(n)\boldsymbol{\phi}(n))^* = -\boldsymbol{\phi}^H(n)\boldsymbol{w}(n)$ to obtain

$$\boldsymbol{w}(n+1) = \left[\boldsymbol{I} - \mu\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right]\boldsymbol{w}(n). \tag{12.124}$$

This is a linear system, but not a time-invariant one. It will be stable if the eigenvalues $\lambda_i(n)$ of the matrix $\boldsymbol{A}(n) = \boldsymbol{I} - \mu\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)$ satisfy

$$\max_i |\lambda_i(n)| \le 1, \quad \text{for all } n \ge 0. \tag{12.125}$$

Be careful, if $d(n)$ were not identically zero, condition (12.125) would have to be slightly modified to guarantee stability (see, e.g., [36,37]).

The eigenvalues of $\boldsymbol{A}(n)$ are easy to find, if we note that $\boldsymbol{\phi}(n)$ is an eigenvector of $\boldsymbol{A}(n)$ (the case $\boldsymbol{\phi}(n) = \boldsymbol{0}$ is trivial):

$$\boldsymbol{A}(n)\boldsymbol{\phi}(n) = \boldsymbol{\phi}(n) - \mu\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n) = (1 - \mu\|\boldsymbol{\phi}(n)\|^2)\boldsymbol{\phi}(n),$$

where $\|\boldsymbol{\phi}(n)\|^2 = \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)$ is the square of the Euclidean norm. The corresponding eigenvalue is $\lambda_1 = 1 - \mu\|\boldsymbol{\phi}(n)\|^2$. All other eigenvalues are equal to one (the eigenvectors are all vectors orthogonal to $\boldsymbol{\phi}(n)$.) Therefore, the LMS algorithm (12.124) will be stable whenever

$$-1 \le 1 - \mu\|\boldsymbol{\phi}(n)\|^2 \le 1 \iff 0 \le \mu \le \frac{2}{\sup_{n\ge 0}\|\boldsymbol{\phi}(n)\|^2}. \tag{12.126}$$

The *supremum* (sup) of a sequence is an extension to the maximum, defined to avoid a technicality. Consider the sequence $f(n) = \{1 - 0.5^n\}$, $n \ge 0$. Strictly speaking, it has no maximum, since there is no n_0 for which $f(n_0) \ge f(n)$ for all $n \ge 0$. The supremum is defined so we can avoid this problem: $\sup_{n\ge 0} f(n) = 1$. Therefore, the supremum is equal to the maximum whenever the later exists, but is also defined for some sequences that do not have a maximum. The *infimum* (inf) has a similar relation to the minimum.

This condition is sufficient for stability, but is too conservative, since $\|\boldsymbol{\phi}(n)\|^2$ is not at its highest value at all instants. A popular solution is to use *normalization*. The idea is simple: adopt a time-variant step-size $\mu(n)$ such that

$$\mu(n) = \frac{\tilde{\mu}}{\varepsilon + \|\boldsymbol{\phi}(n)\|^2}, \tag{12.127}$$

with $0 < \tilde{\mu} < 2$ and where $\varepsilon > 0$ is used to avoid division by zero. The resulting algorithm is known as *normalized* LMS (NLMS). We will talk more about it in the next section.

1.12.3.2 Normalized LMS algorithm

One problem with the LMS algorithm is how to choose the step-size. How large should it be to enable a high convergence rate, provide an acceptable misadjustment, and even ensure the stability of LMS? More importantly, how can this choice be made so that the filter will work correctly even for very different input signals? One answer to this problem is the normalized LMS algorithm which we describe now. It uses a time varying step-size, which is particularly useful when the statistics of the input signals change quickly.

Thus, replacing the fixed step-size μ by a time varying step-size $\mu(n)$ in (12.123), we obtain

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu(n)\boldsymbol{\phi}(n)e^*(n). \tag{12.128}$$

In order to increase the convergence speed, we could try to find $\mu(n)$ such that the *a posteriori* estimation error,

$$\xi(n) = d(n) - \boldsymbol{w}^H(n+1)\boldsymbol{\phi}(n), \tag{12.129}$$

is zeroed. Note that, in contrast to the *a priori* estimation error defined in (12.119), $\boldsymbol{\xi}(n)$ depends on the updated estimate of the weight vector, hence the name *a posteriori* estimation error. Replacing (12.128) in (12.129), we obtain

$$\begin{aligned}\xi(n) &= \underbrace{d(n) - \boldsymbol{w}^H(n)\boldsymbol{\phi}(n)}_{e(n)} - \mu(n)\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)e(n)\\ &= [1-\mu(n)\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)]e(n).\end{aligned} \tag{12.130}$$

In order to enforce $\xi(n) = 0$ at each time instant n, we must select

$$\mu(n) = \frac{1}{\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)} = \frac{1}{\|\boldsymbol{\phi}(n)\|^2}. \tag{12.131}$$

Replacing (12.131) in (12.128), we obtain the update equation of the normalized least-squares algorithm, i.e.,

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{1}{\|\boldsymbol{\phi}(n)\|^2}\boldsymbol{\phi}(n)e^*(n). \tag{12.132}$$

Note that the time varying step-size $\mu(n)$ depends inversely on the instantaneous power of the input vector $\boldsymbol{\phi}(n)$. Indeed, we will show in Section 1.12.4.3 that the maximum value of the fixed step-size μ to ensure the convergence and mean-square stability of LMS depends inversely on the power of the input signal.

In order to make (12.132) more reliable for practical implementations, it is common to introduce two positive constants: a fixed step-size $\tilde{\mu}$ to control the rate of convergence (and the misadjustment) and the regularization factor ε to prevent division by a small value when $\|\boldsymbol{\phi}(n)\|^2$ is small. With these constants, the NLMS update equation reduces to

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{\tilde{\mu}}{\varepsilon + \|\boldsymbol{\phi}(n)\|^2}\boldsymbol{\phi}(n)e^*(n). \tag{12.133}$$

Some authors refer to (12.132) as NLMS and to (12.133) as ε-NLMS. However, for simplicity we will refer to both as NLMS. The name "normalized" is due to a most useful property of NLMS: the range of values of $\tilde{\mu}$ for which the algorithm remains stable is independent of the statistics of $x(n)$, i.e., $0 < \tilde{\mu} < 2$, as can be observed in Section 1.12.3.1.1.

As pointed out in [3], the NLMS algorithm can also be interpreted as a quasi-Newton algorithm. As briefly explained in Box 1, a quasi-Newton algorithm updates the weights using approximations for the Hessian matrix and gradient vector, i.e.,

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \frac{\mu}{\beta}\left[\widehat{\nabla}^2_{\boldsymbol{w}} J(n)\right]^{-1} \widehat{\nabla}_{\boldsymbol{w}} J(n). \tag{12.134}$$

Assuming the following instantaneous approximations

$$\widehat{\nabla}_{\boldsymbol{w}} J(n) = -\beta\boldsymbol{\phi}(n)d^*(n) + \beta\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{w}(n) = -\beta\boldsymbol{\phi}(n)e^*(n) \tag{12.135}$$

and

$$\widehat{\nabla}^2_{\boldsymbol{w}} J(n) = \varepsilon\boldsymbol{I} + \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n), \tag{12.136}$$

and replacing them in (12.134), we arrive at the stochastic recursion

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\left[\varepsilon\boldsymbol{I} + \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right]^{-1} \boldsymbol{\phi}(n)e^*(n). \tag{12.137}$$

The term $\varepsilon\boldsymbol{I}$ guarantees that (12.136) is invertible (a rank-one matrix such as $\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)$ is never invertible).

To compute $[\varepsilon\boldsymbol{I} + \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)]^{-1}$, we can use the *matrix inversion lemma* (Box 3), which gives us an expression for the inverse of an $M \times M$ matrix of the form

$$\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{C}\boldsymbol{D}\boldsymbol{C}^H, \tag{12.138}$$

where $\boldsymbol{B}$ and $\boldsymbol{D}$ are two square matrices with dimensions M and N, respectively, and $\boldsymbol{C}$ is an $M \times N$ matrix. According to the matrix inversion lemma, the inverse of $\boldsymbol{A}$ is given by

$$\boldsymbol{A}^{-1} = \boldsymbol{B}^{-1} - \boldsymbol{B}^{-1}\boldsymbol{C}(\boldsymbol{D}^{-1} + \boldsymbol{C}^H\boldsymbol{B}^{-1}\boldsymbol{C})^{-1}\boldsymbol{C}^H\boldsymbol{B}^{-1}. \tag{12.139}$$

Thus, identifying

$$\boldsymbol{A} = \varepsilon\boldsymbol{I} + \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n), \quad \boldsymbol{B} = \varepsilon\boldsymbol{I}, \quad \boldsymbol{C} = \boldsymbol{\phi}(n), \quad \text{and} \quad \boldsymbol{D} = 1,$$

we obtain

$$[\varepsilon\boldsymbol{I} + \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)]^{-1} = \varepsilon^{-1}\boldsymbol{I} - \frac{\varepsilon^{-2}}{1 + \varepsilon^{-1}\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)}\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n). \tag{12.140}$$

Multiplying (12.140) from the right by $\boldsymbol{\phi}(n)$, after some algebraic manipulations, we get

$$[\varepsilon\boldsymbol{I} + \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)]^{-1}\boldsymbol{\phi}(n) = \frac{\boldsymbol{\phi}(n)}{\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)} = \frac{\boldsymbol{\phi}(n)}{\varepsilon + \|\boldsymbol{\phi}(n)\|^2}. \tag{12.141}$$

Table 12.4 Summary of the NLMS Algorithm

Initialization:
Set $\boldsymbol{w}(0) = \mathbf{0}$ and $0 < \tilde{\mu} < 2$
For $n = 0, 1, 2, \ldots,$ compute
$\hat{y}(n) = \boldsymbol{w}^H(n)\boldsymbol{\phi}(n)$
$e(n) = d(n) - \hat{y}(n)$
$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \dfrac{\tilde{\mu}}{\varepsilon + \Vert\boldsymbol{\phi}(n)\Vert^2} e^*(n)\boldsymbol{\phi}(n)$

Replacing (12.141) in (12.137), we arrive at (12.133). Since Newton-based algorithms converge faster than gradient-based algorithms, it is expected that NLMS exhibits a higher convergence rate than that of LMS, which is indeed the case in general.

The NLMS algorithm is summarized in Table 12.4. To obtain its computational cost, we can use the same procedure considered in the computation of the number of operations of LMS (see Table 12.3). For complex-valued signals, NLMS requires $10M + 2$ real multiplications, $10M$ real additions and one real division per iteration. For real-valued signals, it requires $3M + 1$ real multiplications and $3M$ real additions per iteration. Note that for input vectors with a shift structure such as the FIR filters of Eq. (12.113), its computational cost can be reduced if $\Vert\boldsymbol{\phi}(n)\Vert^2$ is computed by the following recursion

$$\Vert\boldsymbol{\phi}(n)\Vert^2 = \Vert\boldsymbol{\phi}(n-1)\Vert^2 - |x(n-M)|^2 + |x(n)|^2 \tag{12.142}$$

with initialization $\Vert\boldsymbol{\phi}(-1)\Vert = 0$.

1.12.3.3 RLS algorithm

The convergence rate of LMS-type algorithms varies considerably with the input signal $x(n)$, since they are based on steepest-descent optimization algorithms. If $x(n)$ is a white noise, the convergence rate is high. On the other hand, if the correlation between successive samples of $x(n)$ is high, LMS-type algorithms converge slowly. The RLS algorithm does not have this drawback. However, this advantage has a price: a considerable increase in the computational cost. Furthermore, numerical errors play a more important role in RLS and may even lead to divergence of the weight estimates. Different versions of RLS were proposed to circumvent this problem. In the sequel, we obtain the equations of the conventional RLS algorithm.

The RLS can be derived in different ways (much as NLMS). Each form of arriving at the algorithm highlights a different set of its properties: one that leads to many new algorithms and insights is the connection between RLS and Kalman filters described in [38].

We present here shorter routes, starting with a derivation based on deterministic arguments. In fact, RLS solves a deterministic least-squares problem, which is based on the weighted least-squares cost function

$$J_{\text{LS}}(n) = \sum_{\ell=0}^{n} \lambda^{n-\ell}|\xi(\ell)|^2, \tag{12.143}$$

where $0 \ll \lambda < 1$ is a constant known as *forgetting factor*,

$$\xi(\ell) = d(\ell) - \boldsymbol{w}^H \boldsymbol{\phi}(\ell), \tag{12.144}$$

are *a posteriori* errors and

$$\boldsymbol{w}(n+1) = \arg\min_{\boldsymbol{w}} J_{\text{LS}}(n). \tag{12.145}$$

The factor $\lambda^{n-\ell}$ emphasizes the most recent errors, forgetting the errors from the remote past. This is important, otherwise the algorithm would give too much weight to the remote past and would not be able to track variations in the input signals. Note that $\boldsymbol{w}(n+1)$ minimizes $J_{\text{LS}}(n)$ at each time instant. Thus, RLS provides an exact solution for the least-squares problem at each time instant. As the NLMS algorithm, RLS seeks to minimize the *a posteriori* error $|\xi(n)|^2$, searching for an exact solution to an optimization problem. The difference is that RLS searches for a weight vector $\boldsymbol{w}(n+1)$ that takes into consideration the whole past history of inputs, minimizing $\lambda^n|\xi(0)|^2 + \lambda^{n-1}|\xi(1)|^2 + \cdots + \lambda|\xi(n-1)|^2 + |\xi(n)|^2$ at each time instant n, not only the current value $|\xi(n)|^2$ as in NLMS.

The error sequence $\xi(\ell)$ weighted by $\lambda^{(n-\ell)/2}$ in the interval $0 \le \ell \le n$ can be written in a vectorial form, i.e.,

$$\boldsymbol{\xi}(n) = \boldsymbol{d}(n) - \boldsymbol{\Phi}(n)\boldsymbol{w}^*, \tag{12.146}$$

where

$$\boldsymbol{\xi}(n) = \begin{bmatrix} \xi(n) & \lambda^{1/2}\xi(n-1) & \cdots & \lambda^{n/2}\xi(0) \end{bmatrix}^T, \tag{12.147}$$

$$\boldsymbol{d}(n) = \begin{bmatrix} d(n) & \lambda^{1/2}d(n-1) & \cdots & \lambda^{n/2}d(0) \end{bmatrix}^T \tag{12.148}$$

and

$$\boldsymbol{\Phi}(n) = \begin{bmatrix} \phi_0(n) & \phi_1(n) & \cdots & \phi_{M-1}(n) \\ \lambda^{1/2}\phi_0(n-1) & \lambda^{1/2}\phi_1(n-1) & \cdots & \lambda^{1/2}\phi_{M-1}(n-1) \\ \lambda\phi_0(n-2) & \lambda\phi_1(n-2) & \cdots & \lambda\phi_{M-1}(n-2) \\ \vdots & \ddots & \ddots & \vdots \\ \lambda^{n/2}\phi_0(0) & \lambda^{n/2}\phi_1(0) & \cdots & \lambda^{n/2}\phi_{M-1}(0) \end{bmatrix}. \tag{12.149}$$

Note that the weighted least-squares cost function can be rewritten in terms of the vector $\boldsymbol{\xi}(n)$, i.e.,

$$J_{\text{LS}}(n) = \boldsymbol{\xi}^H(n)\boldsymbol{\xi}(n) = \|\boldsymbol{\xi}(n)\|^2 = \sum_{\ell=0}^{n} \lambda^{n-\ell}|\xi(\ell)|^2. \tag{12.150}$$

Now, we have to find the weight vector $\boldsymbol{w}(n+1)$ that minimizes (12.150). The gradient of $J_{\text{LS}}(n)$ with relation to $\boldsymbol{w}^H$ is given by

$$\nabla_{\boldsymbol{w}} J_{\text{LS}}(n) = -\beta\boldsymbol{\Phi}^T(n)\boldsymbol{d}^*(n) + \beta\boldsymbol{\Phi}^T(n)\boldsymbol{\Phi}^*(n)\boldsymbol{w}, \tag{12.151}$$

where as usual, $\beta = 1$ (resp., $\beta = 2$) for complex (resp., real) data. Defining

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n) = \boldsymbol{\Phi}^T(n)\boldsymbol{\Phi}^*(n) = \sum_{\ell=0}^{n} \lambda^{n-\ell}\boldsymbol{\phi}(\ell)\boldsymbol{\phi}^H(\ell) \tag{12.152}$$

and

$$\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n) = \boldsymbol{\Phi}^T(n)\boldsymbol{d}^*(n) = \sum_{\ell=0}^{n} \lambda^{n-\ell} d^*(\ell)\boldsymbol{\phi}(\ell), \tag{12.153}$$

in which $\boldsymbol{\phi}(n)$ is defined as before, (12.151) can be rewritten as

$$\nabla_{\boldsymbol{w}} J_{\text{LS}}(n) = -\beta\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n) + \beta\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\boldsymbol{w}. \tag{12.154}$$

Equating $\nabla_{\boldsymbol{w}} J_{\text{LS}}(n)$ to the null vector, we get the normal equations

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\boldsymbol{w}(n+1) = \hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n). \tag{12.155}$$

Note that these are a deterministic version of the normal equations we saw in Section 1.12.2.1. Therefore, to minimize $J_{\text{LS}}(n)$, the weight vector $\boldsymbol{w}(n+1)$ must satisfy

$$\boldsymbol{w}(n+1) = \widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}^{-1}(n)\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n), \tag{12.156}$$

which requires the computation of the inverse correlation matrix $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}^{-1}(n)$ at each time instant. Note that this is only valid if $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}^{-1}(n)$ is nonsingular. In particular, the inverse does not exist if $n < M$ (Box 3). Assuming $\lambda = 1$ and that the signals $x(n)$ and $d(n)$ are jointly stationary and ergodic, we obtain the following steady-state relations

$$\lim_{n\to\infty} \frac{1}{n+1}\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)|_{\lambda=1} = \boldsymbol{R}_{\boldsymbol{\phi}} \quad \text{and} \quad \lim_{n\to\infty} \frac{1}{n+1}\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n)|_{\lambda=1} = \boldsymbol{r}_{d\boldsymbol{\phi}}.$$

Therefore, the deterministic normal equations converge to the stochastic normal equations when $\lambda = 1$, and the weight vector obtained from (12.156) converges to the Wiener solution, i.e.,

$$\lim_{n\to\infty} \boldsymbol{w}(n+1)|_{\lambda=1} = \boldsymbol{R}_{\boldsymbol{\phi}}^{-1}\boldsymbol{r}_{d\boldsymbol{\phi}} = \boldsymbol{w}_{\text{o}}. \tag{12.157}$$

We should mention that with $\lambda = 1$ the convergence rate of RLS goes to zero, i.e., it loses the ability to follow the statistical variations of the observed data. An equivalent result is obtained for LMS or NLMS when the fixed step-size μ or $\tilde{\mu}$ is replaced by $1/n$ [39].

The computational cost of (12.156) is high due to the computation of the inverse matrix. In order to solve the normal equations in real-time and with less cost, we should rewrite (12.152) and (12.153) as recursions, i.e.,

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n) = \lambda\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n), \tag{12.158}$$

and

$$\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n) = \lambda\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1) + d^*(n)\boldsymbol{\phi}(n), \tag{12.159}$$

with initializations $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(-1) = \delta\boldsymbol{I}$ and $\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(-1) = \boldsymbol{0}$, where δ is a small positive constant. Note that (12.159) is identical to (12.153), but (12.158) leads to the estimate

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n) = \lambda^{n+1}\delta\boldsymbol{I} + \sum_{\ell=0}^{n} \lambda^{n-\ell}\boldsymbol{\phi}(\ell)\boldsymbol{\phi}^H(\ell), \tag{12.160}$$

and therefore the initialization guarantees the existence of the inverse at all instants. For $0 \ll \lambda < 1$, the initialization $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(-1) = \delta \boldsymbol{I}$ is forgotten and (12.158) becomes close to (12.152) as n increases. Furthermore, if you want the algorithm to forget quickly the initialization, besides using $0 \ll \lambda < 1$, you should choose δ small (δ^{-1} large). Note however that a certain amount of regularization may be useful in practice, to avoid problems when the input signal has long stretches with low power.

In order to obtain a recursion for the inverse matrix $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}^{-1}(n)$, we can use the matrix inversion lemma (Box 3) by comparing (12.138) with (12.158) and identifying

$$\boldsymbol{A} = \widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n), \quad \boldsymbol{B} = \lambda \widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1), \quad \boldsymbol{C} = \boldsymbol{\phi}(n), \quad \text{and} \quad \boldsymbol{D} = 1.$$

Thus, using (12.139), we obtain

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}^{-1}(n) = \lambda^{-1}\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}^{-1}(n-1) - \frac{\lambda^{-2}\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}^{-1}(n-1)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}^{-1}(n-1)}{1+\lambda^{-1}\boldsymbol{\phi}^H(n)\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}^{-1}(n-1)\boldsymbol{\phi}(n)}. \tag{12.161}$$

Defining $\boldsymbol{P}(n) \triangleq \widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}^{-1}(n)$, with some algebraic manipulations in (12.161), we arrive at

$$\boldsymbol{P}(n) = \frac{1}{\lambda}\left[\boldsymbol{P}(n-1) - \frac{\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)}{\lambda + \boldsymbol{\phi}^H\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)}\right] \tag{12.162}$$

with initialization $\boldsymbol{P}(-1) = \delta^{-1}\boldsymbol{I}$. This equation allow us to compute the inverse correlation matrix in a recurrent form, avoiding the explicit computation of a matrix inverse. However, the implementation of (12.162) in finite precision arithmetic requires some care to avoid numerical problems, as explained in Section 1.12.3.3.1.

To simplify the following equations, it is convenient to define the column vector

$$\boldsymbol{k}(n) = \frac{\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)}{\lambda + \boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)}, \tag{12.163}$$

which is the so-called Kalman gain. Thus, (12.162) can be rewritten as

$$\boldsymbol{P}(n) = \lambda^{-1}\left[\boldsymbol{I} - \boldsymbol{k}(n)\boldsymbol{\phi}^H(n)\right]\boldsymbol{P}(n-1). \tag{12.164}$$

Using (12.164) and some algebraic manipulations in (12.163), we arrive at

$$\begin{aligned}\boldsymbol{k}(n) &= \lambda^{-1}\boldsymbol{P}(n-1)\boldsymbol{\phi}(n) - \lambda^{-1}\boldsymbol{k}(n)\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)\\ &= \lambda^{-1}\left[\boldsymbol{I} - \boldsymbol{k}(n)\boldsymbol{\phi}^H(n)\right]\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)\\ &= \boldsymbol{P}(n)\boldsymbol{\phi}(n).\end{aligned} \tag{12.165}$$

Thus, using (12.159) and (12.165) in (12.156), the weight vector $\boldsymbol{w}(n+1)$ can also be computed recursively, i.e.,

$$\begin{aligned}\boldsymbol{w}(n+1) &= \boldsymbol{P}(n)\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n)\\ &= \lambda\boldsymbol{P}(n)\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1) + \boldsymbol{P}(n)\boldsymbol{\phi}(n)d^*(n)\\ &= \lambda\boldsymbol{P}(n)\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1) + \boldsymbol{k}(n)d^*(n).\end{aligned} \tag{12.166}$$

Table 12.5 Summary of the Conventional RLS Algorithm

Initialization:
Set $\boldsymbol{P}(-1) = \delta^{-1}\boldsymbol{I}$; δ = small positive constant
$\boldsymbol{w}(0) = \mathbf{0}$ and $0 \ll \lambda < 1$
For $n = 0, 1, 2, \ldots$, compute
$\boldsymbol{k}(n) = \dfrac{\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)}{\lambda + \boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)}$
$\hat{y}(n) = \boldsymbol{w}^H(n)\boldsymbol{\phi}(n)$
$e(n) = d(n) - \hat{y}(n)$
$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \boldsymbol{k}(n)e^*(n)$
$\boldsymbol{P}(n) = \lambda^{-1}\left[\boldsymbol{I} - \boldsymbol{k}(n)\boldsymbol{\phi}^H(n)\right]\boldsymbol{P}(n-1)$

Replacing (12.164) in (12.166), we arrive at

$$\begin{aligned}\boldsymbol{w}(n+1) &= \boldsymbol{P}(n-1)\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1) - \boldsymbol{k}(n)\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1) + \boldsymbol{k}(n)d^*(n)\\ &= \boldsymbol{w}(n) - \boldsymbol{k}(n)\boldsymbol{\phi}^H(n)\boldsymbol{w}(n) + \boldsymbol{k}(n)d^*(n)\\ &= \boldsymbol{w}(n) + \boldsymbol{k}(n)[\underbrace{d(n) - \boldsymbol{w}^H(n)\boldsymbol{\phi}(n)}_{e(n)}]^*. \end{aligned} \tag{12.167}$$

The RLS algorithm was derived here as an exact solution for the least-squares problem. However, it also can be interpreted as a *quasi*-Newton algorithm assuming the instantaneous approximation for the gradient vector as in (12.135) and considering $\boldsymbol{P}(n)$ as an approximation for the inverse Hessian matrix. Replacing these approximations in (12.134), we arrive at the same stochastic recursion (12.167).

The RLS algorithm is summarized in Table 12.5. Its computational cost increases with M^2, much faster than those of LMS and NLMS. Different ways of carrying out the calculations may result in a slightly different computational cost, but all with the same order of magnitude, i.e., $\mathcal{O}(M^2)$. Performing the calculations in the following order [3]:

$$\begin{aligned}
&\boldsymbol{\phi}^H(n) \times [\boldsymbol{w}(n)]\\
&d^*(n) - [\boldsymbol{\phi}^H(n)\boldsymbol{w}(n)]\\
&\lambda^{-1}\boldsymbol{\phi}(n)\\
&\boldsymbol{P}(n-1) \times [\lambda^{-1}\boldsymbol{\phi}(n)]\\
&\boldsymbol{\phi}^H(n) \times [\lambda^{-1}\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)]\\
&1 + [\lambda^{-1}\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)]\\
&1/[1 + \lambda^{-1}\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)]\\
&[\lambda^{-1}\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)] \times \left[\frac{1}{1 + \lambda^{-1}\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)}\right]
\end{aligned}$$

$$
\begin{aligned}
&[\lambda^{-1}\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)] \times \left[\frac{\lambda^{-1}\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)}{1+\lambda^{-1}\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)}\right]\\
&\boldsymbol{P}(n)\boldsymbol{\phi}(n) = [\lambda^{-1}\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)] - \left[\frac{\lambda^{-2}\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)}{1+\lambda^{-1}\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)}\right]\\
&[\boldsymbol{P}(n)\boldsymbol{\phi}(n)] \times [d^*(n) - \boldsymbol{\phi}^H(n)\boldsymbol{w}(n)]\\
&\boldsymbol{w}(n+1) = [\boldsymbol{w}(n)] + [\boldsymbol{P}(n)\boldsymbol{\phi}(n)(d^*(n) - \boldsymbol{\phi}^H(n)\boldsymbol{w}(n))],
\end{aligned}
$$

the conventional RLS requires $4M^2+16M+1$ real multiplications, $4M^2+12M-1$ real additions, and one real division per iteration for complex-valued signals. For real-valued data, it requires M^2+5M+1 real multiplications, M^2+3M real additions, and one real division per iteration. Note that the quantities inside brackets were computed in previous steps.

1.12.3.3.1 *Practical implementation of RLS*

The RLS algorithm must be implemented carefully, because it is sensitive to numerical errors, which may accumulate and make the algorithm unstable. The problem can be understood intuitively as follows. In the conventional RLS algorithm, the inverse of the autocorrelation matrix is computed recursively by (expanding (12.161))

$$
\boldsymbol{P}(n) = \lambda^{-1}\boldsymbol{P}(n-1) - \frac{\lambda^{-2}\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)}{1+\lambda^{-1}\boldsymbol{\phi}^H(n)\boldsymbol{P}(n-1)\boldsymbol{\phi}(n)},
$$

i.e., as the difference between two positive (semi-) definite matrices. Note that, since $\lambda \leq 1$, the factor multiplying the current $\boldsymbol{P}(n-1)$ is $\lambda^{-1} \geq 1$. In infinite precision arithmetic, any growth due to this factor will be compensated by the second term. However, in finite precision arithmetic this compensation may not take place, and the factor λ^{-1} may make the recursion numerically unstable—usually, what happens is that $\boldsymbol{P}(n)$ looses its positive character, so the matrix will have a negative eigenvalue, which in its turn leads to the divergence of the coefficients. Thus, to avoid problems one would need to guarantee that $\boldsymbol{P}(n)$ stays symmetric and positive-definite for all time instants n (these properties are known as *backward consistency* of RLS [40,41]). Symmetry can be guaranteed by computing only the upper or lower triangular part of $\boldsymbol{P}(n)$, and copying the result to obtain the rest of elements. However, it is also necessary to guarantee positive-definiteness. Reference [42] describes how this can be achieved.

Due to the fast convergence rate of RLS, a large literature is devoted to finding numerically stable and low-cost (i.e., $\mathcal{O}(M)$) versions of RLS. This is not an easy problem, that required the work of numerous researchers to be solved. Nowadays the user has a very large number of different versions of RLS to choose from. There are versions with formal proofs of stability, versions that work well in practice, but without formal proofs, versions with $\mathcal{O}(M^2)$ complexity, and versions with $\mathcal{O}(M)$ complexity. The interested reader can find more references in Section 1.12.5.1.2.

One approach to ensure the consistency of $\boldsymbol{P}(n)$ uses a property of symmetric and positive-definite matrices (see Box 3): Cholesky factorizations. A symmetric and positive-definite matrix $\boldsymbol{P}(n)$ may always be factored as

$$\boldsymbol{P}(n) = \boldsymbol{\mathcal{P}}(n)\boldsymbol{\mathcal{P}}^H(n), \tag{12.168}$$

where $\boldsymbol{\mathcal{P}}(n)$ is a lower triangular $M \times M$ matrix, called Cholesky factor of $\boldsymbol{P}(n)$. Thus, an algorithm that computes $\boldsymbol{\mathcal{P}}(n)$ instead of $\boldsymbol{P}(n)$ can avoid the numerical instability of the conventional RLS algorithm. There are many algorithms based on this main idea; for some of them a precise stability proof is available [40]. A recent and very complete survey of QR-RLS algorithms is available in [43].

Another approach is available when the regressor sequence $\{\boldsymbol{\phi}(n)\}$ has shift structure. In this case, it is possible to derive *lattice*-based RLS filters that are in practice stable, although no formal proof is available at this time, as described, for example, in [5].

A more practical solution for the implementation of the RLS algorithm was proposed recently in [44]. This approach avoids propagation of the inverse covariance matrix, using an iterative method to solve (12.155) at each step. The computational complexity is kept low by using the previous solution as initialization, and by restricting the majority of the multiplications to multiplications by powers of two (which can be implemented as simple shifts). See Section 1.12.5.1.3.

1.12.3.4 Comparing convergence rates

To compare the convergence of LMS, NLMS, and RLS, we show next two examples. In Example 1, we consider a system identification application and in Example 2, we use these algorithms to update the weights of an equalizer.

As we will see in more detail in Section 1.12.4, adaptive filters are compared based on how well they handle different classes of signals. For example, we could be interested in seeing how two filters handle white noise, or how they handle a certain level of correlation at the input. Given the stochastic nature of the inputs, the performance measures are defined in terms of averaged cost functions, most commonly the mean-square error (MSE), that is, the average of $|e(n)|^2$ over all possible inputs in a given class. These averages can be computed theoretically, as we show in Section 1.12.4, or via simulations.

For example, the MSE as a function of time, that is, the curve $\mathrm{E}\{|e(n)|^2\} \times n$, is called a filter's *learning curve*. A comparison of learning curves obtained theoretically and from simulations can be seen in Figure 12.30 further ahead. When simulations are used, one runs a filter L times for an interval of N samples, starting always from the same conditions. For each run ℓ, the error $e^{(\ell)}(n)$ is computed for $1 \le n \le N$, and one obtains the so-called *ensemble-average learning curve*

$$\widehat{E}(n) \triangleq \frac{1}{L}\sum_{\ell=1}^{L} |e^{(\ell)}(n)|^2.$$

Usually the ensemble-average learning curve will be a reasonable approximation of the true learning curve for a reasonably small value of L (say, from 50 to 1000), but in some situations this may not be true—see [45,46] for examples and a detailed explanation.

Example 1. The aim of this example is to identify the system $\boldsymbol{w}_\mathrm{o} = [1\ -0.8]^T$, assuming that the regressor $\boldsymbol{\phi}(n)$ is obtained from a process $x(n)$ generated with a first-order autoregressive model, whose transfer function is $1/(1-0.8z^{-1})$. This model is fed with an i.i.d. Gaussian random process with unitary variance. Moreover, additive i.i.d. noise $v(n)$ with variance $\sigma_v^2 = 0.01$ is added to form the desired signal.

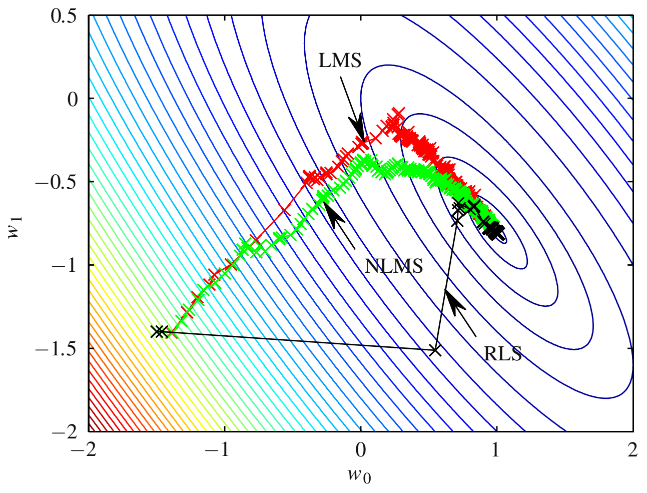


FIGURE 12.25

LMS ($\mu = 0.01$), NLMS ($\tilde{\mu} = 0.05$, $\varepsilon = 0.5$) and RLS ($\lambda = 0.99$, $\delta = 1$) initialized at $\boldsymbol{w}(0) = [-1.5\ \ -1.4]^T$, $\sigma_v^2 = 0.01$. Click the animation video file.

Considering an adaptive filter with $M = 2$ coefficients, we can obtain a contour plot of the mean-square error cost as a function of w_0 and w_1 as shown in Figure 12.25. The animations in Figure 12.25 illustrate the behavior of LMS ($\mu = 0.01$), NLMS ($\tilde{\mu} = 0.05$, $\varepsilon = 0.5$), and RLS ($\lambda = 0.99$, $\delta = 1$) initialized at $\boldsymbol{w}(0) = [-1.5 - 1.4]^T$. The correspondent curves of MSE along the iterations, estimated from the ensemble-average of 1000 independent runs, are shown in Figure 12.26. As expected for a colored input signal, RLS converges much faster than NLMS and LMS. NLMS converges faster than LMS and achieves the solution through a different path. To obtain a good behavior of NLMS in this example, we had to choose a relatively large regularization factor ε. This is always necessary when NLMS is used with few coefficients since in this case, $\|\boldsymbol{\phi}(n)\|^2$ has a high probability of becoming too close to zero and de-stabilizing the algorithm. This is shown in the analysis of Section 1.12.4.

Example 2. Now, we assume the transmission of a QPSK (quadrature phase shift-keying) signal, whose symbols belong to the alphabet $\{\pm 1 \pm j1\}$. This signal suffers the effect of the channel (taken from [47])

$$\begin{aligned} H(z) = (-0.2 + j0.3) + (-0.5 + j0.4)z^{-1} + (0.7 - j0.6)z^{-2} \\ +(0.4 + j0.3)z^{-3} + (0.2 + j0.1)z^{-4} \end{aligned} \tag{12.169}$$

and of additive white Gaussian noise (AWGN) under a signal-to-noise ratio (SNR) of 30 dB. The equalizer is an adaptive filter with $M = 15$ coefficients initialized with zero and the desired signal is the transmitted sequence delayed by $L = 9$ samples (go back to Figure 12.16 to see the equalization scheme). The adaptation parameters (μ, $\tilde{\mu}$, ε, λ) were chosen in order to obtain the same steady-state performance of LMS, NLMS, and RLS (note that with a longer filter, the regularization parameter ε can be much

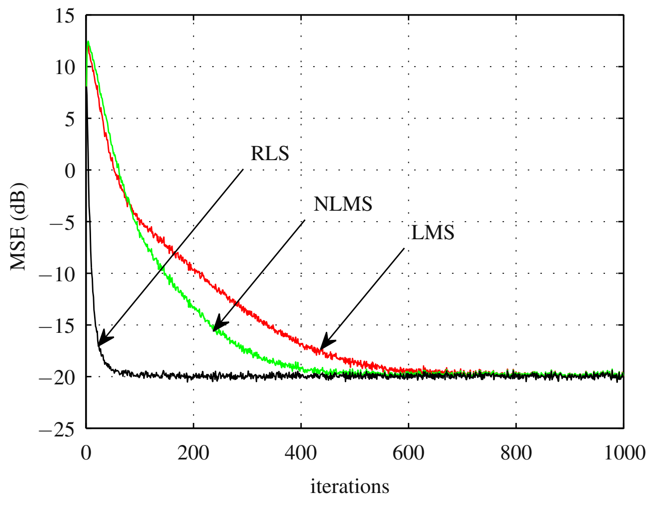


FIGURE 12.26

MSE along the iterations for LMS ($\mu = 0.01$), NLMS ($\tilde{\mu} = 0.05$, $\varepsilon = 0.5$) and RLS ($\lambda = 0.99$, $\delta = 1$) initialized at $\boldsymbol{w}(0) = [-1.5 \;\; -1.4]^T$, $\sigma_v^2 = 0.01$, and ensemble-average of 1000 independent runs.

smaller than in the previous example). Figure 12.27 shows the MSE along the iterations, estimated from the ensemble-average of 1000 independent runs. Again, RLS presents the fastest convergence rate, followed by NLMS and LMS. These algorithms achieve the same steady-state MSE and therefore, present the same performance after convergence, which can be observed in Figure 12.28 where the equalizer output signal constellations are shown. The Matlab file used in this simulation is available at http://www.lps.usp.br.

1.12.4 Statistical analysis

Closed-form expressions for the mean-square performance of adaptive filters facilitate comparisons between different algorithms and provide information about stability, convergence rate, mean-square error, and tracking capability. Since adaptive algorithms are obtained from stochastic approximations of the cost-functions, gradients and Hessians, their performance will degrade in comparison with the performance of the exact algorithms. Thus, stochastic analyses are also important to measure the performance degradation in relation to the exact algorithms.

The analysis of an adaptive algorithm constitutes a very hard problem, since adaptive filters are time-variant, stochastic, and nonlinear systems. Therefore, it is common to introduce simplifying approximations. They consist in most cases of disregarding the dependence between variables, for example, approximating the expected value of the product of two random variables, say, $\mathrm{E}\{xy\}$, by the product of the expected values, $\mathrm{E}\{x\}\mathrm{E}\{y\}$. For historical reasons, these approximations are usually referred to as *assumptions* in the literature.

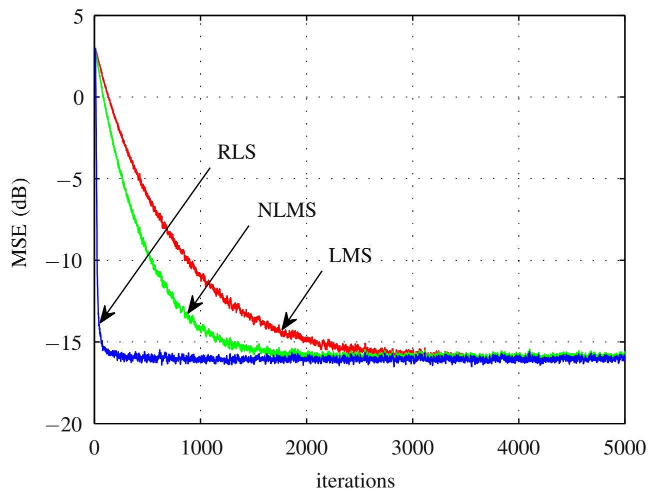


FIGURE 12.27

MSE along the iterations for LMS ($\mu = 10^{-3}$), NLMS ($\tilde{\mu} = 0.08$, $\varepsilon = 10^{-5}$), and RLS ($\lambda = 0.997$, $\delta = 1$) initialized at $\boldsymbol{w}(0) = \mathbf{0}$; QPSK, $L = 9$, $M = 15$, and ensemble-average of 1000 independent runs; equalization of the channel (12.169).

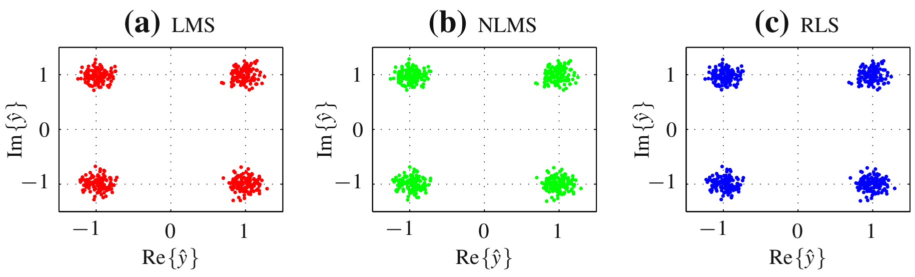


FIGURE 12.28

Equalizer output signal constellations after convergence for (a) LMS ($\mu = 10^{-3}$), (b) NLMS ($\tilde{\mu} = 0.08$, $\varepsilon = 10^{-5}$), and (c) RLS ($\lambda = 0.997$, $\delta = 1$); initialized at $\boldsymbol{w}(0) = \mathbf{0}$; QPSK, $L = 9$, $M = 15$; equalization of the channel (12.169).

In some situations, in particular for LMS and some of its variants, the approximations are proved to converge to the exact solution in the limiting case of slow adaptation ($\mu \to 0$ in the case of LMS), and their predictions tend to be reasonable for the range of step-sizes used in practice. We will not reproduce these proofs here, since they are in general very technical, but they can be found in [48–53] for the case of LMS, and [39] for more general filters.

The literature contains a few exact analytical results, but only for LMS, under very particular conditions [54–56]. In addition, the complexity of these exact models grows very quickly with the number of coefficients, so they are only useful for short filters (up to about five coefficients.).

The performance of an adaptive filter is usually described by evaluating the mean and the mean square (variance) of the weight error vector and of the error signal $e(n)$. This usually works very well for slow adaptation (i.e., small step-size). However, for fast adaptation, the behavior of an adaptive filter is considerably more complicated than can be seen by looking only at means and variances. Some of the phenomena that appear in LMS with large step-sizes are described in [46,57]. References [45,58–60] study algorithms with strong nonlinearities, for which the usual approximations are not always valid.

The analysis of NLMS was considered in [61–68]. Many works consider only tapped-delay line regressors, the case of Volterra filters is considered for example in [69–73]. For RLS filters, important models for infinite-precision arithmetic can be found in [74,75]. The case of finite-precision arithmetic is discussed in Section 1.12.5.1.2.

The real importance of the approximations used in the adaptive filtering literature is that they lead to reasonably simple models that capture well the overall behavior of an adaptive filter. Simple models provide intuition about how a filter will behave if a given parameter is changed, which is important for a designer to make good decisions. Some approximations are made because without them the mathematical analysis would not be feasible, but others are used mainly because they lead to simpler models. Of course, there are situations in which it is important to study a filter in more detail, reducing the number of assumptions used in the analysis, as in the references just cited.

In this section we analyze three of the most common adaptive filters—LMS, NLMS, and RLS, starting with LMS in Section 1.12.4.3, and then showing how to extend these results to the other algorithms in Section 1.12.4.4. To start, we need to define some useful performance measures. The *mean-square error* (MSE) is simply the expected value of $|e(n)|^2$:

$$J(n) \triangleq \mathrm{E}\{|e(n)|^2\}. \tag{12.170}$$

The most widely measure of performance used in the literature of adaptive filtering is the *excess* mean-square error (EMSE), which is defined as

$$\zeta(n) \triangleq \mathrm{E}\{|e_a(n)|^2\}, \tag{12.171}$$

where the *excess a priori error* is given by

$$e_a(n) = \tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n), \tag{12.172}$$

and the weight error vector by

$$\tilde{\boldsymbol{w}}(n) = \boldsymbol{w}_{\mathrm{o}}(n) - \boldsymbol{w}(n). \tag{12.173}$$

Note that the optimal solution $\boldsymbol{w}_{\mathrm{o}}(n)$ is now allowed to be time-variant. Note also the difference between $e_a(n)$ and $e(n)$, defined in (12.119). Both are *a priori* errors, but $e_a(n)$ is computed with $\tilde{\boldsymbol{w}}(n)$, whereas $e(n)$ is computed with $\boldsymbol{w}(n)$. The EMSE measures how much $J(n)$ exceeds its minimum value $J_{\min}(n)$ due to adaptation: if you were able to find the optimum filter coefficients $\boldsymbol{w}_{\mathrm{o}}(n)$ at each time, the EMSE would be zero. Since the actual filter coefficients are never exactly equal to the optimum values, the

EMSE measures the effect of this difference on the error variance. Closely related to the EMSE, the *misadjustment* is defined as

$$\mathcal{M}(n) \triangleq \zeta(n)/\sigma_v^2. \tag{12.174}$$

The *excess a posteriori error* $e_p(n)$ is mostly used in the analysis and derivations of algorithms. It is the error obtained *after* using the current data $\boldsymbol{\phi}(n)$ to update the coefficient vector (thus the names *a priori* and *a posteriori*), i.e.,

$$e_p(n) = \tilde{\boldsymbol{w}}^H(n+1)\boldsymbol{\phi}(n). \tag{12.175}$$

Again, note the difference with respect to $\xi(n)$, defined in (12.129).

Another common performance measure is the *mean-square deviation* (MSD). It measures how far $\boldsymbol{w}(n)$ is from $\boldsymbol{w}_\text{o}(n)$ in average:

$$\chi(n) \triangleq \mathrm{E}\{\|\tilde{\boldsymbol{w}}(n)\|^2\}. \tag{12.176}$$

The EMSE is the most frequently performance measure used for comparisons between adaptive filters; however in some applications, particularly for system identification, the MSD is more adequate.

We now will discuss a general model for the input signals, considering a set of approximations that leads to reasonable, yet simple models.

1.12.4.1 Data model

From the orthogonality principle (recall Section 1.12.2), the optimal solution $\boldsymbol{w}_\text{o}(n)$ satisfies the property

$$\mathrm{E}\{\boldsymbol{\phi}(n)[d(n) - \boldsymbol{w}_\text{o}^H(n)\boldsymbol{\phi}(n)]\} = \boldsymbol{0}, \tag{12.177}$$

i.e., the input regressor vector $\boldsymbol{\phi}(n)$ is uncorrelated with the optimal estimation error

$$v_\text{o}(n) \triangleq d(n) - \boldsymbol{w}_\text{o}^H(n)\boldsymbol{\phi}(n), \tag{12.178}$$

whose mean is zero and whose variance $\sigma_v^2(n) = \mathrm{E}\{|v_\text{o}(n)|^2\}$ is equal to the minimum cost $J_\text{min}(n)$. Thus, a linear regression model for $d(n)$ holds, i.e.,

$$d(n) = \boldsymbol{w}_\text{o}^H(n)\boldsymbol{\phi}(n) + v_\text{o}(n). \tag{12.179}$$

It then follows that

$$e(n) = d(n) - \boldsymbol{w}^H(n)\boldsymbol{\phi}(n) = \tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n) + v_\text{o}(n) = e_a(n) + v_\text{o}(n), \tag{12.180}$$

and the MSE is given by

$$\begin{aligned} J(n) = \mathrm{E}\{|e(n)|^2\} &= \mathrm{E}\{|e_a(n)|^2\} + 2\mathrm{E}\{e_a(n)v_\text{o}(n)\} + \mathrm{E}\{|v_\text{o}(n)|^2\} = \\ &= \zeta(n) + 2\mathrm{E}\{e_a(n)v_\text{o}(n)\} + \sigma_v^2(n). \end{aligned}$$

The first approximation (or assumption) used in adaptive filtering analysis is to disregard the cross-term $\mathrm{E}\{e_a(n)v_\text{o}(n)\}$. Since $v_\text{o}(n)$ and $\boldsymbol{\phi}(n)$ are uncorrelated, but not necessarily independent, the cross-term does not vanish in general. From observations, however, one notes that the cross-term is usually negligible, compared to the other terms. In order to obtain a simple model for the adaptive filter, we will assume that $\boldsymbol{\phi}(n)$ and $v_\text{o}(n)$ are independent of each other, and of previous samples $\boldsymbol{\phi}(n-k)$, $v(n-k)$ for $k > 0$:

Assumption 1. The sequences $\{\boldsymbol{\phi}(n)\}$ and $\{v_o(n)\}$ are independent and identically distributed (iid), and independent of each other (not only orthogonal). In addition, the sequence $\{\boldsymbol{\phi}(n)\}$ is assumed stationary and $\{v_o(n)\}$ is zero-mean.

The last two assumptions are not in fact necessary, but they simplify the analysis. In general the sequence $\{\boldsymbol{\phi}(n)\}$ is not iid: for example if $\boldsymbol{\phi}(n)$ is formed from a tap-delay line or from a Volterra series expansion (see Eqs. (12.113) and (12.114)) $\boldsymbol{\phi}(n)$ and $\boldsymbol{\phi}(n-1)$ will have terms in common. However, assuming that the sequences are iid leads to simple models that in general are good approximations for the range of step-sizes usually found in practice, in particular for small step-sizes. The analysis based on Assumption 1 is known in the literature as *independence theory*.

Going back to the adaptive filter recursions (see Tables 12.2–12.5, and also Eq. (12.221) further ahead) we see that the current weight error $\tilde{\boldsymbol{w}}(n)$ is a function of past inputs $\boldsymbol{\phi}(n-1), \boldsymbol{\phi}(n-2), \ldots$ and noise samples $v(n-1), v(n-2), \ldots$, but *not* of the current samples $\boldsymbol{\phi}(n)$ and $v_o(n)$. Therefore, under Assumption 1, $\tilde{\boldsymbol{w}}(n)$ is independent of $v_o(n)$ and $\boldsymbol{\phi}(n)$, so that

$$\mathrm{E}\{e_a(n)v_o(n)\} = \mathrm{E}\{\tilde{\boldsymbol{w}}^H(n)\}\mathrm{E}\{\boldsymbol{\phi}(n)\}\mathrm{E}\{v_o(n)\} = 0.$$

We conclude that under Assumption 1, the MSE $J(n) = \mathrm{E}\{|e(n)|^2\}$ is related to the EMSE via

$$J(n) = \zeta(n) + J_{\min}(n) = \zeta(n) + \sigma_v^2, \tag{12.181}$$

since Assumption 1 implies that $J_{\min}(n)$ is a constant.

We also need a model for the variation of the optimal solution. Again, our choice is a model that captures the main consequences of a non-constant $\boldsymbol{w}_o(n)$, without unnecessarily complicating the analysis.

Assumption 2 (Random-walk model). In a nonstationary environment, $\boldsymbol{w}_o$ varies as

$$\boldsymbol{w}_o(n+1) = \boldsymbol{w}_o(n) + \boldsymbol{q}(n), \tag{12.182}$$

where $\boldsymbol{q}(n)$ is zero-mean i.i.d vector with positive-definite autocorrelation matrix $\boldsymbol{Q} = \mathrm{E}\{\boldsymbol{q}(n)\boldsymbol{q}^H(n)\}$, independent of the initial condition $\boldsymbol{w}(0)$, and of $\{\boldsymbol{\phi}(\ell), v(\ell)\}$ for all ℓ.

The covariance matrix of $\boldsymbol{w}_o$, according to (12.182), grows slowly to infinity. A more general model for the variation of $\boldsymbol{w}_o$ uses the recursion

$$\begin{cases} \boldsymbol{w}_o(n+1) = \boldsymbol{w}_o(n) + \boldsymbol{\theta}(n), \\ \boldsymbol{\theta}(n) = \alpha\boldsymbol{\theta}(n-1) + \boldsymbol{q}(n), \quad 0 \le |\alpha| < 1, \end{cases}$$

whose covariance matrix remains bounded, but the analysis becomes more complicated [3]. Therefore, it is common in the literature to adopt the model (12.182). Note that both models assume that $\{\boldsymbol{x}(n)\}$ is a stationary random sequence, although $\boldsymbol{w}_o(n)$ is allowed to vary with n.

1.12.4.2 Relating the covariance matrix of the weight-error vector to the EMSE and the MSD

The covariance matrix of the weight-error vector is defined as

$$\boldsymbol{S}(n) \triangleq \mathrm{E}\left\{\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\right\}. \tag{12.183}$$

This matrix is important in the statistical analysis of adaptive filters since it is related to the MSD and the EMSE as we show next.

Recall that the MSD is given by

$$\chi(n) = \mathrm{E}\left\{\|\tilde{\boldsymbol{w}}(n)\|^2\right\} = \sum_{i=0}^{M-1} \mathrm{E}\left\{\tilde{w}_i^2(n)\right\}.$$

Now, the terms $\mathrm{E}\{\tilde{w}_i^2(n)\}$ are exactly the terms that appear in the diagonal of $\boldsymbol{S}(n)$, therefore, recalling that the *trace* of a square matrix is the sum of its main-diagonal elements (see Box 3), we obtain

$$\chi(n) = \sum_{i=0}^{M-1} \mathrm{E}\{\tilde{w}_i^2(n)\} = \sum_{m=0}^{M-1} s_{mm}(n) = \mathrm{Tr}(\boldsymbol{S}(n)). \tag{12.184}$$

Under Assumption 1, $\boldsymbol{\phi}(n)$ and $\tilde{\boldsymbol{w}}(n)$ are independent, as we just saw. Therefore,

$$\begin{aligned}\zeta(n) &= \mathrm{E}\{|\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)|^2\} = \mathrm{E}\{\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)(\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n))^*\}\\ &= \mathrm{E}\{\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n)\} = \mathrm{E}\left\{\mathrm{E}\{\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n)|\tilde{\boldsymbol{w}}(n)\}\right\}\\ &= \mathrm{E}\left\{\tilde{\boldsymbol{w}}^H(n)\mathrm{E}\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)|\tilde{\boldsymbol{w}}(n)\}\tilde{\boldsymbol{w}}(n)\right\} = \mathrm{E}\{\tilde{\boldsymbol{w}}^H(n)\boldsymbol{R}_{\boldsymbol{\phi}}\tilde{\boldsymbol{w}}(n)\},\end{aligned}$$

where we used Assumption 1 to write $\mathrm{E}\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\} = \boldsymbol{R}_{\boldsymbol{\phi}}$ (constant), and two properties of expected values. The first is that, for any two random variables a, b,

$$\mathrm{E}\{ab\} = \mathrm{E}\{\mathrm{E}\{ab|b\}\} = \mathrm{E}\{\mathrm{E}\{a|b\}b\},$$

where the inner expected value is taken with respect to a, and the second with respect to b. The second property is that, if a and b are independent, then $\mathrm{E}\{a|b\}$ does not depend on b, i.e., $\mathrm{E}\{a|b\} = \mathrm{E}\{a\}$.

To proceed, we apply a useful trick from matrix theory. You can check by direct computation that, if $\boldsymbol{A} \in \mathcal{C}^{K\times L}$ and $\boldsymbol{B} \in \mathcal{C}^{L\times K}$, then $\mathrm{Tr}(\boldsymbol{AB}) = \mathrm{Tr}(\boldsymbol{BA})$. Applying this property with $\boldsymbol{A} \leftarrow \tilde{\boldsymbol{w}}^H(n)$ and $\boldsymbol{B} \leftarrow \boldsymbol{R}_{\boldsymbol{\phi}}\tilde{\boldsymbol{w}}(n)$, we obtain

$$\zeta(n) = \mathrm{Tr}\left(\boldsymbol{R}_{\boldsymbol{\phi}}\mathrm{E}\{\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\}\right) = \mathrm{Tr}\left(\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\right). \tag{12.185}$$

Since $\boldsymbol{R}_{\boldsymbol{\phi}}$ is symmetric, there always exists an orthogonal matrix $\boldsymbol{U}$ (i.e., $\boldsymbol{U}^{-1} = \boldsymbol{U}^H$) that diagonalizes $\boldsymbol{R}_{\boldsymbol{\phi}}$, that is,

$$\boldsymbol{U}^H\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{U} = \begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \lambda_M\end{bmatrix} \triangleq \boldsymbol{\Lambda}, \tag{12.186}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_M$ are the eigenvalues of $\boldsymbol{R}_{\boldsymbol{\phi}}$ (see Box 3). Defining the rotated matrix

$$\boldsymbol{S}'(n) \triangleq \boldsymbol{U}^H\boldsymbol{S}(n)\boldsymbol{U}$$

and recalling that $\boldsymbol{U}\boldsymbol{U}^H = \boldsymbol{I}$, (12.184) and (12.185) can be rewritten respectively as

$$\chi(n) = \mathrm{Tr}(\boldsymbol{U}\underbrace{\boldsymbol{U}^H\boldsymbol{S}(n)\boldsymbol{U}}_{\boldsymbol{S}'(n)}\boldsymbol{U}^H) = \mathrm{Tr}(\boldsymbol{U}\boldsymbol{S}'(n)\boldsymbol{U}^H) \tag{12.187}$$

and

$$\zeta(n) = \mathrm{Tr}(\boldsymbol{U}\underbrace{\boldsymbol{U}^H\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{U}}_{\boldsymbol{\Lambda}}\underbrace{\boldsymbol{U}^H\boldsymbol{S}(n)\boldsymbol{U}}_{\boldsymbol{S}'(n)}\boldsymbol{U}^H) = \mathrm{Tr}(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{S}'(n)\boldsymbol{U}^H). \tag{12.188}$$

Using again the fact that $\mathrm{Tr}(\boldsymbol{A}\boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{B}\boldsymbol{A})$, with $\boldsymbol{A} \leftarrow \boldsymbol{U}$, we obtain

$$\chi(n) = \mathrm{Tr}\left(\boldsymbol{S}'(n)\right) \tag{12.189}$$

and

$$\zeta(n) = \mathrm{Tr}\left(\boldsymbol{\Lambda}\boldsymbol{S}'(n)\right). \tag{12.190}$$

Note that both the MSD (12.189) and the EMSE (12.190) depend only on the diagonal entries of $\boldsymbol{S}'(n)$. We can work therefore only with these diagonal entries, defining the vectors

$$\boldsymbol{s}'(n) = \mathrm{diag}(\boldsymbol{S}'(n)) = \left[s_0'(n)s_1'(n)\cdots s_{M-1}'(n)\right]^T, \tag{12.191}$$

and

$$\boldsymbol{l} = \left[\lambda_0, \ldots, \lambda_{M-1}\right]^T, \tag{12.192}$$

where $\mathrm{diag}(\boldsymbol{A})$ represents a column vector with the diagonal elements of $\boldsymbol{A}$, and evaluating the MSD and EMSE respectively as

$$\chi(n) = \sum_{k=0}^{M-1} s_k'(n) \tag{12.193}$$

and

$$\zeta(n) = \boldsymbol{l}^T\boldsymbol{s}'(n) = \sum_{k=0}^{M-1} \lambda_k s_k'(n). \tag{12.194}$$

Again, the MSD and EMSE depend only on the elements of the diagonal of the rotated matrix $\boldsymbol{S}'(n)$. In the case of EMSE, these elements are weighed by the eigenvalues of the autocorrelation matrix $\boldsymbol{R}_{\boldsymbol{\phi}}$. As we shall see in Section 1.12.4.3, the stability of LMS can be studied through a recursion for the vector $\boldsymbol{s}'(n)$.

Thus, according to the previous results, in order to obtain closed-form expressions for the MSD and the EMSE of an adaptive algorithm, one path is to find a recursion for the covariance matrix $\boldsymbol{S}(n)$ or for the vector $\boldsymbol{s}'(n)$. However, this is not the only method to analyze adaptive algorithms, as we shall see in Section 1.12.4.4.

1.12.4.3 Statistical analysis of the LMS algorithm

In this section, we present a statistical analysis for the LMS algorithm, assuming that the optimal solution $\boldsymbol{w}_{\rm o}$ remains constant ($\boldsymbol{q}(n) \equiv \boldsymbol{0}$, for all n). This analysis predicts the behavior of LMS in steady-state and in transient, and provides a range of the step-size μ for which LMS operates adequately. If you are familiar with the LMS analysis, go to Section 1.12.4.4, where we present a unified analysis for LMS, NLMS, and RLS algorithms in a nonstationary environment.

We first rewrite (12.123) in terms of the weight-error vector $\tilde{\boldsymbol{w}}$. Thus, subtracting both sides of (12.123) from $\boldsymbol{w}_{\rm o}$, we obtain

$$\begin{aligned}\boldsymbol{w}_{\rm o} - \boldsymbol{w}(n+1) &= \boldsymbol{w}_{\rm o} - \boldsymbol{w}(n) - \mu\boldsymbol{\phi}(n)e^*(n),\\ \tilde{\boldsymbol{w}}(n+1) &= \tilde{\boldsymbol{w}}(n) - \mu\boldsymbol{\phi}(n)e^*(n).\end{aligned} \tag{12.195}$$

Replacing $e^*(n) = e_a^*(n) + v_{\rm o}^*(n) = \boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n) + v_{\rm o}^*(n)$ in (12.195), we arrive at

$$\tilde{\boldsymbol{w}}(n+1) = \left[\boldsymbol{I} - \mu\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right]\tilde{\boldsymbol{w}}(n) - \mu\boldsymbol{\phi}(n)v_{\rm o}^*(n). \tag{12.196}$$

First-order analysis. Taking expectations on both sides of (12.196), we obtain

$$\begin{aligned}\mathrm{E}\{\tilde{\boldsymbol{w}}(n+1)\} &= \mathrm{E}\left\{\left[\boldsymbol{I} - \mu\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right]\tilde{\boldsymbol{w}}(n)\right\} - \mu\mathrm{E}\{\boldsymbol{\phi}(n)v_{\rm o}^*(n)\}\\ &= \mathrm{E}\left\{\left[\boldsymbol{I} - \mu\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right]\tilde{\boldsymbol{w}}(n)\right\},\end{aligned} \tag{12.197}$$

where the last equality follows from the fact that $\boldsymbol{\phi}(n)$ is orthogonal to $v_{\rm o}^*(n)$, according to the orthogonality condition (see Eq. (12.96)). Since $\tilde{\boldsymbol{w}}(n)$ is assumed to be independent of $\boldsymbol{\phi}(n)$ (see Assumption 1), we arrive at

$$\mathrm{E}\{\tilde{\boldsymbol{w}}(n+1)\} = \left[\boldsymbol{I} - \mu\boldsymbol{R}_{\boldsymbol{\phi}}\right]\mathrm{E}\{\tilde{\boldsymbol{w}}(n)\}. \tag{12.198}$$

This is the same recursion we saw in Box 1. According to linear systems theory, this recursion converges as long as the largest eigenvalue of $\boldsymbol{A} = \boldsymbol{I} - \mu\boldsymbol{R}_{\boldsymbol{\phi}}$ has absolute value less than one, which implies (see Box 1)

$$\begin{aligned}&|1-\mu\lambda| < 1 \Leftrightarrow 0 < \mu < \frac{2}{\lambda_k}, \quad \forall k \Leftrightarrow\\ &0 < \mu < \frac{2}{\lambda_{\max}},\end{aligned} \tag{12.199}$$

where λ_k, $k = 1, 2, \ldots, M$ are the eigenvalues of $\boldsymbol{R}_{\boldsymbol{\phi}}$, and $\lambda_{\max}$ is its maximum eigenvalue. We should notice that this range for μ does not ensure the stability of LMS, since it ensures the convergence of the mean of $\tilde{\boldsymbol{w}}(n) = \boldsymbol{w}_o - \boldsymbol{w}(n)$ towards 0, but the autocovariance of $\tilde{\boldsymbol{w}}(n)$ may still be diverging. We show next that the range for μ based on a second-order analysis is indeed more restrict.

Second-order analysis. Now, we use (12.196) to find a recursion for $\boldsymbol{S}(n)$ and (12.185) to evaluate the EMSE of LMS. Multiplying (12.196) by its Hermitian, taking the expectations of both sides, we arrive at

$$\begin{aligned}
\mathrm{E}\{\tilde{\boldsymbol{w}}(n+1)\tilde{\boldsymbol{w}}^H(n+1)\} = \mathrm{E}\{\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\}&\\
&-\overbrace{\mu\mathrm{E}\{\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\}}^{\mathcal{A}}\\
&-\overbrace{\mu\mathrm{E}\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\}}^{\mathcal{B}}\\
&+\overbrace{\mu^2\mathrm{E}\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\}}^{\mathcal{C}}\\
&+\overbrace{\mu^2\mathrm{E}\{|v_o(n)|^2\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\}}^{\mathcal{D}}\\
&-\overbrace{\mu\mathrm{E}\{v_o(n)\tilde{\boldsymbol{w}}(n)\boldsymbol{\phi}^H(n)\}}^{\mathcal{E}}\\
&-\overbrace{\mu\mathrm{E}\{v_o^*(n)\boldsymbol{\phi}(n)\tilde{\boldsymbol{w}}^H(n)\}}^{\mathcal{F}}\\
&+\overbrace{\mu^2\mathrm{E}\{v_o(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n)\boldsymbol{\phi}^H(n)\}}^{\mathcal{G}}\\
&+\overbrace{\mu^2\mathrm{E}\{v_o^*(n)\boldsymbol{\phi}(n)\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\}}^{\mathcal{H}}.
\end{aligned} \tag{12.200}$$

Now, using Assumption 1, we can evaluate the terms $\mathcal{A}-\mathcal{H}$ of (12.200):

$\mathcal{A}$- Recalling that Assumption 1 implies that $\tilde{\boldsymbol{w}}(n)$ is independent from $\boldsymbol{\phi}(n)$, the term $\mathcal{A}$ can be approximated by

$$\begin{aligned}
\mathcal{A} &= \mu\mathrm{E}\left\{\mathrm{E}\left\{\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)|\boldsymbol{\phi}(n)\right\}\right\}\\
&\approx \mu\mathrm{E}\left\{\mathrm{E}\left\{\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\right\}\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}\\
&= \mu\boldsymbol{S}(n)\mathrm{E}\left\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\} = \mu\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}.
\end{aligned} \tag{12.201}$$

$\mathcal{B}$- Analogously, we obtain for $\mathcal{B}$

$$\mathcal{B} \approx \mu\mathrm{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n). \tag{12.202}$$

$\mathcal{C}$- Using Assumption 1, the term $\mathcal{C}$ can be rewritten as

$$\begin{aligned}
\mathcal{C} &= \mu^2\mathrm{E}\left\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}\\
&\approx \mu^2\mathrm{E}\left\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{S}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}.
\end{aligned} \tag{12.203}$$

We return to this term further on.

$\mathcal{D}$- Using the fact that $v_o(n)$ is independent of $\boldsymbol{\phi}(n)$ and $\tilde{\boldsymbol{w}}(n)$, $\mathcal{D}$ reduces to

$$\mathcal{D} = \mu^2 \mathrm{E}\{|v_o(n)|^2 \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\} \approx \mu^2\sigma_v^2 \boldsymbol{R}_{\boldsymbol{\phi}}. \tag{12.204}$$

$\mathcal{E}$- Using the fact that $v_o(n)$ is a zero-mean random variable, which is independent of $\boldsymbol{\phi}(n)$ and $\tilde{\boldsymbol{w}}(n)$, $\mathcal{E}$ is a $M \times M$ null matrix. Using the same arguments, the terms $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{H}$ are also null matrices. We should point out that the zero-mean assumption for $v_o(n)$ is not necessary, but it really facilitates the analysis. Furthermore, as we saw in Section 1.12.2.1.4, we can enforce $\mathrm{E}\{v_o(n)\} = 0$ using simple methods.

From the previous results, (12.200) reduces to

$$\begin{aligned}\mathbf{S}(n+1) \approx\ & \mathbf{S}(n) - \mu \boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}} - \mu \boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n) \\ & + \underbrace{\mu^2 \mathrm{E}\left\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{S}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}}_{\mathcal{C}} + \mu^2\sigma_v^2 \boldsymbol{R}_{\boldsymbol{\phi}}.\end{aligned} \tag{12.205}$$

This recursion is convenient to obtain closed-form expressions for the EMSE and the MSD of LMS. Observe that the four first terms on the right-hand side of (12.205) are linear in $\boldsymbol{S}(n)$. Assuming that μ is sufficiently small, the term $\mathcal{C}$ can be neglected with respect to the three first terms. Thus, for small step-sizes, (12.205) reduces to

$$\mathbf{S}(n+1) \approx \mathbf{S}(n) - \mu\left[\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}} + \boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\right] + \mu^2\sigma_v^2\boldsymbol{R}_{\boldsymbol{\phi}}, \tag{12.206}$$

with initialization $\boldsymbol{S}(0) = \left[\mathbf{w}_o - \boldsymbol{w}(0)\right]\left[\mathbf{w}_o - \boldsymbol{w}(0)\right]^H$.

Since we can obtain the EMSE and the MSD from $\boldsymbol{S}(n)$ and $\boldsymbol{R}_{\boldsymbol{\phi}}$ through (12.185) and (12.184), we should use recursion (12.206) to compute $\boldsymbol{S}(n)$ and then, evaluate $\zeta(n) = \mathrm{Tr}(\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}})$ and $\chi(n) = \mathrm{Tr}(\boldsymbol{S}(n))$. Particularly for $n \to \infty$, since $\boldsymbol{S}(n+1) \approx \boldsymbol{S}(n)$, we obtain

$$\boldsymbol{S}(\infty)\boldsymbol{R}_{\boldsymbol{\phi}} + \boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(\infty) = \mu\sigma_v^2\boldsymbol{R}_{\boldsymbol{\phi}}. \tag{12.207}$$

Taking the trace of both sides of (12.207), we arrive at an analytical expression for the steady-state EMSE of LMS, i.e.,

$$\zeta^{\mathrm{LMS}}(\infty) \approx \frac{\mu\sigma_v^2\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}})}{2} \text{ (for sufficiently small } \mu). \tag{12.208}$$

This expression indicates that the EMSE of LMS increases with μ and with the length of the filter. The dependence with the filter length is through the trace of $\boldsymbol{R}_{\phi}$. It is more evident when the class of filters we are considering is FIR, that is, when $\boldsymbol{\phi}(n) = \left[x(n)\ \ldots x(n-M+1)\right]^T$ (a tapped-delay line). In this case, if $\{x(n)\}$ is stationary, then $\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}}) = M\mathrm{E}\{x^2(n)\}$, M times the average power of $x(n)$.

Turning now to the filter coefficients, if we multiply both sides of (12.207) from the right by $\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}$, taking the trace of both sides, and recalling that $\mathrm{Tr}(\boldsymbol{AB}) = \mathrm{Tr}(\boldsymbol{BA})$, we obtain an expression for the steady-state MSD of LMS, i.e.,

$$\chi^{\mathrm{LMS}}(\infty) \approx \frac{\mu\sigma_v^2 M}{2} \text{ (for sufficiently small } \mu). \tag{12.209}$$

Again, the smaller the step-size μ and the filter length M, the smaller the MSD of LMS.

These results explain what we observed in the simulations in Section 1.12.1.2 (Figures 12.9 and 12.10): for larger values of μ the filter converges faster, but hovers around the optimum solution in a larger region. The size of this region is what the MSD attempts to measure—the EMSE measures how much this hovering affects the error.

These quantities are important in practical situations. For example, in an actual implementation of an acoustic echo canceler, when the near-end user is speaking, his voice is part of $v(n)$. Since the near-end speech is normally stronger than the echo, σ_v^2 tends to be quite large when the near-end speaker is talking. If the adaptive filter were allowed to keep adapting at these moments, (12.208) shows that the filter estimates would wander too far from the optimum. To avoid this problem, the step-size is reduced to zero whenever the near-end user is speaking. A *double-talk detector* is used to decide when to stop adaptation.

It can be shown that (12.206) is stable for sufficiently small μ. However, (12.206) cannot be used to find a range of values of μ that guarantee stability, due to the approximations made, particularly the discarding of term $\boldsymbol{\mathcal{C}}$. Thus, in order to study the stability of LMS, we consider next an approximation for this term, based the assumption that the input vector is Gaussian. This assumption allows us to approximate $\boldsymbol{\mathcal{C}}$ as

$$\boldsymbol{\mathcal{C}} \approx \boldsymbol{R}_{\boldsymbol{\phi}}\mathrm{Tr}(\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}) + \beta\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}, \tag{12.210}$$

where as usual, $\beta = 1$ (resp., $\beta = 2$) for complex (resp., real) data. See Box 7 for the derivation of (12.210).

Replacing (12.210) in (12.205), we arrive at

$$\begin{aligned}\boldsymbol{S}(n+1) \approx \boldsymbol{S}(n) - \mu\left[\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}} + \boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\right]&\\ +\mu^2\left[\boldsymbol{R}_{\boldsymbol{\phi}}\mathrm{Tr}\left(\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}\right) + \beta\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}} + \sigma_v^2\boldsymbol{R}_{\boldsymbol{\phi}}\right]&.\end{aligned} \tag{12.211}$$

The stability of (12.211) is determined through a recursion for the vector $\boldsymbol{s}'(n)$, which contains the elements of the diagonal of the rotated matrix $\boldsymbol{S}'(n) = \boldsymbol{U}^H\boldsymbol{S}(n)\boldsymbol{U}$ (see Section 1.12.4.2). A recursion for $\boldsymbol{S}'(n)$ can be obtained by multiplying (12.211) from the left by $\boldsymbol{U}^H$ and from the right by $\boldsymbol{U}$ and recalling that $\boldsymbol{U}\boldsymbol{U}^H = \boldsymbol{I}$, which leads to

$$\begin{aligned}\boldsymbol{S}'(n+1) \approx \mathbf{S}'(n) - \mu\left[\boldsymbol{S}'(n)\boldsymbol{\Lambda} + \boldsymbol{\Lambda}\boldsymbol{S}'(n)\right]&\\ +\mu^2\left[\boldsymbol{\Lambda}\mathrm{Tr}(\boldsymbol{\Lambda}\boldsymbol{S}'(n)) + \beta\boldsymbol{\Lambda}\boldsymbol{S}'(n)\boldsymbol{\Lambda}\right] + \mu^2\sigma_v^2\boldsymbol{\Lambda}&,\end{aligned} \tag{12.212}$$

where $\boldsymbol{\Lambda}$ is defined as in (12.186). Thus, a recursion for $\boldsymbol{s}'(n+1)$ is given by

$$\boldsymbol{s}'(n+1) = \left[\boldsymbol{I} - 2\mu\boldsymbol{\Lambda} + \mu^2\left(\beta\boldsymbol{\Lambda}^2 + \boldsymbol{l}\boldsymbol{l}^T\right)\right]\boldsymbol{s}'(n) + \mu^2\sigma_v^2\boldsymbol{l}, \tag{12.213}$$

where $\boldsymbol{l}$ is defined in (12.192).

The system matrix $\boldsymbol{A}$ of (12.213) is given by

$$\boldsymbol{A} = \boldsymbol{I} - 2\mu\boldsymbol{\Lambda} + \mu^2\left(\beta\boldsymbol{\Lambda}^2 + \boldsymbol{l}\boldsymbol{l}^T\right) = (\boldsymbol{I} - \mu\boldsymbol{\Lambda})^2 + \mu^2(\beta-1)\boldsymbol{\Lambda}^2 + \mu^2\boldsymbol{l}\boldsymbol{l}^T.$$

Although it is possible to obtain the exact values of all eigenvalues of $\boldsymbol{A}$, as done in [36], we follow here a much simpler, approximate route. We can use Fact 7 in Box 3 to find an upper bound for the largest eigenvalue ν_i of $\boldsymbol{A}$, using the 1-norm of $\boldsymbol{A}$:

$$\max_{0\le i\le M-1} |\nu_i| \le \|\boldsymbol{A}\|_1 = \max_{0\le i\le M-1} \sum_{j=0}^{M-1} |a_{ij}|,$$

where a_{ij} are the entries of $\boldsymbol{A}$. If we find a range of μ for which $\|\boldsymbol{A}\|_1 \le 1$, then the recursion will be stable. This will be of course a conservative condition (sufficient, but not necessary).

Now note that the ith column of $\boldsymbol{A}$ has entries $\lambda_i\lambda_j$ if $i \neq j$, and $(1-\mu\lambda_i)^2+\mu^2(\beta-1)\lambda_i^2+\mu^2\lambda_i^2$ in the term corresponding to the diagonal. Since all terms are positive, the 1-norm is

$$\begin{aligned}&\max_{0\le i\le M-1}\left[(1-\mu\lambda_i)^2+\mu^2(\beta-1)\lambda_i^2+\mu^2\lambda_i\sum_{j=0}^{M-1}\lambda_j\right]\\ &=\max_{0\le i\le M-1}\left[(1-\mu\lambda_i)^2+\mu^2(\beta-1)\lambda_i^2+\mu^2\lambda_i\mathrm{Tr}(\boldsymbol{\Lambda})\right].\end{aligned}$$

Recall that $\mathrm{Tr}(\boldsymbol{\Lambda}) = \mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}})$ (Fact 1 from Box 3). Therefore, the recursion will be stable if

$$1-2\mu\lambda_i+\mu^2\lambda_i^2+\mu^2(\beta-1)\lambda_i^2+\mu^2\lambda_i\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}}) \le 1, \quad 0\le i\le M-1.$$

Simplifying, we obtain the condition

$$\mu \le \frac{2}{\beta\lambda_i+\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}})}, \quad 0\le i\le M-1.$$

The smallest bound is for $\lambda_i = \lambda_{\max}$:

$$\mu \le \frac{2}{\beta\lambda_{\max}+\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}})}. \tag{12.214}$$

If we replace $\lambda_{\max}$ by $\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}}) \ge \lambda_{\max}$ in the denominator, we obtain a simpler, but more conservative condition:

$$\mu \le \frac{2}{(\beta+1)\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}})}. \tag{12.215}$$

For real data, this range reduces to

$$0<\mu<\frac{2}{3\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}})} \quad \text{(real data)} \tag{12.216}$$

and for complex data to

$$0<\mu<\frac{1}{\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}})} \quad \text{(complex data).} \tag{12.217}$$

Table 12.6 Parameters of LMS, NLMS, and RLS Algorithms

Algorithm	$\rho(n)$	$M^{-1}(n)$
LMS	μ	I
NLMS	$\dfrac{\tilde{\mu}}{\varepsilon + \lVert\boldsymbol{\phi}(n)\rVert^2}$	I
RLS	$1-\lambda$	$(1-\lambda)\sum_{\ell=1}^{n}\lambda^{n-\ell}\boldsymbol{\phi}(\ell)\boldsymbol{\phi}^H(\ell)$

If the input is a tapped-delay line as in (12.113), $\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}}) = \sum_{k=0}^{M-1}\mathrm{E}\{|x(n-k)|^2\}$. If $\{x(n)\}$ is a stationary sequence, the stability condition for LMS is easy to obtain: we need to choose μ in the range

$$0 < \mu < \frac{2}{(\beta+1)\times M \times (\text{average power of } x(n))}. \tag{12.218}$$

This concludes our stability analysis for LMS. We now show how to extend these results to NLMS and RLS, in a unified way. At the end of the next section, we give examples comparing the models to simulated results.

1.12.4.4 A unified statistical analysis

In this section, we assume a nonstationary environment, where the optimal solution varies according to the random walk model (see Assumption 2). We start by obtaining a general update equation for the LMS, NLMS, and RLS algorithms, in order to provide a unified statistical analysis for these algorithms.

1.12.4.4.1 A general update equation

In order to provide a unified analysis for the LMS, NLMS and RLS algorithms, we consider the following general update equation

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \rho(n)\boldsymbol{M}(n)\boldsymbol{\phi}(n)e^*(n), \tag{12.219}$$

where $\rho(n)$ is a step-size, $\boldsymbol{M}(n)$ is a non-singular matrix with hermitian symmetry ($\boldsymbol{M}^H(n) = \boldsymbol{M}(n)$), and $e(n) = d(n) - \boldsymbol{w}^H(n)\boldsymbol{\phi}(n)$ is the estimation error. Many adaptive algorithms can be written as in (12.219), by proper choices of $\rho(n)$ and $\boldsymbol{M}(n)$. The LMS, NLMS, and RLS algorithms employ the step-sizes $\rho(n)$ and the matrices $\boldsymbol{M}(n)$ as in Table 12.6, where $\boldsymbol{I}$ is the $M \times M$ identity matrix, μ and $0 < \tilde{\mu} < 2$ are step-sizes, and $0 \ll \lambda < 1$ is a forgetting factor. For RLS, $\boldsymbol{M}(n) = (1-\lambda)^{-1}\boldsymbol{P}(n)$ (see Eq. (12.162)). Note that, since $(1-\lambda)$ plays the role of step-size in RLS, we multiply the matrix $\boldsymbol{P}(n)$ by $(1-\lambda)^{-1}$, keeping the update equation of the RLS weights as in (12.167).

We must rewrite (12.219) in terms of the weight-error vector $\tilde{\boldsymbol{w}}$. Thus, subtracting both sides of (12.219) from $\boldsymbol{w}_o(n+1) = \boldsymbol{w}_o(n) + \boldsymbol{q}(n)$ (Eq. (12.182)), we obtain

$$\begin{aligned}\boldsymbol{w}_o(n+1) - \boldsymbol{w}(n+1) &= \boldsymbol{w}_o(n) + \boldsymbol{q}(n) - \boldsymbol{w}(n) - \rho(n)\boldsymbol{M}(n)\boldsymbol{\phi}(n)e^*(n),\\ \tilde{\boldsymbol{w}}(n+1) &= \tilde{\boldsymbol{w}}(n) - \rho(n)\boldsymbol{M}(n)\boldsymbol{\phi}(n)e^*(n) + \boldsymbol{q}(n).\end{aligned} \tag{12.220}$$

Replacing $e^*(n) = e_a^*(n) + v^*(n) = \boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n) + v^*(n)$ in (12.220), we arrive at

$$\tilde{\boldsymbol{w}}(n+1) = \left[\boldsymbol{I} - \rho(n)\boldsymbol{M}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right]\tilde{\boldsymbol{w}}(n) - \rho(n)\boldsymbol{M}(n)\boldsymbol{\phi}(n)v^*(n) + \boldsymbol{q}(n). \tag{12.221}$$

We next use recursion (12.221) to evaluate the EMSE and the MSD of LMS, NLMS, and RLS.

1.12.4.4.2 Alternative analysis methods

Closed-form expressions for the EMSE and MSD can be obtained in two ways:

i. Finding a recursion for the correlation matrix of the weight-error vector, denoted by $\boldsymbol{S}(n)$. The EMSE and MSE can be easily computed given $\boldsymbol{S}(n)$, as we saw in Section 1.12.4.2. This is the earlier approach in the literature [1,2] and therefore, we refer to it here as *traditional* method.
ii. Using the energy conservation analysis of [3,76–78]. The energy conservation approach relies on an energy conservation relation that holds for a large class of adaptive filters. The energy conservation method presents some advantages when compared to the traditional one. For example, using energy conservation, one can obtain steady-state results directly, bypassing several of the difficulties encountered in obtaining them as a limiting case of a transient analysis. This method can also be used in transient analyses of adaptive algorithms (see, e.g., Part V of [3]).

It is important to understand that both methods involve different sets of approximations (assumptions), so they do not always arrive at the same expressions. Generally, they tend to be equal in the limit of slow adaptation (i.e., if $\rho(n) \to 0$ in (12.221)). For larger values of $\rho(n)$, which method will provide the best approximation depends on the algorithm. As for the work necessary to arrive at the final answers, the energy conservation approach is almost always easier to use if one is interested only in steady-state results. If one is interested in transient performance and stability studies, then the traditional method may be simpler, depending on the algorithm under study.

To give an overview of both methods, we first use the traditional method to obtain the transient and steady-state performance of the LMS, NLMS, and RLS algorithms. This is an extension of what we did for LMS in Section 1.12.4.3, taking advantage of the similarities between the algorithms. In the sequel, we employ the energy conservation method to obtain closed-form expressions for the steady-state EMSE of the same algorithms.

In this section we allow the environment to be nonstationary, i.e., the optimal solution $\boldsymbol{w}_o$ is time-variant. The results for stationary environments can be obtained simply by making $\boldsymbol{Q} = \boldsymbol{0}$ in the equations below. In order to simplify the arguments, we require that the regressor sequence $\{\boldsymbol{\phi}(n)\}$ be wide-sense stationary, that is, we assume that $\boldsymbol{R}_{\boldsymbol{\phi}}$ is constant.

1.12.4.4.3 Analysis with the traditional method

The traditional analysis method consists of finding a recursion for $\boldsymbol{S}(n)$, and using (12.184) and (12.185) to evaluate the MSD and the EMSE, respectively. A recursion for $\boldsymbol{S}(n)$ can be obtained by multiplying (12.221) by its Hermitian, taking the expectations of both sides, and recalling that Assumption 1 implies that $\tilde{\boldsymbol{w}}(n)$ is independent of $\boldsymbol{\phi}(n)$ and of the zero-mean $v_o(n)$. Furthermore, using the fact that

$\mathrm{E}\{\tilde{\boldsymbol{w}}(n)\boldsymbol{q}^H(n)\} = \boldsymbol{0}$ since the sequence $\{\boldsymbol{q}(n)\}$ is iid (Assumption 2), we arrive at

$$\begin{aligned}
\mathrm{E}\{\tilde{\boldsymbol{w}}(n+1)\tilde{\boldsymbol{w}}^H(n+1)\} = \mathrm{E}\{\boldsymbol{q}(n)\boldsymbol{q}^H(n)\} + \mathrm{E}\{\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\} \\
- \overbrace{\mathrm{E}\{\rho(n)\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{M}(n)\}}^{\mathcal{A}} \\
- \overbrace{\mathrm{E}\{\rho(n)\boldsymbol{M}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\}}^{\mathcal{B}} \\
+ \overbrace{\mathrm{E}\{\rho^2(n)\boldsymbol{M}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{M}(n)\}}^{\mathcal{C}} \\
+ \overbrace{\mathrm{E}\{\rho^2(n)|v_o(n)|^2\boldsymbol{M}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{M}(n)\}}^{\mathcal{D}}.
\end{aligned} \tag{12.222}$$

To simplify (12.222), we also need two additional assumptions:

Assumption 3. Matrix $\boldsymbol{M}(n)$ varies slowly in relation to $\tilde{\boldsymbol{w}}(n)$. Thus, when $\boldsymbol{M}(n)$ appears inside the expectations of (12.222), we simply replace it by its mean. For LMS and NLMS, this assumption is not necessary, since $\boldsymbol{M}(n) = \boldsymbol{I}$. For RLS, $\boldsymbol{M}(n) = (1-\lambda)^{-1}\boldsymbol{P}(n) \approx (1-\lambda)^{-1}\mathrm{E}\{\boldsymbol{P}(n)\}$ if $\lambda \approx 1$;

Assumption 4. The input regressor vector $\boldsymbol{\phi}(n)$ is assumed to be Gaussian. This assumption makes the analysis more tractable, since Gaussianity facilitates the computation of the expected values in $\mathcal{C}$.

Now, using the Assumptions 1–4, we can evaluate the terms $\mathcal{A}$-$\mathcal{D}$ of (12.222):

$\mathcal{A}$- The term $\mathcal{A}$ can be approximated by

$$\begin{aligned}
\mathcal{A} &= \mathrm{E}\left\{\mathrm{E}\left\{\rho(n)\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{M}(n)|\boldsymbol{\phi}(n)\right\}\right\} \\
&\approx \mathrm{E}\left\{\rho(n)\mathrm{E}\left\{\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\right\}\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}\mathrm{E}\{\boldsymbol{M}(n)\} \\
&= \boldsymbol{S}(n)\mathrm{E}\left\{\rho(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}\mathrm{E}\{\boldsymbol{M}(n)\}.
\end{aligned} \tag{12.223}$$

For LMS, this term reduces to

$$\mathcal{A}^{\mathrm{LMS}} \approx \mu\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}. \tag{12.224}$$

For NLMS, we must obtain an approximation for

$$\mathrm{E}\left\{\rho(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\} = \mathrm{E}\left\{\frac{\tilde{\mu}\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)}{\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)}\right\}. \tag{12.225}$$

It is possible to evaluate this expected value exactly (see [61,63,64]), but our intention here is to obtain the simplest possible model. Therefore, we will use two additional approximations:

Assumption 5. The number of coefficients M is large enough for each element $\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)$ in the numerator to be approximately independent from the denominator $\sum_{k=0}^{M-1}|\boldsymbol{\phi}(n-k)|^2$.

This is equivalent to apply the averaging principle proposed in [79], since for large M, $\|\boldsymbol{\phi}(n)\|^2$ tends to vary slowly compared to the individual entries of $\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)$.
Thus, (12.225) can be approximated as

$$\mathrm{E}\left\{\frac{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)}{\varepsilon+\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)}\right\} \approx \mathrm{E}\left\{\frac{1}{\varepsilon+\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)}\right\}\mathrm{E}\left\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}. \tag{12.226}$$

Under Assumption 5, the first expected value in (12.226) can by approximated by:

$$\mathrm{E}\left\{\frac{1}{\varepsilon+\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)}\right\} \approx \frac{1}{\mathrm{E}\{\varepsilon+\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)\}} = \frac{1}{\varepsilon+\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}})}. \tag{12.227}$$

An alternative, more accurate approximation is obtained when $\boldsymbol{\phi}(n)$ is Gaussian and a tapped-delay line:

Assumption 6. The regressor $\boldsymbol{\phi}(n)$ is formed by a tapped-delay line with Gaussian entries as in (12.113), and $\varepsilon = 0$.

Under Assumptions 4, 5, and 6 it can be shown (see [65] for details) that

$$\mathrm{E}\left\{\frac{1}{\varepsilon+\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)}\right\} \approx \frac{1}{\sigma_x^2(M-2)}. \tag{12.228}$$

Both approximations are only valid for long filters. However, (12.228) shows that NLMS needs a large regularization constant if $M < 3$, which is verified in practice [63]. We will present the final results for NLMS in terms of Assumption 6, but recall that a less precise, but more general option is available using only Assumption 4 and approximations such as (12.227).
Thus, the term $\boldsymbol{\mathcal{A}}$ for NLMS reduces to (using (12.228))

$$\boldsymbol{\mathcal{A}}^{\mathrm{NLMS}} \approx \frac{\tilde{\mu}}{\sigma_x^2(M-2)}\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}. \tag{12.229}$$

Finally, for RLS, we obtain

$$\boldsymbol{\mathcal{A}}^{\mathrm{RLS}} \approx \boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}\overline{\boldsymbol{P}}(n), \tag{12.230}$$

where $\overline{\boldsymbol{P}}(n) \triangleq \mathrm{E}\{\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}(n)\}$, which we will evaluate later on.
$\boldsymbol{\mathcal{B}}$- Analogously, we obtain for $\boldsymbol{\mathcal{B}}$

$$\boldsymbol{\mathcal{B}} \approx \mathrm{E}\{\boldsymbol{M}(n)\}\,\mathrm{E}\left\{\rho(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}\boldsymbol{S}(n). \tag{12.231}$$

Particularizing for each algorithm, we have

$$\boldsymbol{\mathcal{B}}^{\mathrm{LMS}} \approx \mu\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n), \tag{12.232}$$

$$\boldsymbol{\mathcal{B}}^{\mathrm{NLMS}} \approx \frac{\tilde{\mu}}{\sigma_x^2(M-2)}\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n), \tag{12.233}$$

and

$$\boldsymbol{\mathcal{B}}^{\mathrm{RLS}} \approx \overline{\boldsymbol{P}}(n)\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n). \tag{12.234}$$

$\mathcal{C}$- The term $\mathcal{C}$ can be approximated by

$$\begin{aligned}\mathcal{C} &= \mathrm{E}\left\{\rho^2(n)\boldsymbol{M}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{M}(n)\right\}\\ &\approx \mathrm{E}\left\{\boldsymbol{M}(n)\right\}\mathrm{E}\left\{\rho^2(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{S}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}\mathrm{E}\left\{\boldsymbol{M}(n)\right\}. \qquad (12.235)\end{aligned}$$

Under the Gaussianity assumption 4, it holds that (see Box 7)

$$\mathrm{E}\left\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{S}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\} = \boldsymbol{R}_{\boldsymbol{\phi}}\mathrm{Tr}(\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}) + \beta\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}, \qquad (12.236)$$

where $\beta = 1$ (resp., $\beta = 2$) for complex (resp. real) data. Thus, for LMS we have

$$\mathcal{C}^{\mathrm{LMS}} \approx \mu^2\left[\boldsymbol{R}_{\boldsymbol{\phi}}\mathrm{Tr}(\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}) + \beta\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}\right]. \qquad (12.237)$$

For NLMS, using the same arguments to get (12.226), the following approximation holds for large M:

$$\begin{aligned}&\mathrm{E}\left\{\rho^2(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{S}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}\\ &\approx \tilde{\mu}^2\mathrm{E}\left\{\frac{1}{\left[\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)\right]^2}\right\}\mathrm{E}\left\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{S}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}. \qquad (12.238)\end{aligned}$$

Under Assumption 5, the first expectation on the r.h.s. of (12.238) is given by

$$\mathrm{E}\left\{\frac{1}{\left[\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)\right]^2}\right\} = \frac{1}{\varepsilon + \mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}}) + \beta\mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}}^2)}. \qquad (12.239)$$

On the other hand, under Assumption 6, we obtain [65]

$$\mathrm{E}\left\{\frac{1}{\left[\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)\right]^2}\right\} = \frac{1}{\sigma_x^4(M-2)(M-4)}. \qquad (12.240)$$

Replacing (12.240) and (12.236) in (12.238), we arrive at

$$\mathcal{C}^{\mathrm{NLMS}} \approx \frac{\tilde{\mu}^2}{\sigma_x^4(M-2)(M-4)}\left[\boldsymbol{R}_{\boldsymbol{\phi}}\ \mathrm{Tr}(\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}) + \beta\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}\right]. \qquad (12.241)$$

Finally, for RLS we obtain

$$\mathcal{C}^{\mathrm{RLS}} \approx \overline{\boldsymbol{P}}(n)\left[\boldsymbol{R}_{\boldsymbol{\phi}}\ \mathrm{Tr}(\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}) + \beta\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}\right]\overline{\boldsymbol{P}}(n). \qquad (12.242)$$

$\mathcal{D}$- Using the fact that $v_{\mathrm{o}}(n)$ is independent of $\boldsymbol{\phi}(n)$ and $\tilde{\boldsymbol{w}}(n)$, $\mathcal{D}$ reduces to

$$\begin{aligned}\mathcal{D} &= \mathrm{E}\{\rho^2(n)|v_{\mathrm{o}}(n)|^2\boldsymbol{M}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{M}(n)\}\\ &\approx \sigma_v^2\mathrm{E}\{\boldsymbol{M}(n)\}\mathrm{E}\{\rho^2(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\}\mathrm{E}\{\boldsymbol{M}(n)\}. \qquad (12.243)\end{aligned}$$

Particularizing for each algorithm, we obtain

$$\mathcal{D}^{\text{LMS}} \approx \mu^2 \sigma_v^2 \boldsymbol{R}_{\boldsymbol{\phi}}(n), \tag{12.244}$$

$$\mathcal{D}^{\text{NLMS}} \approx \frac{\tilde{\mu}^2}{\sigma_x^4 (M-2)(M-4)} \sigma_v^2 \boldsymbol{R}_{\boldsymbol{\phi}}(n), \tag{12.245}$$

where we could also have used (12.239), and

$$\mathcal{D}^{\text{RLS}} \approx \sigma_v^2 \overline{\boldsymbol{P}}(n) \boldsymbol{R}_{\boldsymbol{\phi}} \overline{\boldsymbol{P}}(n). \tag{12.246}$$

From the previous results, (12.222) reduces to

$$\begin{aligned}\mathbf{S}(n+1) \approx \mathbf{S}(n) &- \mathrm{E}\{\rho(n)\} \left[\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}} \mathrm{E}\{\boldsymbol{M}(n)\} + \mathrm{E}\{\boldsymbol{M}(n)\}\boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\right] \\ &+ \mathrm{E}\{\rho^2(n)\}\mathrm{E}\{\boldsymbol{M}(n)\} \left[\boldsymbol{R}_{\boldsymbol{\phi}} \operatorname{Tr}(\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}) + \beta \boldsymbol{R}_{\boldsymbol{\phi}}\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}}\right] \mathrm{E}\{\boldsymbol{M}(n)\} \\ &+ \sigma_v^2 \mathrm{E}\{\rho^2(n)\}\mathrm{E}\{\boldsymbol{M}(n)\}\boldsymbol{R}_{\boldsymbol{\phi}}\mathrm{E}\{\boldsymbol{M}(n)\} + \boldsymbol{Q}\end{aligned} \tag{12.247}$$

with initialization $\boldsymbol{S}(0) = \left[\boldsymbol{w}_{\mathrm{o}}(0) - \boldsymbol{w}(0)\right]\left[\boldsymbol{w}_{\mathrm{o}}(0) - \boldsymbol{w}(0)\right]^H$.

For RLS, we still need an approximation for $\overline{\boldsymbol{P}}(n) = \mathrm{E}\{\boldsymbol{P}(n)\}$. Initializing the estimate of the autocorrelation matriz with $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(-1) = \delta \boldsymbol{I}$, we will have at iteration n

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n) = \lambda^n \delta \boldsymbol{I} + \sum_{\ell=1}^{n} \lambda^{n-\ell} \boldsymbol{\phi}(\ell)\boldsymbol{\phi}^H(\ell). \tag{12.248}$$

Taking the expectations of both sides of (12.248), we obtain

$$\mathrm{E}\left\{\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\right\} = \lambda^n \delta \boldsymbol{I} + \boldsymbol{R}_{\boldsymbol{\phi}} \frac{1-\lambda^n}{1-\lambda}. \tag{12.249}$$

For $\lambda \approx 1$, the variance of $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)$ is small, and we can approximate $\overline{\boldsymbol{P}}(n) \approx \left[\mathrm{E}\left\{\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\right\}\right]^{-1}$, so that

$$\overline{\boldsymbol{P}}(n) \approx \left[\lambda^n \delta \boldsymbol{I} + \boldsymbol{R}_{\boldsymbol{\phi}} \frac{1-\lambda^n}{1-\lambda}\right]^{-1}. \tag{12.250}$$

This approximation is good for large n and in steady-state. During the initial phases of the transient, however, the approximation will only be reasonable if δ is large enough to make the last term in (12.249) small compared to $\lambda^n \delta \boldsymbol{I}$. This is because for $n < M$, the term $\sum_{\ell=1}^{n} \lambda^{n-\ell} \boldsymbol{\phi}(\ell)\boldsymbol{\phi}^H(\ell)$ in (12.248) will be singular, and thus will not be well approximated by $(1-\lambda^n)\boldsymbol{R}_{\boldsymbol{\phi}}/(1-\lambda)$.

Using the previous approximations, (12.247) can be particularized for the LMS, NLMS, and RLS algorithms as shown in Table 12.7. Since we can obtain the EMSE and the MSD from $\boldsymbol{S}(n)$ and $\boldsymbol{R}_{\boldsymbol{\phi}}$ through (12.185) and (12.184), we should use the recursions of Table 12.7 to compute $\boldsymbol{S}(n)$ and then evaluate $\zeta(n) = \operatorname{Tr}(\boldsymbol{S}(n)\boldsymbol{R}_{\boldsymbol{\phi}})$ and $\chi(n) = \operatorname{Tr}(\boldsymbol{S}(n))$.

Example 3. To verify the accuracy of the expressions of Table 12.7, we consider a system identification application in a stationary environment ($\boldsymbol{Q} = \mathbf{0}$). The optimal solution is a lowpass FIR filter with linear

Table 12.7 Recursions for Covariance Matrix $\boldsymbol{S}(n+1)$. The Expressions for NLMS Assume that the Regressor is a Tapped-Delay Line. Alternative Expressions for NLMS are Available Using (12.227) and (12.239)

Algorithm	$\boldsymbol{S}(n+1)$
LMS	$\boldsymbol{S}(n+1) \approx \boldsymbol{S}(n) - \mu\left[\boldsymbol{S}(n)\boldsymbol{R}_{\phi} + \boldsymbol{R}_{\phi}\boldsymbol{S}(n)\right] + \mu^2\left[\boldsymbol{R}_{\phi}\mathrm{Tr}\left(\boldsymbol{S}(n)\boldsymbol{R}_{\phi}\right) + \beta\boldsymbol{R}_{\phi}\boldsymbol{S}(n)\boldsymbol{R}_{\phi} + \sigma_v^2\boldsymbol{R}_{\phi}\right] + \mathbf{Q}$
NLMS	$\boldsymbol{S}(n+1) \approx \boldsymbol{S}(n) - \dfrac{\tilde{\mu}}{\sigma_x^2(M-2)}\left[\boldsymbol{S}(n)\boldsymbol{R}_{\phi} + \boldsymbol{R}_{\phi}\boldsymbol{S}(n)\right] + \dfrac{\tilde{\mu}^2}{\sigma_x^4(M-2)(M-4)}\left[\boldsymbol{R}_{\phi}\mathrm{Tr}\left(\boldsymbol{S}(n)\boldsymbol{R}_{\phi}\right) + \beta\boldsymbol{R}_{\phi}\boldsymbol{S}(n)\boldsymbol{R}_{\phi} + \sigma_v^2\boldsymbol{R}_{\phi}\right] + \mathbf{Q}$
RLS	$\boldsymbol{S}(n+1) \approx \boldsymbol{S}(n) - \left[\boldsymbol{S}(n)\boldsymbol{R}_{\phi}\overline{\boldsymbol{P}}(n) + \overline{\boldsymbol{P}}(n)\boldsymbol{R}_{\phi}\boldsymbol{S}(n)\right] + \overline{\boldsymbol{P}}(n)\left[\boldsymbol{R}_{\phi}\mathrm{Tr}\left(\boldsymbol{S}(n)\boldsymbol{R}_{\phi}\right) + \beta\boldsymbol{R}_{\phi}\boldsymbol{S}(n)\boldsymbol{R}_{\phi} + \sigma_v^2\boldsymbol{R}_{\phi}\right]\overline{\boldsymbol{P}}(n) + \mathbf{Q}$

phase, whose coefficients are shown in Figure 12.29. The regressor $\boldsymbol{\phi}(n)$ is obtained from a process $x(n)$ generated with a first-order autoregressive model, whose transfer function is $\sqrt{1-b^2}/(1-bz^{-1})$. Note that the sequence $\{\boldsymbol{\phi}(n)\}$ is not iid. This model is fed with an iid Gaussian random process, whose variance is such that $\mathrm{Tr}(\boldsymbol{R}_{\phi}) = 1$. Moreover, additive iid noise $v_o(n)$ with variance $\sigma_v^2 = 0.01$ is added to form the desired signal.

Figure 12.30 shows the EMSE for RLS ($\lambda = 0.995, \delta = 1$), NLMS ($\tilde{\mu} = 0.1, \varepsilon = 10^{-5}$), and LMS ($\mu = 0.05$), estimated from the ensemble-average of 2000 independent runs. The dashed lines represent the theoretical EMSE computed as $\zeta(n) = \mathrm{Tr}(\boldsymbol{S}(n)\boldsymbol{R}_{\phi})$. We can observe a good agreement between theory and simulation.

Figure 12.31 shows the EMSE of RLS ($\lambda = 0.995$), assuming different values for δ. We can observe that the theoretical EMSE of RLS presents a good agreement with simulation during the initial phases of the transient, but only for large values of δ due to the approximation (12.250).

1.12.4.4.4 *Steady-state analysis with the energy conservation method*

This method (also known as *feedback* approach [78,80]) is based on an energy conservation relation that holds for a large class of adaptive filters. To obtain this relation for the algorithms of the form (12.219), we first assume a stationary environment ($\boldsymbol{q}(n) \equiv \mathbf{0}$).

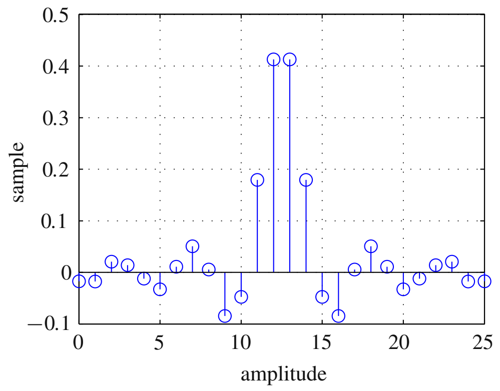


FIGURE 12.29

Impulse response $\boldsymbol{w}_o$ of the unknown lowpass filter.

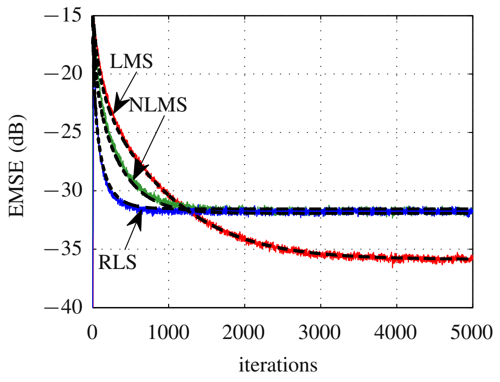


FIGURE 12.30

EMSE for RLS ($\lambda = 0.995$, $\delta = 1$), NLMS ($\tilde{\mu} = 0.1$, $\varepsilon = 10^{-5}$), and LMS ($\mu = 0.05$); $b = 0.8$, $\sigma_v^2 = 10^{-2}$, $\sigma_q^2 = 0$; mean of 2000 independent runs. The dashed lines represent the predicted values of $\zeta(n)$ for each algorithm.

Recall that the excess a posteriori error is given by

$$e_p(n) = \tilde{\boldsymbol{w}}^H(n+1)\boldsymbol{\phi}(n). \tag{12.251}$$

Then, we multiply both sides of the Hermitian of (12.220) (with $\boldsymbol{q}(n) \equiv \boldsymbol{0}$) from the right by $\boldsymbol{\phi}(n)$ to obtain

$$\tilde{\boldsymbol{w}}^H(n+1)\boldsymbol{\phi}(n) = \tilde{\boldsymbol{w}}^H(n)\boldsymbol{\phi}(n) - \rho(n)\boldsymbol{\phi}^H(n)\boldsymbol{M}(n)\boldsymbol{\phi}(n)e(n). \tag{12.252}$$

Using (12.172) and (12.251), (12.252) reduces to

$$e_p(n) = e_a(n) - \rho(n)\|\boldsymbol{\phi}(n)\|^2_{\boldsymbol{M}(n)}e(n), \tag{12.253}$$

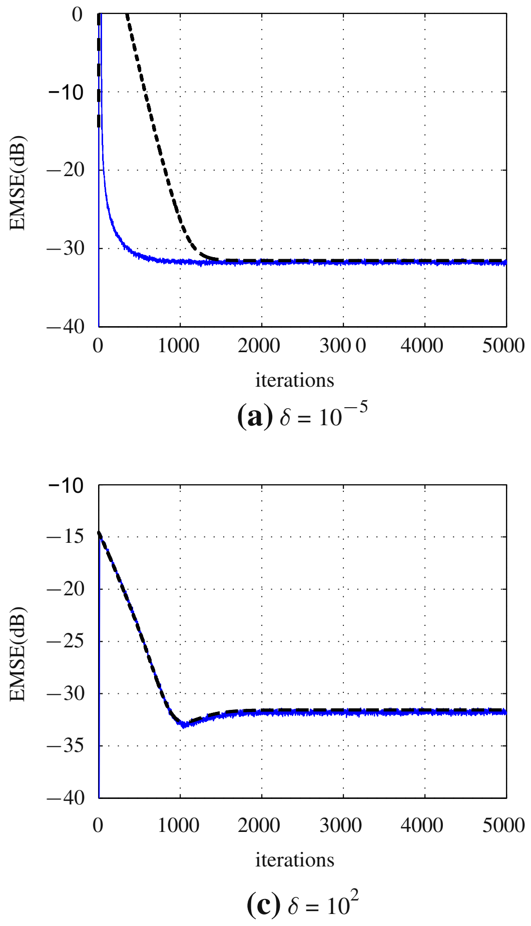


FIGURE 12.31

EMSE for RLS considering $\lambda = 0.995$ and different values of δ; $b = 0.8$, $\sigma_v^2 = 10^{-2}$, $\sigma_q^2 = 0$; mean of 2000 independent runs. The dashed lines represent the predicted values of $\zeta(n)$ for each algorithm.

where $\|\boldsymbol{u}\|_{\boldsymbol{A}}^2 = \boldsymbol{u}^H \boldsymbol{A}\boldsymbol{u}$ stands for the Euclidean norm of $\boldsymbol{u}$ weighted by the matrix $\boldsymbol{A}$. Using (12.253) to eliminate $e(n)$ in (12.220) (with $\boldsymbol{q}(n) \equiv \boldsymbol{0}$) and assuming that $\boldsymbol{\phi}(n) \neq \boldsymbol{0}$, we get

$$\tilde{\boldsymbol{w}}(n+1) + \frac{\boldsymbol{M}(n)\boldsymbol{\phi}(n)}{\|\boldsymbol{\phi}(n)\|_{\boldsymbol{M}(n)}^2} e_a^*(n) = \tilde{\boldsymbol{w}}(n) + \frac{\boldsymbol{M}(n)\boldsymbol{\phi}(n)}{\|\boldsymbol{\phi}(n)\|_{\boldsymbol{M}(n)}^2} e_p^*(n). \tag{12.254}$$

Defining

$$\bar{\rho}(n) \triangleq \begin{cases} 1/\|\boldsymbol{\phi}(n)\|_{\boldsymbol{M}(n)}^2, & \text{if } \boldsymbol{\phi}(n) \neq \boldsymbol{0} \\ 0, & \text{otherwise} \end{cases} \tag{12.255}$$

to include the case of zero regressor and equating the squared weighted norms on both sides of (12.254) with $\boldsymbol{M}^{-1}(n)$ as a weighting matrix, the cross-terms cancel and we obtain

$$\|\tilde{\boldsymbol{w}}(n+1)\|^2_{\boldsymbol{M}^{-1}(n)} + \bar{\rho}(n)|e_a(n)|^2 = \|\tilde{\boldsymbol{w}}(n)\|^2_{\boldsymbol{M}^{-1}(n)} + \bar{\rho}(n)|e_p(n)|^2. \tag{12.256}$$

This relation shows how the weighted energies of the weight-error vectors at two successive time instants are related to the energies of the excess *a priori* and excess *a posteriori* errors. This relation is the reason why this approach is known as the energy conservation method—(12.256) relates the (weighted) coefficient error energies at instants n and $n+1$ to the *a priori* and *a posteriori* errors. In [3], this energy conservation relation is used extensively to find models for a wide class of adaptive filters. We follow here a shorter route, however, and derive in a more direct way the relations necessary for the study of just LMS, NLMS and RLS.

We can obtain the steady-state EMSE by computing the squared weighted norms on both sides of (12.220) with $\boldsymbol{M}^{-1}(n)$ as a weighting matrix, which leads to

$$\begin{aligned}\|\tilde{\boldsymbol{w}}(n+1)\|^2_{\boldsymbol{M}^{-1}(n)} = {}& \|\tilde{\boldsymbol{w}}(n)\|^2_{\boldsymbol{M}^{-1}(n)} + \rho^2(n)|e(n)|^2\|\boldsymbol{\phi}(n)\|^2_{\boldsymbol{M}(n)} \\ & -\rho(n)\left[e_a(n)e^*(n) + e_a^*(n)e(n)\right] + \|\boldsymbol{q}(n)\|_{\boldsymbol{M}^{-1}(n)}.\end{aligned} \tag{12.257}$$

By taking expectations on both sides of (12.257), we arrive at (we already used Assumption 2 to eliminate cross-terms with $\boldsymbol{q}(n)$ on the right-hand side)

$$\begin{aligned}\overbrace{\mathrm{E}\left\{\|\tilde{\boldsymbol{w}}(n+1)\|^2_{\boldsymbol{M}^{-1}(n)}\right\}}^{\mathcal{A}} = {}& \overbrace{\mathrm{E}\left\{\|\tilde{\boldsymbol{w}}(n)\|^2_{\boldsymbol{M}^{-1}(n)}\right\}}^{\mathcal{B}} \\ & + \overbrace{\mathrm{E}\left\{\rho^2(n)|e(n)|^2\|\boldsymbol{\phi}(n)\|^2_{\boldsymbol{M}(n)}\right\}}^{\mathcal{C}} \\ & - \overbrace{\mathrm{E}\left\{\rho(n)\left[e_a(n)e^*(n) + e_a^*(n)e(n)\right]\right\}}^{\mathcal{D}} \\ & + \overbrace{\|\boldsymbol{q}(n)\|_{\boldsymbol{M}^{-1}(n)}}^{\mathcal{E}}.\end{aligned} \tag{12.258}$$

To proceed, we have to assume again that matrix $\boldsymbol{M}^{-1}(n)$ varies slowly in relation to $\tilde{\boldsymbol{w}}(n)$ and $\boldsymbol{q}(n)$, as in Assumption 3. Since now we are dealing only with the steady-state, this approximation is in general good. Instead of Assumption 1, in energy conservation analysis the following assumptions are used:

Assumption 7. $\|\boldsymbol{\phi}(n)\|^2_{\boldsymbol{M}(n)}$ is independent of $e_a(n)$ at the steady-state.

Assumption 8. The noise sequence $\{v_o(n)\}$ is iid and independent of the input sequence $\{\boldsymbol{\phi}(n)\}$.

Assumption 7 is referred to in the literature as *separation principle* and is reasonable in steady-state since for large n, $e_a(n)$ tends to be less sensitive to the regressor data, in particular for long filters. Assumption 8 is a weakened version of Assumption 1, which is necessary to relate the MSE with the EMSE through (12.181).

Under model 2 and Assumptions 3 and 7, we can now evaluate the terms $\mathcal{A} - \mathcal{E}$:

$\mathcal{A}$- Using 3, we obtain

$$\mathcal{A} \approx \mathrm{E}\left\{\|\tilde{\boldsymbol{w}}(n+1)\|^2_{\mathrm{E}\{\boldsymbol{M}^{-1}(n)\}}\right\}. \tag{12.259}$$

For LMS and NLMS, we have

$$\mathcal{A}^{\mathrm{LMS}} = \mathcal{A}^{\mathrm{NLMS}} \approx \mathrm{E}\left\{\|\tilde{\boldsymbol{w}}(n+1)\|^2\right\}. \tag{12.260}$$

For RLS,

$$\mathcal{A}^{\mathrm{RLS}} \approx \mathrm{E}\left\{\|\tilde{\boldsymbol{w}}(n+1)\|^2_{\boldsymbol{R}_{\boldsymbol{\phi}}}\right\}. \tag{12.261}$$

$\mathcal{B}$- Assuming that the filter is stable and reaches a steady-state, we have

$$\mathcal{B} = \mathrm{E}\left\{\|\tilde{\boldsymbol{w}}(n)\|^2_{\boldsymbol{M}^{-1}(n)}\right\} \approx \mathrm{E}\left\{\|\tilde{\boldsymbol{w}}(n+1)\|^2_{\boldsymbol{M}^{-1}(n)}\right\}. \tag{12.262}$$

$\mathcal{C}$- Given that $e(n) = e_a(n) + v_o(n)$ and since $v_o(n)$ is assumed independent of $e_a(n)$ (Assumption 8), we obtain

$$\mathrm{E}\{|e(n)|^2\} \approx \mathrm{E}\{|e_a(n)|^2\} + \sigma_v^2.$$

Then, using Assumptions 3 and 8, we arrive at

$$\mathcal{C} \approx \mathrm{E}\left\{\rho^2(n)\boldsymbol{\phi}^H(n)\mathrm{E}\{\boldsymbol{M}(n)\}\boldsymbol{\phi}(n)\right\}\left(\mathrm{E}\{|e_a(n)|^2\} + \sigma_v^2\right). \tag{12.263}$$

For LMS, this term reduces to

$$\mathcal{C}^{\mathrm{LMS}} \approx \mu^2 \mathrm{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}})\left(\mathrm{E}\{|e_a(n)|^2\} + \sigma_v^2\right). \tag{12.264}$$

For NLMS,

$$\mathcal{C}^{\mathrm{NLMS}} \approx \tilde{\mu}^2 \mathrm{E}\left\{\frac{1}{\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)}\right\}\left(\mathrm{E}\{|e_a(n)|^2\} + \sigma_v^2\right), \tag{12.265}$$

which for long filters and Gaussian tapped-delay line regressors can be approximated as (see Eq. (12.228), and the alternative (12.227) for general regressors)

$$\mathcal{C}^{\mathrm{NLMS}} \approx \frac{\tilde{\mu}^2}{\sigma_x^2(M-2)}\left(\mathrm{E}\{|e_a(n)|^2\} + \sigma_v^2\right). \tag{12.266}$$

Finally, for RLS we obtain

$$\mathcal{C}^{\mathrm{RLS}} \approx (1-\lambda)^2 M\left(\mathrm{E}\{|e_a(n)|^2\} + \sigma_v^2\right). \tag{12.267}$$

$\mathcal{D}$- Using the same arguments, $\mathcal{D}$ reduces to

$$\mathcal{D} \approx 2\mathrm{E}\{\rho(n)\}\mathrm{E}\{|e_a(n)|^2\}. \tag{12.268}$$

For LMS, $\mathcal{D}^{\mathrm{LMS}} \approx 2\mu\mathrm{E}\{|e_a(n)|^2\}$. Using (Eq. (12.228)) for NLMS, we get

$$\mathcal{D}^{\mathrm{NLMS}} \approx \frac{2\tilde{\mu}^2}{\sigma_x^2(M-2)}\mathrm{E}\{|e_a(n)|^2\}.$$

For RLS, this term is given by $\mathcal{D}^{\mathrm{RLS}} \approx 2(1-\lambda)\mathrm{E}\{|e_a(n)|^2\}$.

Table 12.8 Expressions for the Steady-State EMSE Obtained from the Energy Conservation Method for LMS, NLMS, and RLS

Algorithm	$\zeta(\infty)$	$\zeta(\infty)$
	for a wider range of μ, $\tilde{\mu}$, and λ	for small μ and $\tilde{\mu}$, and $\lambda \approx 1$
LMS	$\dfrac{\mu\sigma_v^2\text{Tr}(\boldsymbol{R_\phi})+\mu^{-1}\text{Tr}(\boldsymbol{Q})}{2-\mu\text{Tr}(\boldsymbol{R_\phi})}$	$\dfrac{\mu\sigma_v^2\text{Tr}(\boldsymbol{R_\phi})+\mu^{-1}\text{Tr}(\boldsymbol{Q})}{2}$
NLMS	$\dfrac{\tilde{\mu}\sigma_v^2+\tilde{\mu}^{-1}\sigma_x^2(M-2)\text{Tr}(\boldsymbol{Q})}{2-\tilde{\mu}}$	$\dfrac{\tilde{\mu}\sigma_v^2+\tilde{\mu}^{-1}\sigma_x^2(M-2)\text{Tr}(\boldsymbol{Q})}{2}$
RLS	$\dfrac{\sigma_v^2(1-\lambda)M+\dfrac{\text{Tr}(\boldsymbol{QR_\phi})}{1-\lambda}}{2-(1-\lambda)M}$	$\dfrac{\sigma_v^2(1-\lambda)M+\dfrac{\text{Tr}(\boldsymbol{QR_\phi})}{1-\lambda}}{2}$

$\mathcal{E}$- Using Assumption 3 and recalling that $\text{Tr}(\boldsymbol{AB})=\text{Tr}(\boldsymbol{BA})$, we obtain

$$\mathcal{E}\approx \text{E}\left\{\boldsymbol{q}^H(n)\text{E}\{\boldsymbol{M}^{-1}(n)\}\boldsymbol{q}(n)\right\}=\text{Tr}\left(\text{E}\left\{\boldsymbol{M}^{-1}(n)\right\}\text{E}\{\boldsymbol{q}(n)\boldsymbol{q}^H(n)\}\right).$$

For LMS and NLMS this reduces to

$$\mathcal{E}^{\text{LMS}}=\mathcal{E}^{\text{NLMS}}\approx\text{Tr}(\boldsymbol{Q}). \tag{12.269}$$

For RLS,

$$\mathcal{E}^{\text{RLS}}\approx\text{Tr}(\boldsymbol{R_\phi Q}). \tag{12.270}$$

Using these approximations in (12.258), we arrive at the following steady-state approximation

$$\text{E}\{|e_a(n)|^2\}\approx\frac{\sigma_v^2\text{E}\left\{\rho^2(n)\boldsymbol{\phi}^H(n)\text{E}\{\boldsymbol{M}(n)\}\boldsymbol{\phi}(n)\right\}+\text{Tr}(\boldsymbol{Q}\text{E}\{\boldsymbol{M}^{-1}(n)\})}{2\text{E}\{\rho(n)\}-\text{E}\left\{\rho^2(n)\boldsymbol{\phi}^H(n)\text{E}\{\boldsymbol{M}(n)\}\boldsymbol{\phi}(n)\right\}}. \tag{12.271}$$

Particularizing (12.271) for LMS, NLMS, and RLS, we obtain the results of the second column of Table 12.8, which hold over a wide range of step-sizes and forgetting factors.

1.12.4.4.5 *Relation between the results obtained with both analysis methods*

As we explained before, each method is based on a different set of approximations. We now show how they are related.

The energy conservation method is capable of obtaining closed-form expressions for the EMSE using less restrictive assumptions, in particular $\boldsymbol{\phi}(n)$ is not required to be Gaussian. These results are equivalent to those obtained for LMS and RLS using the traditional analysis, taking the recursions of Table 12.7 to the limit as $n\to\infty$, and assuming $\beta=0$ (recall that the values of β used for the traditional analysis are only valid if $\boldsymbol{\phi}(n)$ is Gaussian). The assumption of $\beta=0$ implies the following approximation:

$$\text{E}\left\{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{S}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\right\}\approx\boldsymbol{R_\phi}\text{Tr}(\boldsymbol{S}(n)\boldsymbol{R_\phi}).$$

Table 12.9 Expressions for the Optimal Adaptation Parameters (ρ_o) and Minimum Steady-State EMSE ($\zeta_{\min}(\infty)$) Obtained from the Expressions of the Third Column of Table 12.8

Algorithm	ρ_o	$\zeta_{\min}(\infty)$
LMS	$\mu_o = \sqrt{\dfrac{\mathrm{Tr}(\boldsymbol{Q})}{\sigma_v^2 \mathrm{Tr}(\boldsymbol{R_\phi})}}$	$\sqrt{\sigma_v^2 \mathrm{Tr}(\boldsymbol{R_\phi})\mathrm{Tr}(\boldsymbol{Q})}$
NLMS	$\tilde{\mu}_o = \sqrt{\dfrac{(M-2)\sigma_x^2 \mathrm{Tr}(\boldsymbol{Q})}{\sigma_v^2}}$	$\sqrt{\sigma_v^2\sigma_x^2(M-2)\mathrm{Tr}(\boldsymbol{Q})}$
RLS	$1-\lambda_o = \sqrt{\dfrac{\mathrm{Tr}(\boldsymbol{QR_\phi})}{M\sigma_v^2}}$	$\sqrt{\sigma_v^2 M \mathrm{Tr}(\boldsymbol{QR_\phi})}$

For small step-sizes, this approximation does not have a large impact, since we are disregarding a term of $\mathcal{O}(\mu^2)$, that is small compared to other $\mathcal{O}(\mu)$ terms. For NLMS, in order to recover (12.271) using the traditional method, we would choose $\beta = 0$ and make

$$\mathrm{E}\left\{\frac{1}{\left[\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)\right]^2}\right\} \approx \left[\mathrm{E}\left\{\frac{1}{\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)}\right\}\right]^2,$$

which leads to reasonable results at the steady-state for long filters.

For small step-sizes (or forgetting factors close to one), the denominator of (12.271) can be approximated by $2\mathrm{E}\{\rho(n)\}$. In this case, the expressions of the second column reduce to those of the third column of Table 12.8. Again, these results can be obtained from the recursions of Table 12.7, disregarding the term $\boldsymbol{\mathcal{C}}$ (Eq. (12.235)), which will be negligible compared to $\mathcal{B}$ in (12.222) if $\rho \approx 0$.

The results for steady-state analysis using the traditional method are commonly obtained as the limiting case of a transient analysis. Although this procedure is adequate for understanding both the steady-state and the transient behavior of an adaptive algorithm, it tends to be more laborious than using the the energy conservation method to analyze the steady-state behavior. This is one of the largest advantages of this last method.

1.12.4.4.6 *Optimal step-size for tracking*

From the results of the third column of Table 12.8, we can observe that the expressions for the steady-state EMSE have two terms: one for the stationary environment, that increases as the step-size increases, and another that appears in the nonstationary case, and increases as the step-size decreases. Therefore, there exist optimal adaptation parameters ρ_o, that minimize the EMSE. These parameters are μ_o, $\tilde{\mu}_o$, and $1-\lambda_o$ for LMS, NLMS, and RLS, respectively. The corresponding minima, denoted by $\zeta_{\min}(\infty)$, can also be evaluated. All these results are summarized in Table 12.9. The results for a wider range of adaptation parameters lead to more complicated expressions for ρ_o and $\zeta_o(\infty)$ (see, e.g., [3]).

We can use the results of Table 12.9 to compare the tracking performance of the algorithms. For instance, comparing the minimum EMSE for LMS to that of RLS, we obtain the ratio

$$\frac{\zeta_{\min}^{\mathrm{RLS}}}{\zeta_{\min}^{\mathrm{LMS}}} = \sqrt{\frac{M\mathrm{Tr}(\boldsymbol{QR_\phi})}{\mathrm{Tr}(\boldsymbol{R_\phi})\mathrm{Tr}(\boldsymbol{Q})}}. \qquad (12.272)$$

Clearly, the results of this comparison depend on the environment. There are situations where RLS has superior tracking capability compared to LMS, and vice versa. This is highlighted considering three different choices for matrix $\boldsymbol{Q}$. It can be shown [81] that

i. If $\boldsymbol{Q}$ is a multiple of $\boldsymbol{I}$: the performance of LMS is similar to that of RLS;
ii. If $\boldsymbol{Q}$ is a multiple of $\boldsymbol{R}_{\boldsymbol{\phi}}$: LMS is superior; and
iii. If $\boldsymbol{Q}$ is a multiple of $\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}$: RLS is superior.

These choices for $\boldsymbol{Q}$ do not model practical situations, but they highlight that, even though the convergence rate of RLS is in general much higher than that of LMS, this advantage, unexpectedly, does not necessarily follow to the problem of tracking a slowly-varying parameter vector.

Example 4. To verify the accuracy of the expressions of Tables 12.8 and 12.9, we use the LMS, NLMS, and RLS filters to identify the same lowpass FIR system of Figure 12.29. Now, the environment is assumed nonstationary with $\boldsymbol{Q} = 10^{-6}\boldsymbol{I}$. The input signal and minimum MSE (σ_v^2) are the same of Example 3.

Figure 12.32 shows the measured steady-state EMSE with varying adaptation factors considering the theoretical and experimental results for LMS, NLMS, and RLS. Each value of experimental EMSE was obtained from the ensemble-average of 100 independent runs. The theoretical minimum EMSE predicted by the expressions of Table 12.9 are $\zeta_{\rm o}^{\rm LMS}(\infty) = \zeta_{\rm o}^{\rm RLS}(\infty) = -32.9251$ dB and $\zeta_{\rm o}^{\rm NLMS}(\infty) = -33.0989$ dB, which correspond to the optimal adaptation parameters $\mu_{\rm o} = 5.1 \times 10^{-2}$, $\tilde{\mu}_{\rm o} = 4.9 \times 10^{-2}$ and $1 - \lambda_{\rm o} = 2 \times 10^{-3}$ ($\lambda_{\rm o} = 0.998$). The experimental values for $\zeta_{\min}(\infty)$ and $\rho_{\rm o}$ are close to the predicted by the expressions of Table 12.9 as shown in Figure 12.32.

Example 5. Again, we consider a system identification application but with the initial optimal solution given by

$$\boldsymbol{w}_{\rm o}^T(0) = [\,0.5349\ 0.9527\ -0.9620\ -0.0158\ -0.1254\,].$$

The input signal is assumed to be colored Gaussian noise with variance $\sigma_x^2 = 0.2$. This signal is obtained from the filtering of a white Gaussian noise by a first-order autoregressive model ($b = 0.8$). We also assume that the minimum MSE is $\sigma_v^2 = 0.01$.

As predicted by (12.272), the performance of RLS and LMS are similar when $\boldsymbol{Q} = \sigma_q^2\boldsymbol{I}$. Thus, assuming $\boldsymbol{Q} = 10^{-6}\boldsymbol{I}$, we computed $\mu_{\rm o}$ and $\lambda_{\rm o}$ from the expressions of Table 12.9 and used these parameters in the adaptation, such that the same minimum EMSE could be achieved by both algorithms in this situation. Figure 12.33 shows the EMSE estimated from the ensemble-average of 5000 independent runs for RLS ($\lambda = \lambda_{\rm o} = 0.9955$, $\delta = 4.5 \times 10^{-3}$) and LMS ($\mu = \mu_{\rm o} = 0.0224$). At every 7×10^4 iterations, the nonstationary environment is changed. During the first 7×10^4 iterations, $\boldsymbol{Q} = 10^{-6}\boldsymbol{R}_{\boldsymbol{\phi}}$ and LMS presents a slightly better tracking performance than that of RLS. When $\boldsymbol{Q} = 10^{-6}\boldsymbol{R}_{\boldsymbol{\phi}}^{-1}$, this behavior changes: RLS becomes better than LMS. Finally for $\boldsymbol{Q} = 10^{-6}\boldsymbol{I}$, RLS and LMS present similar performance. The dashed lines represent the theoretical values of $\zeta(\infty)$ for each algorithm, predicted from the expressions of the second column of Table 12.8.

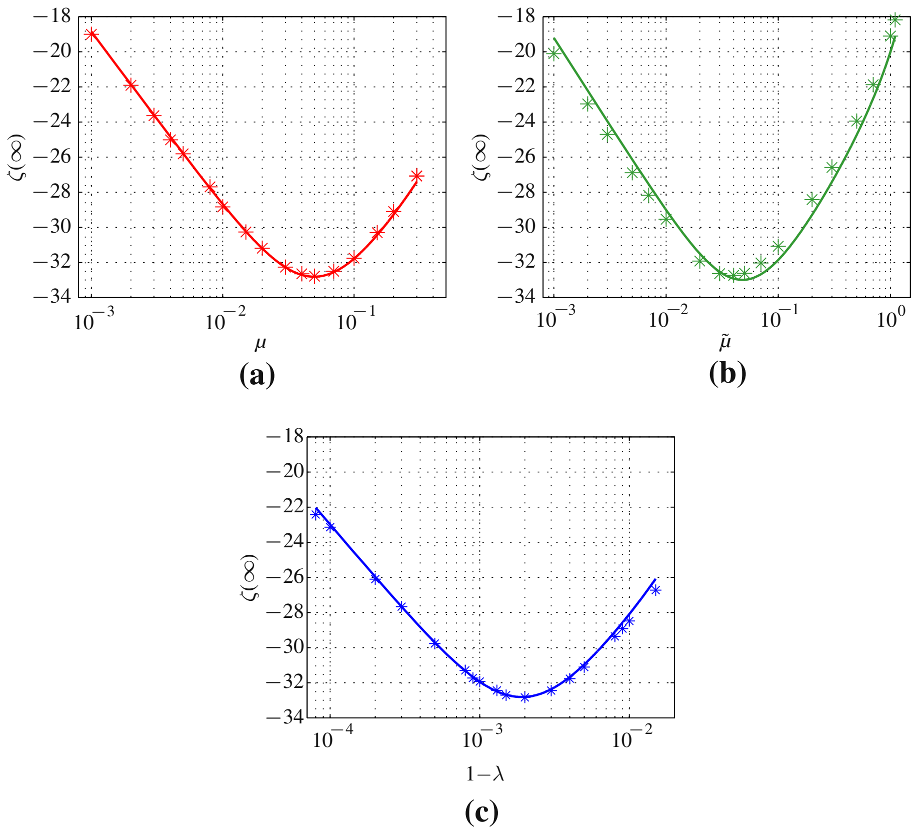


FIGURE 12.32

Theoretical and experimental EMSE as a function of the adaptation parameter for a) LMS, b) NLMS, and c) RLS. The asterisks represent simulation results and the solid lines represent the theoretical models.

Box 7: Fourth-order moments of Gaussian vectors

The second-order moments of a random sequence are related to its average power (of course, this is an approximation unless the sequence is stationary and ergodic.) This makes second-order moments intuitive to work with and relatively easy to evaluate in practice. Therefore, when studying an adaptive filter, it is common to try to describe its performance in terms of the autocorrelation of the input sequence $\boldsymbol{x}(n)$.

During the analysis of LMS, one encounters a term that contains fourth-order powers of the input vector $\boldsymbol{x}(n)$:

$$\boldsymbol{F} \triangleq \mathrm{E}\{\boldsymbol{x}(n)\boldsymbol{x}^H(n)\tilde{\boldsymbol{w}}(n)\tilde{\boldsymbol{w}}^H(n)\boldsymbol{x}(n)\boldsymbol{x}^H(n)\}. \tag{12.273}$$

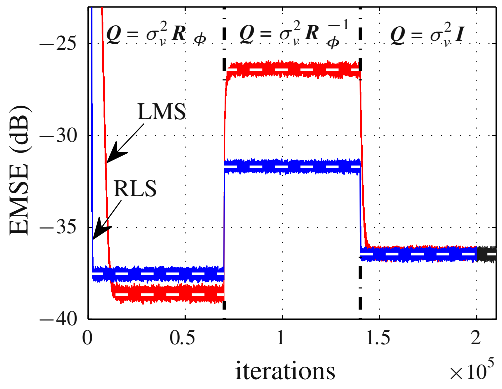


FIGURE 12.33

EMSE for RLS ($\lambda = 0.9955$, $\delta = 4.5 \times 10^{-3}$), and LMS ($\mu = 0.0224$); $b = 0.8$, $\sigma_v^2 = 10^{-2}$, $\sigma_q^2 = 10^{-6}$; mean of 5000 independent runs. The dashed lines represent the predicted values of $\zeta(\infty)$ for each algorithm.

In order to evaluate a term such as this, one needs to know much more information about the statistics of $\boldsymbol{x}(n)$ than is provided simply by its mean and autocorrelation. However, in general this additional information is not available, and one assumes that the signals are Gaussian in order to proceed.

Why not any other distribution? First, because many processes in Nature are indeed approximately Gaussian. Second, because uncorrelated Gaussian variables are also independent, which is of great help when evaluating fourth-order terms, such as $\mathrm{E}\{x^2(n)x^2(n-1)\}$ or $\mathrm{E}\{x^3(n)x(n-2)\}$.

Although in many cases the input sequence is not Gaussian (for example, speech does not follow a Gaussian distribution), this approximation describes the most important features of adaptive filter learning with reasonable accuracy, and leads to a relatively simple model.

In order to simplify the notation, we will write simply $\boldsymbol{x}$ and $\tilde{\boldsymbol{w}}$ instead of $\boldsymbol{x}(n)$ and $\tilde{\boldsymbol{w}}(n)$ in the following. Assume then that $\boldsymbol{x}$ is a Gaussian random vector, with autocorrelation $\mathrm{E}\{\boldsymbol{x}\boldsymbol{x}\} = \boldsymbol{R}$, and that $\tilde{\boldsymbol{w}}$ is another vector independent of $\boldsymbol{x}$, and with autocorrelation $\mathrm{E}\{\tilde{\boldsymbol{w}}\tilde{\boldsymbol{w}}^H\} = \boldsymbol{S}$ (we do not need to know the exact distribution of $\tilde{\boldsymbol{w}}$.) Assume also that all variables have zero mean. Our goal is to evaluate

$$\boldsymbol{F} = \mathrm{E}\{\boldsymbol{x}\boldsymbol{x}^H\tilde{\boldsymbol{w}}\tilde{\boldsymbol{w}}^H\boldsymbol{x}\boldsymbol{x}^H\}.$$

Since $\tilde{\boldsymbol{w}}$ is independent of $\boldsymbol{x}$, we can write

$$\boldsymbol{F} = \mathrm{E}\{\boldsymbol{x}\boldsymbol{x}^H\tilde{\boldsymbol{w}}\tilde{\boldsymbol{w}}^H\boldsymbol{x}\boldsymbol{x}^H\} = \mathrm{E}\{\boldsymbol{x}\boldsymbol{x}^H\mathrm{E}\{\tilde{\boldsymbol{w}}\tilde{\boldsymbol{w}}^H\}\boldsymbol{x}\boldsymbol{x}^H\} = \mathrm{E}\{\boldsymbol{x}\boldsymbol{x}^H\boldsymbol{S}\boldsymbol{x}\boldsymbol{x}^H\}. \tag{12.274}$$

The autocorrelation of $\boldsymbol{x}$ is in general a full matrix, that is,

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1M} \\ r_{12}^* & r_{22} & \dots & r_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{1M}^* & r_{2M}^* & \dots & r_{MM} \end{bmatrix},$$

that is, in general the elements of $\boldsymbol{x}$ are correlated. $\boldsymbol{R}$ has hermitian symmetry and is non-negative definite, which implies that there exists a unitary matrix $\boldsymbol{U}$ such that

$$\boldsymbol{U}^H\boldsymbol{R}\boldsymbol{U} = \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_M \end{bmatrix},$$

where $\lambda_i \geq 0$ for $i = 1, \dots, M$. Define $\boldsymbol{x}' = \boldsymbol{U}^H\boldsymbol{x}$. Then $\mathrm{E}\{\boldsymbol{x}'\boldsymbol{x}'^H\} = \mathrm{E}\{\boldsymbol{U}^H\boldsymbol{x}\boldsymbol{x}^H\boldsymbol{U}\} = \boldsymbol{U}^H\mathrm{E}\{\boldsymbol{x}\boldsymbol{x}^H\}$ $\boldsymbol{U} = \boldsymbol{\Lambda}$, and we see that the entries of $\boldsymbol{x}'$ are uncorrelated. Since $\boldsymbol{x}$ is Gaussian, and a linear transformation of a Gaussian vector is also Gaussian, this means that the entries of $\boldsymbol{x}'$ are independent of each other. Multiplying (12.274) by $\boldsymbol{U}^H$ on the left and $\boldsymbol{U}$ on the right, and recalling that $\boldsymbol{U}^H\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^H = \boldsymbol{I}$, we obtain

$$\boldsymbol{F}' \triangleq \boldsymbol{U}^H\boldsymbol{F}\boldsymbol{U} = \mathrm{E}\{\boldsymbol{U}^H\boldsymbol{x}\boldsymbol{x}^H\underbrace{\boldsymbol{U}\boldsymbol{U}^H}_{=\boldsymbol{I}}\boldsymbol{S}\underbrace{\boldsymbol{U}\boldsymbol{U}^H}_{=\boldsymbol{I}}\boldsymbol{x}\boldsymbol{x}^H\boldsymbol{U}\} = \mathrm{E}\{\boldsymbol{x}'\boldsymbol{x}'^H\boldsymbol{U}^H\boldsymbol{S}\boldsymbol{U}\boldsymbol{x}'\boldsymbol{x}'^H\}.$$

Defining $\bar{\boldsymbol{S}} = \boldsymbol{U}^H\boldsymbol{S}\boldsymbol{U}$, we can expand $\boldsymbol{F}$ in terms of the elements s'_{ij} of $\bar{\boldsymbol{S}}$ and x'_k of $\boldsymbol{x}'$. Each element of $\boldsymbol{F}$ will have terms of the form

$$\mathrm{E}\{x'_{k_1}x'_{k_2}x'^*_{k_3}x'^*_{k_4}\}s'_{ij}.$$

If the signals are real, then, as x'_{k_1} is independent of x'_{k_2} if $k_1 \neq k_2$, these expected values will be nonzero only if $k_1 = k_2 = k_3 = k_4$ or if $k_1 = k_3$ and $k_2 = k_4$, or $k_1 = k_4$ and $k_2 = k_3$, or if $k_1 = k_2$ and $k_3 = k_4$. Evaluating all such terms that appear in the expression for $\boldsymbol{F}'$ will result in

$$\boldsymbol{F}|_{\text{real signals}} = \boldsymbol{R}\mathrm{Tr}(\boldsymbol{S}\boldsymbol{R}) + 2\boldsymbol{R}\boldsymbol{S}\boldsymbol{R}. \tag{12.275}$$

If, on the other hand, the signals are complex-valued, then we need to know how the real and imaginary parts of each entry of $\boldsymbol{x}$ relate to each other. One common assumption is that $\boldsymbol{x}$ is *circularly*-Gaussian. We explain more about this in Section 1.12.2.2.1. For now, we only remark that if this is the case, the real and imaginary parts of each entry of $\boldsymbol{x}$ and of $\boldsymbol{x}'$ are such that $\mathrm{E}\{x_k^2\} = \mathrm{E}\{x_k'^2\} = 0$ for all k (note that $\mathrm{E}\{|x_k|^2\} = r_{kk}$, $\mathrm{E}\{|x'_k|^2\} = \lambda_k$ are still nonzero.) This will result in a different expression for $\boldsymbol{F}$:

$$\boldsymbol{F}|_{\text{complex signals}} = \boldsymbol{R}\mathrm{Tr}(\boldsymbol{S}\boldsymbol{R}) + \boldsymbol{R}\boldsymbol{S}\boldsymbol{R}. \tag{12.276}$$

1.12.5 Extensions and current research

In this section we briefly describe some important extensions to the basic adaptive filtering algorithms, as well as some promising current research topics. The intention is not to describe in any detail all these techniques, but rather to give a list of key references where the interested reader may find more information.

It is usual when starting such lists to put a disclaimer, that the authors do not claim to have included all the important contributions, nor that their list is really of the most important contributions to an area.

This is not intended for legal purposes, nor false modesty: the size of the literature is indeed so large (a search on "adaptive filter" on Google gives over one million hits) that we cannot claim to know with certainty the best of it all. We included therefore the techniques that we found useful during our own research and experience, which is perforce limited.

1.12.5.1 Finite-precision arithmetic

Any operation made using finite-precision arithmetic may involve errors. For example, the product of $a = 0.957$ and $b = 0.542$ would result in $c = 0.518694$. However, if all numbers were stored keeping only three digits, the final result would be $\bar{c} = 0.519$, with an error of $c - \bar{c} = -3.06 \times 10^{-4}$ [82,83]. These *quantization* errors will accumulate and modify the behavior of an adaptive filter in important ways. A precise analysis of the effect of quantization errors is quite complicated, since these errors are highly nonlinear functions of the variables.

When finite-precision arithmetic effects are studied, it is usually important to know exactly how and in which order the arithmetic operations are performed. In particular, the exact numerical representation used is important: fixed-point vs. floating-point, truncation vs. rounding, etc. For this reason, it is more difficult to perform analyses that are equally valid for a large class of filters. The most important differences are between gradient-based algorithms (i.e., algorithms similar to LMS and NLMS) and Hessian-based algorithms (i.e., algorithms similar to RLS). We will give references to works treating both cases, starting with LMS.

1.12.5.1.1 Finite-precision effects in LMS

The simplest models for adaptive filters in finite-precision arithmetic treat the quantization error as random noise, uniformly distributed and independent of all variables [84–87]. These models show that the quantization errors will add another term to the EMSE. However, this term does not converge to zero when the step-size is reduced to zero, even in a stationary environment: quite the opposite, it is inversely proportional to the step-size (similar to the EMSE in a nonstationary environment).

Another important undesirable effect is that the product $\mu e(n)$ may be rounded down to zero when $e(n)$ becomes small (underflow), virtually stopping the adaptation. When the step-size is small, this may happen when the filter is rather far from the optimum solution. Therefore, there is an optimum value for the step-size—neither too large, to avoid increasing the misadjustment, neither too small, to avoid underflow and excessive growth of the EMSE. A similar effect happens in RLS [1, p. 595].

This underflow problem, which is usually known as *stopping phenomenon*, was studied in more detail in [88–91], using a nonlinear model. These works show that the adaptation does not really stop, but rather convergence is reduced to a very low rate. Note that this phenomenon is less important in floating-point arithmetic, since floating-point is able to represent much smaller numbers without underflow than fixed-point arithmetic.

Finite precision effects may have worse consequences, in particular they may make the adaptive filter unstable. In the case of LMS, this phenomenon is rare, and will only happen when the regressor sequence $\{\boldsymbol{\phi}\}$ has a very ill-conditioned covariance matrix, and the noise v_o has a nonzero mean (this nonzero mean may itself appear due to quantization errors) [92–94]. This problem is not difficult to solve, using regularization as described in Section 1.12.5.2. The problem of numerical stability in RLS is much more serious and harder to solve, and has received much attention, as we see next.

In addition to studying the effect of quantization errors in the performance of an adaptive filter, another important topic is how to reduce the computational cost of a filter, taking advantage of implementation details. One such approach is to replace the error $e(n)$ in the LMS recursion by its sign, i.e., to use the recursion

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu \mathrm{sign}\left(e(n)\right) \boldsymbol{\phi}(n), \tag{12.277}$$

where $\mathrm{sign}(\cdot)$ is the sign function, defined by

$$\mathrm{sign}\left(e\right) = \begin{cases} 1, & \text{if } e > 0, \\ 0, & \text{if } e = 0, \\ -1, & \text{if } e < 0. \end{cases} \tag{12.278}$$

The resulting algorithm, known as *sign-LMS*, has a smaller computational cost, if μ is restricted to be a power of two, that is, if $\mu = 2^k$ for an integer (usually negative) k. In this case, $\mu \mathrm{sign}\left(e(n)\right)$ is always equal to $\pm 2^k$ (or zero), and the product $\left[\mu \mathrm{sign}\left(e(n)\right)\right] \boldsymbol{\phi}(n)$ can be implemented in fixed-point arithmetic as M shifts, instead of M multiplications. In floating-point arithmetic, the multiplications are replaced by sums, since we would need to add k to the exponent terms of all entries of $\boldsymbol{\phi}(n)$. This algorithm is very robust (see also Section 1.12.5.4), although its convergence rate under Gaussian inputs is considerably slower than that of LMS. Sign-LMS was thoroughly studied in [81,95–98].

Another possibility, that works surprisingly well, reducing the computational cost with only a slight decrease in convergence speed is the *power-of-two LMS* algorithm [99]. In it, the error is rounded to the nearest power of two, using the function:

$$f\left(e\right) = \begin{cases} \mathrm{sign}(e), & |e| \geq 1, \\ \mathrm{sign}(e) 2^{\lfloor \ln(|e|) \rfloor}, & 2^{-B+1} \leq |e| < 1, \\ 0, & |e| < 2^{-B+1}, \end{cases}$$

where B is the number of bits used to represent the error, and $\lfloor x \rfloor$ returns the largest integer smaller than x. In this case, μ should also be a power of two.

1.12.5.1.2 *Finite-precision effects in RLS*

As we briefly described in Section 1.12.3.3.1, the conventional RLS is numerically unstable in finite precision arithmetic and has $\mathcal{O}(M^2)$ complexity. To solve these problems, different versions of RLS were proposed in the literature. Many of these versions are based on coordinate transformations of the state-space representation of the conventional RLS algorithm. This representation can be obtained by replacing $\boldsymbol{k}(n)$ by $\boldsymbol{P}(n)\boldsymbol{\phi}(n)$ in(12.166), i.e.,

$$\boldsymbol{w}(n+1) = \lambda \boldsymbol{P}(n) \hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1) + \boldsymbol{P}(n)\boldsymbol{\phi}(n) d^*(n). \tag{12.279}$$

Recalling that $\boldsymbol{w}(n) = \boldsymbol{P}(n-1)\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1)$, we can replace $\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1)$ in (12.279) as a function of $\boldsymbol{P}^{-1}(n-1)$ and $\boldsymbol{w}(n)$, which leads to

$$\boldsymbol{w}(n+1) = \lambda \boldsymbol{P}(n) \boldsymbol{P}^{-1}(n-1) \boldsymbol{w}(n) + \boldsymbol{P}(n)\boldsymbol{\phi}(n) d^*(n). \tag{12.280}$$

This equation in conjunction with the definition of the *a priori* error $e(n)$ characterize the adaptation and filtering operations of the RLS algorithm and constitute its state-space representation, i.e.,

$$\begin{bmatrix} \boldsymbol{w}(n+1) \\ e^*(n) \end{bmatrix} = \begin{bmatrix} \lambda \boldsymbol{P}(n)\boldsymbol{P}^{-1}(n-1) & \boldsymbol{P}(n)\boldsymbol{\phi}(n) \\ -\boldsymbol{\phi}^H(n) & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{w}(n) \\ d^*(n) \end{bmatrix}. \tag{12.281}$$

In this representation, $\boldsymbol{w}(n)$ is the state, $e^*(n)$ is the output, and $d^*(n)$ is the input.

Performing a coordinate transformation, (12.281) can be transformed to an unlimited number of systems with the same input-output relation, and hence solving the same least-squares problem (that is, the output error is always the same). Although all these realizations are equivalent in infinite precision arithmetic, the numerical behavior will vary from one coordinate system to another. Thus, the numerical instability of the conventional RLS can be avoided by choosing a convenient transformation of (12.281) [40,100].

An alternative method of implementing RLS is based on the QR decomposition (QRD) [14] to triangularize the input data matrix. The main advantages of QRD-RLS-based algorithms are the possibility of implementation in systolic arrays (e.g., [101]) and the improved numerical behavior in finite precision arithmetic. Some versions require $\mathcal{O}(M^2)$ operations per iteration, but others have a reduced computational cost of $\mathcal{O}(M)$. The low-cost versions of QRD-RLS algorithms are known as *fast*-QR algorithms. Some important references on this subject are [38,101–116]. A good tutorial on QRD-RLS-based algorithms can be found in [43], where the most recent developments as well as the basic concepts are covered.

Another important class of fast algorithms solves the least-squares problem in a recursive form based on the lattice realization. These algorithms are very attractive due to the possible modular implementation and low cost ($\mathcal{O}(M)$). Lattice-RLS-based algorithms are derived by solving the forward and backward linear prediction problems and require time-update and order-update recursions. They explore the time-shift property of the input signal vector (such as in (12.113)) and do not compute explicitly the filter weights. Recall that many applications do not require the computation of the filter weights and have regressors that satisfy the time-shift property as, for example, channel equalization and interference cancellation (see Section 1.12.1.3). In these applications, a fast RLS algorithm can be a reasonable choice since it presents a good tradeoff between convergence rate and computational cost. One of these algorithms is the modified EF-LSL (error-feedback least-squares lattice algorithm) that presents reliable numerical properties when implemented in finite-precision arithmetic [5]. This algorithm was proposed for echo-cancellation applications and was used in the example of Section 1.12.1.1. Some lattice-RLS-based algorithms can be found in the references [5,117–124].

More recently, lattice-RLS algorithms have been generalized to filter structures other than tapped-delay lines, in particular, to Laguerre filters [125–127].

Other fast algorithms that solve the least-squares problem in a recursive form are the fast transversal RLS (FTRLS) algorithms. Unlike the lattice-based algorithms, the FTRLS algorithms require only time-recursive equations and therefore, compute explicitly the filter weights. The main drawback of these algorithms is that they are very sensitive to quantization effects and become unstable if certain actions are not taken [1]. The list of fast transversal RLS algorithms is vast. Some of the most important versions can be found in the references [128–134].

1.12.5.1.3 DCD-RLS

The DCD-RLS algorithm, proposed in [44], is a low-complexity alternative to the RLS algorithm, based on the dichotomous coordinate-descent (DCD) algorithm for function minimization proposed in [135]. The DCD algorithm is designed to be easily implementable in hardware (such as FPGAs, for example), and thus avoids multiplications, divisions and other operations that are costly to implement in hardware—most of its operations are additions and comparisons.

The RLS weight vector estimates are given by the solution of the following set of linear equations, which we repeat from Section 1.12.3.3:

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\boldsymbol{w}(n+1) = \hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n), \tag{12.282}$$

where $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)$ and $\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n)$ are given by the recursions

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n) = \lambda\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n), \tag{12.283}$$

and

$$\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n) = \lambda\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1) + d^*(n)\boldsymbol{\phi}(n). \tag{12.284}$$

The difficulty in solving these expressions is that a general solution for (12.282) involves a number of operations of the order of $M^3(\mathcal{O}(M^3))$ for a filter with M coefficients. This is usually too costly for practical applications, except perhaps for very short filters. The classical RLS algorithm described in Section 1.12.3.3 solves this problem partially by using the matrix inversion lemma to compute the inverse of $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)$ recursively using the already computed inverse of $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1)$. This reduces the total number of computations to $\mathcal{O}(M^2)$. However, the resulting algorithm is still difficult to implement in practice, because numerical errors may easily cause divergence.

The DCD-RLS algorithm avoids both these problems by using a couple of clever tricks. First, assume that you have a good approximation $\hat{\boldsymbol{w}}(n)$ for $\boldsymbol{w}(n)$. Then you could use a recursive algorithm to find an approximation $\hat{\boldsymbol{w}}(n+1)$ to the solution of (12.282), using your current approximation $\hat{\boldsymbol{w}}(n)$ as an initial condition. Since the initial condition is already close to the solution, you would need only a few iterations of your recursive algorithm to obtain $\hat{\boldsymbol{w}}(n+1)$. However, this idea only helps if each iteration of the recursive algorithm is very cheap to compute. We explain next how this can be done using DCD.

DCD minimization of quadratic functions

Let us consider first a one-dimensional quadratic problem

$$\min_w \left\{ f(w) = \frac{1}{2}aw^2 - bw + c \right\},$$

where $a > 0$, b, and c are constants. Assume in addition that you want to implement your algorithm in hardware, using fixed-point arithmetic.

In order to solve this problem, you could proceed as follows. Assume you have an initial approximation $\hat{w}(0)$. Choose a step-size $H > 0$ and compute

$$\Delta f_+ = f(\hat{w}(0)+H) - f(\hat{w}(0)), \quad \Delta f_- = f(\hat{w}(0)-H) - f(\hat{w}(0)).$$

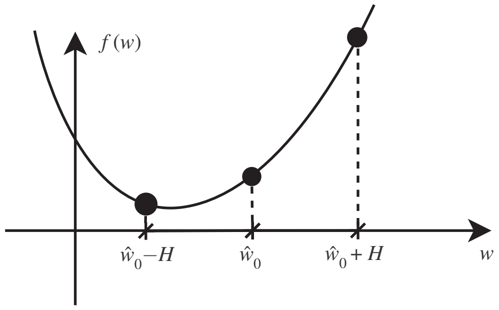


FIGURE 12.34

DCD applied to one-dimensional minimization.

Then, if $\Delta f_+ < 0$, choose $\hat{w}(1) = \hat{w}(0) + H$, and if $\Delta f_- < 0$, choose $\hat{w}(1) = \hat{w}(0) - H$. If both are positive, choose $\hat{w}(1) = \hat{w}(0)$ and reduce the step-size $H \leftarrow H/2$ (note that if $f(\cdot)$ is convex, the case of both $\Delta f_+ < 0$ and $\Delta f_- < 0$ can never occur). With the new estimate $\hat{w}(1)$, the procedure can be repeated to find a new improved estimate $\hat{w}(2)$, and so on. It can be shown that this algorithm will converge to the minimum of $f(\cdot)$ whenever $f(\cdot)$ is a convex function—see Figure 12.34.

Although the description given so far explains the general working of the DCD algorithm, it does not show how it can be efficiently implemented, using as few operations as possible. We will turn to this point now, but already extending the algorithm to minimize a quadratic function of several variables. Consider then the problem of finding the solution $\boldsymbol{w}_o$ to

$$\min_{\boldsymbol{w}} \left\{ f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{R}\boldsymbol{w} - \boldsymbol{b}^T\boldsymbol{w} + c \right\},$$

where $\boldsymbol{R}$ is a positive-definite matrix, $\boldsymbol{b}$ is a vector, and c is a constant (note that the minimum $\boldsymbol{w}_o$ of $f(\boldsymbol{w})$ is also the solution $\boldsymbol{w}_o$ to $\boldsymbol{R}\boldsymbol{w}_o = \boldsymbol{b}$, that is, any algorithm to find $\boldsymbol{w}_o$ can be also used to solve the normal equations (12.282)).

Assume that we have an initial approximation $\hat{\boldsymbol{w}}(0)$ to the solution, and we want to find an improved approximation, changing one entry of $\hat{\boldsymbol{w}}(0)$ at a time. In order to differentiate each new approximation, we use the following notation. Define $\hat{\boldsymbol{w}}^{(0)}(0) = \hat{\boldsymbol{w}}(0)$. We shall first seek an improved estimate to the first entry of $\hat{\boldsymbol{w}}^{(0)}(0)$. The resulting vector will be denoted by $\hat{\boldsymbol{w}}^{(1)}(0)$. Continuing, we find an improved estimate to the second entry of $\hat{\boldsymbol{w}}^{(1)}(0)$, resulting in vector $\hat{\boldsymbol{w}}^{(2)}(0)$ and so on. After updating all entries of the initial estimate, we have $\hat{\boldsymbol{w}}^{(M)}(0)$. We then let $\hat{\boldsymbol{w}}^{(0)}(1) = \hat{\boldsymbol{w}}^{(M)}(0)$, and repeat the procedure.

Then we need to check if $f(\hat{\boldsymbol{w}}^{(0)}(0) \pm H\boldsymbol{e}_1) < f(\hat{\boldsymbol{w}}(0))$ if we add or subtract H to the first entry of $\hat{\boldsymbol{w}}^{(0)}(0)$ ($\boldsymbol{e}_1$ is the vector $[1\ 0\ \dots\ 0]^T$). The change Δf_+ in the cost-function obtained by adding H to the first entry of $\hat{\boldsymbol{w}}^{(0)}(0)$ is

$$\Delta f_+ = \frac{1}{2}\left(\hat{\boldsymbol{w}}^{(0)}(0) + H\boldsymbol{e}_1\right)^T \boldsymbol{R}\left(\hat{\boldsymbol{w}}^{(0)}(0) + H\boldsymbol{e}_1\right) - \boldsymbol{b}^T\left(\hat{\boldsymbol{w}}^{(0)}(0) + H\boldsymbol{e}_1\right) + c \qquad (12.285)$$
$$- \left(\frac{1}{2}\hat{\boldsymbol{w}}^{(0)T}(0)\boldsymbol{R}\hat{\boldsymbol{w}}^{(0)}(0) - \boldsymbol{b}^T\hat{\boldsymbol{w}}^{(0)}(0) + c\right)$$

$$= H\boldsymbol{e}_1^T \boldsymbol{R}\hat{\boldsymbol{w}}^{(0)}(0) + \frac{1}{2}H^2\boldsymbol{e}_1^T\boldsymbol{R}\boldsymbol{e}_1 - H\boldsymbol{b}^T\boldsymbol{e}_1. \tag{12.286}$$

Similarly, the variation Δf_- obtained from subtracting H from the first entry of $\hat{\boldsymbol{w}}(0)$ is

$$\Delta f_- = -H\boldsymbol{e}_1^T \boldsymbol{R}\hat{\boldsymbol{w}}^{(0)}(0) + \frac{1}{2}H^2\boldsymbol{e}_1^T\boldsymbol{R}\boldsymbol{e}_1 + H\boldsymbol{b}^T\boldsymbol{e}_1. \tag{12.287}$$

Therefore, we should make

$$\hat{\boldsymbol{w}}^{(1)}(0) = \begin{cases} \hat{\boldsymbol{w}}^{(0)}(0) + H\boldsymbol{e}_1, & \text{if } \Delta f_+ < 0, \\ \hat{\boldsymbol{w}}^{(0)}(0) - H\boldsymbol{e}_1, & \text{if } \Delta f_- < 0, \\ \hat{\boldsymbol{w}}^{(0)}(0), & \text{otherwise.} \end{cases} \tag{12.288}$$

The same steps can be repeated for the second entry of $\hat{\boldsymbol{w}}^{(1)}(0)$ and so on, that is, we can update the $(m+1)$th entry of $\hat{\boldsymbol{w}}^{(m)}(0)$ to obtain a new approximation $\hat{\boldsymbol{w}}^{(m+1)}(0)$ for $m = 0, \ldots, M-1$. After obtaining $\hat{\boldsymbol{w}}^{(M)}(0)$, we have updated all entries of $\hat{\boldsymbol{w}}(0)$. We can then make $\hat{\boldsymbol{w}}^{(0)}(1) = \hat{\boldsymbol{w}}^{(M)}(0)$ and repeat the procedure. However, if no update is made for $m = 1, \ldots, M$, before repeating we decrease the step H by a factor of two (i.e., $H \leftarrow H/2$). By reducing H only when no update is necessary for any entry of the vector, we avoid problems that would appear if the initial value of H were too small.

Implemented as described so far, the algorithm has a high computational cost. For example, for the evaluation of Δf_+ at each step we must

- Evaluate $\boldsymbol{e}_m^T\boldsymbol{R}\hat{\boldsymbol{w}}^{(m-1)}(i)$. This is equivalent to multiplying the mth row of $\boldsymbol{R}$ by $\hat{\boldsymbol{w}}^{(m-1)}(i)$, involving M multiplications and $M-1$ additions.
- Multiply $-\boldsymbol{e}_m^T\boldsymbol{R}\hat{\boldsymbol{w}}^{(m-1)}(i) + \boldsymbol{b}^T\boldsymbol{e}_m$ by H and add the result to $(H^2/2)R_{m,m}$ ($R_{m,m}$ is the element (m,m) of $\boldsymbol{R}$). This requires 2 multiplications and 2 additions, assuming $H^2/2$ is pre-computed.

The same steps would be necessary for the computation of Δf_-. Note that all these computations would have to be repeated at each step, therefore the update of all entries of $\hat{\boldsymbol{w}}^{(0)}(i)$ would require more than M^2 multiplications.

However, the number of computations can be reduced substantially, if we take advantage of some properties of hardware implementations of arithmetic operations, as we describe next. Assume in the following that all variables are fixed-point binary numbers (see Box 8).

First, we can avoid the computation of $\boldsymbol{R}\hat{\boldsymbol{w}}^{(m-1)}(i)$ by introducing a residue vector that is updated at each step. Let $\boldsymbol{r}^{(0)}(0) = \boldsymbol{b} - \boldsymbol{R}\hat{\boldsymbol{w}}^{(0)}(0)$ be the initial residue vector. Note that this is the residue of solving $\boldsymbol{R}\boldsymbol{w} = \boldsymbol{b}$ using $\hat{\boldsymbol{w}}^{(0)}(0)$ as an approximation to the solution.

At the end of the first step, $\hat{\boldsymbol{w}}^{(0)}(0)$ is updated as in (12.288). Since only one entry of $\hat{\boldsymbol{w}}^{(0)}(0)$ is changed, the modification in the residue is simple to evaluate

$$\boldsymbol{r}^{(1)}(0) = \boldsymbol{b} - \boldsymbol{R}\hat{\boldsymbol{w}}^{(1)}(0) = \begin{cases} \boldsymbol{b} - \boldsymbol{R}(\hat{\boldsymbol{w}}^{(0)}(0) + H\boldsymbol{e}_1) = \boldsymbol{r}^{(0)}(0) - H\boldsymbol{R}\boldsymbol{e}_1, & \text{if } \Delta f_+ < 0, \\ \boldsymbol{b} - \boldsymbol{R}(\hat{\boldsymbol{w}}^{(0)}(0) - H\boldsymbol{e}_1) = \boldsymbol{r}^{(0)}(0) + H\boldsymbol{R}\boldsymbol{e}_1, & \text{if } \Delta f_- < 0, \\ \boldsymbol{r}^{(0)}(0), & \text{otherwise.} \end{cases} \tag{12.289}$$

This still requires M multiplications (H multiplied by the first column of $\boldsymbol{R}$). However, if H is a power of two, that is, if $H = 2^k$ for some integer k, than the computation of $\boldsymbol{r}^{(1)}(0)$ in fixed-point arithmetic requires only M shifts and M additions, which are much easier to implement in hardware than multiplications. Moreover, if both Δf_+ and Δf_- are positive, no update is necessary.

Next, we note that it is not really necessary to compute both Δf_+ and Δf_-: from (12.286) and (12.287), when updating the mth element of $\hat{\boldsymbol{w}}^{(m-1)}(i)$ at step i, we have

$$\Delta f_+ = -H\boldsymbol{e}_m^T\boldsymbol{r}^{(m-1)}(i) + \frac{1}{2}H^2\boldsymbol{e}_m^T\boldsymbol{R}\boldsymbol{e}_m = -Hr_m^{(m-1)}(i) + \frac{1}{2}H^2R_{m,m}, \tag{12.290}$$

where $r_m^{(m-1)}(i)$ is the mth entry of the residue vector $\boldsymbol{r}^{(m-1)}(i)$, $\boldsymbol{r}^{(0)}(i+1) \overset{\Delta}{=} \boldsymbol{r}^{(M)}(i)$, and $\boldsymbol{e}_m$ is the mth column of the $M \times M$ identity matrix. Similarly,

$$\Delta f_- = Hr_m^{(m-1)}(i) + \frac{1}{2}H^2R_{m,m}. \tag{12.291}$$

Then, recalling that H and $R_{m,m}$ are positive ($\boldsymbol{R}$ is positive-definite), $\Delta f_+ < 0$ only if

$$r_m^{(m-1)}(i) > \frac{1}{2}HR_{m,m} \geq 0.$$

Repeating the arguments for Δf_-, we see that the mth entry of $\hat{\boldsymbol{w}}^{(m-1)}(i)$ will be updated if and only if

$$|r_m^{(m-1)}(i)| > \frac{1}{2}HR_{m,m}, \tag{12.292}$$

with updates

$$\hat{\boldsymbol{w}}^{(m)}(i) = \hat{\boldsymbol{w}}^{(m-1)}(i) + H\text{sign}(r_m^{(m-1)}(i))\boldsymbol{e}_m, \tag{12.293}$$

$$\boldsymbol{r}^{(m)}(i) = \boldsymbol{r}^{(m-1)}(i) - H\text{sign}(r_m^{(m-1)}(i))\boldsymbol{R}\boldsymbol{e}_m. \tag{12.294}$$

Equations 12.292–12.294 require one comparison, $M+1$ shifts, and $M+1$ additions. Table 12.10 summarizes the DCD algorithm for minimization of quadratic cost-functions. Note that the algorithm will update the entries of $\hat{\boldsymbol{w}}(i)$ bit by bit. It therefore makes sense to choose the initial step-size H the power-of-two value represented by the most-significant bit in the fixed-point representation being used. The conditional jump in step 10 prevents problems in case H is chosen too small.

As seen in Table 12.10, the total number of operations never exceeds $(2M+2)(B-1)+N_u(4M+3)$ (note that here we are counting additions, comparisons and shits as operations—the algorithm does not involve multiplications). In DCD-RLS, the maximum number of updates N_u can be chosen between 1 and 4 with good performance, while keeping the cost low.

When applied to RLS, one final simplification can be used when implementing DCD: instead of updating all entries of vector $\hat{\boldsymbol{w}}(i)$, we can only update the entry corresponding to the largest residue. As explained in [44], the resulting algorithm is still guaranteed to converge to the optimum solution.

DCD-RLS

As we mentioned before, DCD-RLS uses the DCD algorithm just described to solve the normal equations as each new sample arrives. Since DCD is an iterative algorithm, we can use the current solution $\boldsymbol{w}(n)$ as the initial value for computation of $\boldsymbol{w}(n+1)$. In this way, the number of updates N_u of DCD can be

Table 12.10 Summary of DCD Algorithm for Minimization of a Quadratic Cost-Function

Step	**Initialization:** Choose step $H = 2^k$, number of bits B, and maximum update count N_u. Let $\hat{w} = 0$, $r = b$, $h = H$, and $c = 0$.	**Number of** Additions, Shifts, Comparisons	**Number of** Multiplications
1	for $b = 0 : B - 1$		
2	$\quad h \leftarrow h/2$	1	0
3	$\quad$ flag $\leftarrow 0$	0	0
4	$\quad$ for $m = 1 : M$		
5	$\qquad$ if $\lvert r_m \rvert > \frac{1}{2} h R_{m,m}$	2	0
6	$\qquad\quad \hat{w}_m \leftarrow \hat{w}_m + h\,\text{sign}(r_m)$	1	0
7	$\qquad\quad \boldsymbol{r} \leftarrow \boldsymbol{r} - h\,\text{sign}(r_m)\boldsymbol{R}\boldsymbol{e}_m$	2M	0
8	$\qquad\quad c \leftarrow c + 1$, flag $\leftarrow 1$	1	0
9	$\qquad\quad$ if $c > N_u$, stop algorithm.	1	0
10	$\quad$ if flag = 1, go to step 2	1	0
	Total: (worst case)	$\leq (2M + 2)(B - 1)$ $N_u(2M + 3)$	0

restricted to a small number (in many situations from 1 to 4 updates is enough for performance close to that of exact RLS). This further reduces the computational cost of the algorithm.

In the following, we will use $\boldsymbol{w}(n+1)$ to denote the exact solution of (12.282) (i.e., the exact RLS estimate), and $\hat{\boldsymbol{w}}(n+1)$ to denote the approximation computed by DCD-RLS (both would be equal if we let $N_u \to \infty$). Assume that at time instant n, we have available both $\hat{\boldsymbol{w}}(n)$ and the residue

$$\boldsymbol{r}(n) = \hat{\boldsymbol{r}}_{d\phi}(n-1) - \widehat{\boldsymbol{R}}_{\phi}(n-1)\hat{\boldsymbol{w}}(n), \tag{12.295}$$

In order to apply the DCD algorithm to solve for $\hat{\boldsymbol{w}}(n+1)$, it is convenient to update the difference

$$\Delta\boldsymbol{w}(n+1) = \boldsymbol{w}(n+1) - \hat{\boldsymbol{w}}(n). \tag{12.296}$$

Define also

$$\Delta\widehat{\boldsymbol{R}}_{\phi}(n) = \widehat{\boldsymbol{R}}_{\phi}(n) - \widehat{\boldsymbol{R}}_{\phi}(n-1), \tag{12.297}$$
$$\Delta\hat{\boldsymbol{r}}_{d\phi}(n) = \hat{\boldsymbol{r}}_{d\phi}(n) - \hat{\boldsymbol{r}}_{d\phi}(n-1). \tag{12.298}$$

The normal Eq. (12.282) then become

$$\widehat{\boldsymbol{R}}_{\phi}(n)\left[\hat{\boldsymbol{w}}(n) + \Delta\boldsymbol{w}(n+1)\right] = \hat{\boldsymbol{r}}_{d\phi}(n),$$

and thus

$$\begin{aligned}\widehat{\boldsymbol{R}}_{\phi}(n)\Delta\boldsymbol{w}(n+1) &= \hat{\boldsymbol{r}}_{d\phi}(n-1) + \Delta\hat{\boldsymbol{r}}_{d\phi}(n) - \widehat{\boldsymbol{R}}_{\phi}(n-1)\hat{\boldsymbol{w}}(n) - \Delta\widehat{\boldsymbol{R}}_{\phi}(n)\hat{\boldsymbol{w}}(n) \\ &= \boldsymbol{r}(n) + \Delta\hat{\boldsymbol{r}}_{d\phi}(n) - \Delta\widehat{\boldsymbol{R}}_{\phi}(n)\hat{\boldsymbol{w}}(n) \stackrel{\Delta}{=} \boldsymbol{\beta}(n). \end{aligned} \tag{12.299}$$

Given an approximate solution $\boldsymbol{\Delta}\hat{\boldsymbol{w}}(n+1)$ to these equations, we obtain an updated approximate solution $\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \boldsymbol{\Delta}\hat{\boldsymbol{w}}(n+1)$. Note that the residue of solving (12.282) using $\hat{\boldsymbol{w}}(n+1)$,

$$\begin{aligned}
\boldsymbol{r}(n+1) &= \hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n) - \widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\widehat{\boldsymbol{w}}(n+1) \\
&= \hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1) + \boldsymbol{\Delta}\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n) - \widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\left[\hat{\boldsymbol{w}}(n) + \boldsymbol{\Delta}\hat{\boldsymbol{w}}(n+1)\right] \\
&= \hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1) + \boldsymbol{\Delta}\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n) - \widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\boldsymbol{\Delta}\hat{\boldsymbol{w}}(n+1) - \left[\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1) + \boldsymbol{\Delta}\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\right]\hat{\boldsymbol{w}}(n) \\
&= \boldsymbol{r}(n) + \boldsymbol{\Delta}\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n) - \widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\boldsymbol{\Delta}\hat{\boldsymbol{w}}(n+1) - \boldsymbol{\Delta}\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\hat{\boldsymbol{w}}(n) \\
&= \boldsymbol{\beta}(n) - \widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\boldsymbol{\Delta}\hat{\boldsymbol{w}}(n+1), \qquad (12.300)
\end{aligned}$$

is exactly the residue of approximately solving (12.299) by $\boldsymbol{\Delta}\hat{\boldsymbol{w}}(n+1)$.

To complete, we only need to evaluate $\boldsymbol{\Delta}\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)$ and $\boldsymbol{\Delta}\widehat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n)$. From (12.297), (12.298), (12.283) and (12.284), we obtain

$$\boldsymbol{\Delta}\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n) = (\lambda-1)\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n), \qquad (12.301)$$

$$\boldsymbol{\Delta}\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n) = (\lambda-1)\widehat{\boldsymbol{d}}_{d\boldsymbol{\phi}}(n-1) + d(n)\boldsymbol{\phi}(n). \qquad (12.302)$$

Substituting these results in the definition of $\boldsymbol{\beta}(n)$, we obtain

$$\begin{aligned}
\boldsymbol{\beta}(n) &= \boldsymbol{r}(n) + (\lambda-1)\hat{\boldsymbol{r}}_{d\boldsymbol{\phi}}(n-1) + d(n)\boldsymbol{\phi}(n) - \left[(\lambda-1)\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n)\right]\hat{\boldsymbol{w}}(n) \\
&= \boldsymbol{r}(n) + (\lambda-1)\boldsymbol{r}(n) + e(n)\boldsymbol{\phi}(n) = \lambda\boldsymbol{r}(n) + e(n)\boldsymbol{\phi}(n). \qquad (12.303)
\end{aligned}$$

We can therefore use the DCD algorithm to compute an approximate solution to (12.299) and update the residue defined by (12.300), as described in Table 12.11.

As we mentioned before, the number of operations of the DCD algorithm can be reduced if we modify the algorithm in Table 12.10 to update only the entries of the weight vector corresponding to the largest residue entry, as described in Table 12.12.

The total number of operations of DCD-RLS as described in Tables 12.11 and 12.12 is $M^2 + 4M$ multiplications and $M^2/2 + 3.5M + (2M+1)N_u + B$ additions. Most of these operations are due to the updating of $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}$ in Table 12.11, which is responsible for the terms in M^2. While this cannot be helped for general regressors $\boldsymbol{\phi}(n)$, the operation count can be substantially reduced when our filter is modeling an FIR relation as in (12.41) (i.e., $\boldsymbol{\phi}(n)$ is a tap-delay line), as we show next.

If

$$\boldsymbol{\phi}(n) = \begin{bmatrix} x(n) & x(n-1) & \dots & x(n-M+1) \end{bmatrix}^T, \qquad (12.304)$$

Table 12.11 Summary of the DCD RLS Algorithm for General Regressors

```
Initialization:
$\hat{\boldsymbol{w}}(0) = \mathbf{0}$, $\boldsymbol{r}(0) = \mathbf{0}$, $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(-1) = \boldsymbol{\Pi} > 0$
for $n = 0, 1, 2, \ldots$
   $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n) = \lambda\widehat{\boldsymbol{R}}(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n)$ (use (12.306) for tap-delay lines)
   $\hat{y}(n) = \hat{\boldsymbol{w}}^T(n)\boldsymbol{\phi}(n)$,
   $e(n) = d(n) - \hat{y}(n)$,
   $\boldsymbol{\beta}(n) = \lambda\boldsymbol{r}(n) + e(n)\boldsymbol{\phi}(n)$,
Use DCD algorithm to compute new $\Delta\hat{\boldsymbol{w}}(n+1)$ and $\boldsymbol{r}(n+1)$ from $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)\Delta\hat{\boldsymbol{w}}(n+1) = \boldsymbol{\beta}(n)$,
   $\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \Delta\hat{\boldsymbol{w}}(n+1)$.
```

Table 12.12 Summary of Leading-Element DCD Algorithm for Use in DCD-RLS, from [44]

Step	**Initialization: Choose initial condition step $H = 2^k$, number of bits B, and maximum update count N_u. Let $\Delta\hat{\boldsymbol{w}} = \mathbf{0}$, $\boldsymbol{r} = \boldsymbol{\beta}(n)$, $h = H/2$, $b = 1$.**
	for $k = 1 : N_u$
1	$p = \arg\max_{1\le m\le M}\{\lvert r_m\rvert\}$, go to step 4
2	$b \leftarrow b+1$, $h \leftarrow h/2$
3	Stop if $b > B$
4	Go to step 2 if $\lvert r_p\rvert \le (h/2)R_{p,p}$
5	$\Delta\hat{w}_p = \Delta\hat{w}_p + \text{sign}(r_p)h$
6	$\boldsymbol{r} = \boldsymbol{r} - \text{sign}(r_p)h\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}\boldsymbol{e}_p$

then $\widehat{\boldsymbol{R}}(n)$ and $\widehat{\boldsymbol{R}}(n-1)$ share a common structure. Assume that the initial condition $\widehat{\boldsymbol{R}}(-1) = \boldsymbol{\Pi} = \text{diag}(\lambda^{M-1}, \lambda^{M-2}, \ldots, 1)$. Then

$$\widehat{\boldsymbol{R}}(0) = \lambda\begin{bmatrix}\lambda^{M-1} & 0 & \ldots & 0\\ 0 & \lambda^{M-2} & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & 1\end{bmatrix} + \begin{bmatrix}x(0)\\ 0\\ \vdots\\ 0\end{bmatrix}[x(0)\quad 0\quad \ldots\quad 0]$$

$$= \begin{bmatrix}\lambda^M + x^2(0) & 0 & \ldots & 0\\ 0 & \lambda^{M-1} & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & \lambda\end{bmatrix} = \left[\begin{array}{c|c}\lambda^M + x^2(0) & \mathbf{0}^T\\ \hline \mathbf{0} & [\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(-1)]_{1:M-1,1:M-1}\end{array}\right],$$

where $[\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(-1)]_{1:M-1,1:M-1}$ denotes the top-left $(M-1)\times(M-1)$ block of $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(-1)$. We note that, except for its first row and column, $\widehat{\boldsymbol{R}}(0)$ can be obtained directly from $\widehat{\boldsymbol{R}}(-1)$.

Following this observation, we claim that for tapped-delay line regressors, the computation of $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)$ in Table 12.11 can be replaced by an update of only its first column $[\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)]_1$, that is,

$$[\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)]_1 = \lambda[\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1)]_1 + x(n)\boldsymbol{\phi}(n). \tag{12.305}$$

Since $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)$ is symmetric, its first row is the transpose of (12.306), and does not need to be evaluated again.

We show next by induction that this pattern holds for all n. We just showed that it it true for $n = 0$. Assume then that for a certain n, it holds that

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n) = \left[\begin{array}{c|c} \lambda[\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1)]_1 + x(n)\boldsymbol{\phi}(n) & \begin{array}{c}(\cdot)^T \\ \\ [\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1)]_{1:M-1,1:M-1}\end{array}\end{array}\right], \tag{12.306}$$

where we use $(\cdot)^T$ to indicate that the entries of the first row are the same as the entries of the first column. We show next that $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n+1)$ must follow the same pattern. Indeed,

$$\begin{aligned}
\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n+1) &= \lambda\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n) + \begin{bmatrix} x(n+1) \\ x(n) \\ \vdots \\ n(n-M+2)\end{bmatrix}\begin{bmatrix} x(n+1) & x(n) & \dots & x(n-M+2)\end{bmatrix} \\
&= \left[\begin{array}{c|c} \lambda[\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)]_1 & \begin{array}{c}(\cdot)^T \\ \\ \lambda[\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1)]_{1:M-1,1:M-1}\end{array}\end{array}\right] \\
&+ \left[\begin{array}{c|c} x(n+1)\boldsymbol{\phi}(n+1) & \begin{array}{c}(\cdot)^T \\ \begin{bmatrix} x^2(n) & \dots & x(n)x(n-M+2) \\ \vdots & \ddots & \vdots \\ x(n)x(n-M+2) & \dots & x^2(n-m+2)\end{bmatrix}\end{array}\end{array}\right] \\
&= \left[\begin{array}{c|c} \lambda[\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)]_1 + x(n+1)\boldsymbol{\phi}(n+1) & \begin{array}{c}(\cdot)^T \\ \\ [\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)]_{1:M-1,1:M-1}\end{array}\end{array}\right],
\end{aligned}$$

where in the last step we identified the (2, 2) block of the intermediate result with the first $M-1$ rows and columns of $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)$, from (12.283).

Note that in practice we should not actually move the entries of $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n-1)$ to their new position; rather, we should use an indexing system to access the entries of $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)$ to avoid this costly moving operation. The total number of operations of DCD-RLS using (12.306) is $3M$ multiplications and $2MN_u + 6M$ additions. This is not much larger than the complexity of NLMS. The DCD-RLS algorithm has been implemented in FPGAs and run for long periods of time without observation of divergence [44].

Figure 12.35 shows a comparison of RLS and DCD-RLS in a system identification problem. We are trying to identify an $M = 10$ FIR filter, so we use (12.306) to update $\widehat{\boldsymbol{R}}_{\boldsymbol{\phi}}(n)$ with lower cost. The input

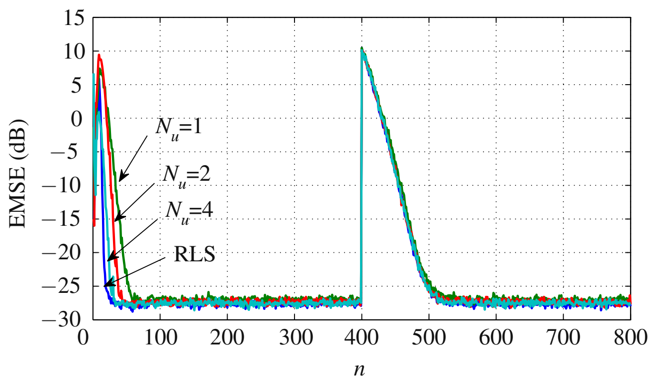


FIGURE 12.35

EMSE for RLS and DCD-RLS for different values of N_u, with $\lambda = 0.95$, $M = 10$, and for DCD, $H = 2$ and $B = 16$. Average of $L = 500$ realizations.

signal $x(n)$ is white Gaussian noise with zero mean and unit variance. The optimum coefficient vector $\boldsymbol{w}_o(n)$ follows a random-walk model as in (12.182) with $\boldsymbol{Q} = 10^{-5}\boldsymbol{I}$ (the initial value, $\boldsymbol{w}_o(0)$ is random, taken from a Gaussian distribution with covariance matrix equal to $\boldsymbol{I}$). Both RLS and RLS-DCD use a forgetting factor of $\lambda = 0.95$. RLS-DCD is implemented with leading DCD (Table 12.12), $B = 16$ and $H = 2$. The maximum number of updates N_u is varied from 1 to 8. The optimum vector $\boldsymbol{w}_o(n)$ suffers an abrupt change at $n = 400$. The learning curves were obtained from the average of $L = 500$ realizations. As the figure shows, even for $N_u = 1$, the performance of RLS-DCD is very close to that of RLS. The only difference is a smaller convergence rate at the start of the algorithm. However, after the abrupt modification of $\boldsymbol{w}_o(n)$ at instant $n = 400$, all filters converge equally quickly to the new vector. This shows that RLS-DCD does not loose in tracking capability when compared to standard RLS.

1.12.5.2 Regularization

There are applications in which the autocorrelation matrix $\boldsymbol{R}_{\boldsymbol{\phi}}$ may become very ill-conditioned (i.e., nearly singular). In these cases, it can be shown that the LMS weight estimates may slowly drift to quite large values, even though the error remains small. The problem arises when the filter is implemented in finite-precision arithmetic, because the variables holding the filter weights may overflow, thereby making the error grow suddenly very large.

This can happen, for example, for certain kinds of equalizers in communications [87,92,93]. The mechanism through which the LMS coefficients may slowly drift is explained in [94,136].

The usual solution to this problem is to modify the cost function to add a term proportional to $\|\boldsymbol{w}\|^2$, so that large values of the weight vector are penalized:

$$J_{\text{leaky}} = \mathrm{E}\left\{\left|d(n) - \boldsymbol{w}^H\boldsymbol{\phi}(n)\right|^2\right\} + \alpha\|\boldsymbol{w}\|^2, \tag{12.307}$$

where $\|\cdot\|$ is the Euclidean norm and $\alpha > 0$ is a small constant. With this cost function, the corresponding algorithm is

$$\boldsymbol{w}(n+1) = (1-\mu\alpha)\boldsymbol{w}(n) + \mu e^*(n)\boldsymbol{\phi}(n). \tag{12.308}$$

This is known as *leaky*-LMS algorithm [4]. It can be shown that the introduction of leakage will bias the solution away from the Wiener solution, but will prevent any excessive growth of the weight coefficients. A detailed analysis of the leaky-LMS algorithm is available in [137]. Reference [94] proposes a modified version of the algorithm without the bias.

Difficulties with singular or near-singular autocorrelation matrices also appears with RLS. For example, if the input is a periodic signal with little noise, the autocorrelation matrix will become nearly singular, and matrix $\boldsymbol{P}(n)$ in RLS will diverge. This problem, as well as a solution to it using variable forgetting-factor, is described in [138]. A constant regularization method based on DCD can be found in [139]. Another approach, valid only for short filters, but which allows constant regularization, is described in [140].

The leaky-LMS algorithm has found other important applications, most particularly in beamforming, in which it is used to turn the beamformer robust against array imperfections [141]. The algorithms used in beamforming are examples of constrained adaptive filters, in which the cost function incorporates equality conditions that must be satisfied at all times—see Section 1.12.5.11.

Another algorithm closely related to leaky-LMS, in which the extra term in the cost function penalizes large values of the filter output (that is, $\alpha|\boldsymbol{w}^H(n)\boldsymbol{\phi}(n)|^2$ instead of $\alpha\|\boldsymbol{w}(n)\|^2$ in (12.307)), was recently employed to improve the performance of FIR adaptive filters in the presence of saturation nonlinearities [142].

More recent applications of regularization involve the use of the ℓ_1-norm,

$$\|\boldsymbol{w}\|_1 = |w_0| + |w_1| + \ldots + |w_{M-1}|,$$

instead of the Euclidean norm. This is because the ℓ_1 norm promotes solutions that are sparse, that is, a filter that minimizes the cost function

$$J_{\text{sparse}} = \mathrm{E}\left\{\left|d(n) - \boldsymbol{w}^H\boldsymbol{\phi}(n)\right|^2\right\} + \alpha\|\boldsymbol{w}\|_1, \tag{12.309}$$

will favor solutions in which the vector $\boldsymbol{w}$ has only a few nonzero entries (an intuitive explanation for this can be found in [143], which is a good introduction to compressive sensing). Algorithms for promoting sparsity have been receiving much attention lately, due to the interest in compressive sensing applications [144–148]. See also the references for the proportionate NLMS (PNLMS) algorithm in Section 1.12.5.3.

1.12.5.3 Variable step-size

The steady-state analysis of Section 1.12.4.4 shows that the step-size μ plays an import role in controlling the performance of the LMS algorithm since it is one of the main factors affecting the convergence rate and the misadjustment. To obtain a better tradeoff between convergence speed and steady-sate EMSE, several authors proposed different forms of adjusting the step-size of LMS. Variable step-size algorithms allow the filters to dynamically adjust their performance in response to conditions in the input data and error signals.

In this context, one of the most popular algorithms is the variable step-size LMS (VSLMS) proposed in [149]. In the VSLMS algorithm, each coefficient $w_k(n)$, $k = 0, \ldots, M-1$ is updated with a time-varying step-size $\mu_k(n)$, whose adjustment is given by

$$\mu_k(n) = \mu_k(n-1) + \rho \text{sign}\left[e(n)\phi_k(n)\right] \text{sign}[e(n-1)\phi_k(n-1)], \tag{12.310}$$

where ρ is a small positive constant. This algorithm operates in a very intuitive way. When the algorithm has not yet converged, the gradient term $e(n)\phi_k(n)$ shows a positive or negative direction in successive iterations and the step-size $\mu_k(n)$ increases up to an allowed maximum value. On the other hand, near steady-state $e(n)\phi_k(n)$ approaches zero and its sign changes in successive iterations, which reduces $\mu_k(n)$ (the step-size is lower-bounded by a small minimum value). Therefore, during the initial convergence VSLMS uses a large step-size, which leads to a high convergence rate. When its coefficients are close to the optimum solution, the algorithm is updated with a small step-size, which allows VSLMS to achieve a small EMSE. A variant of this algorithm is obtained by dropping the sign functions in (12.310), i.e.,

$$\mu_k(n) = \mu_k(n-1) + \rho e(n)e(n-1)\phi_k(n)\phi_k(n-1). \tag{12.311}$$

The choice between (12.310) and (12.311) depends on the application [150].

The NLMS algorithm, proposed independently by Nagumo and Noda [151] and Albert and Gardner [152], can be interpreted a variable step-size LMS with a particular choice for $\mu(n)$ (see Section 1.12.3.2). An important variant of NLMS is the so-called *power-normalized* LMS (PN-LMS) algorithm, which uses the following variable step-size

$$\mu(n) = \frac{\tilde{\mu}}{\varepsilon + p(n)}, \tag{12.312}$$

where

$$p(n) = \gamma p(n-1) + (1-\gamma)|\phi_0(n)|^2, \tag{12.313}$$

with $p(-1) = 0$ and γ being a positive scalar chosen from within the interval $0 < \gamma \leq 1$. This algorithm is useful when the regressor is a tapped-delay line, that is, when $\phi_i(n) = x(n-i)$, so that the scalar $p(n)$ is an estimate for the power of the input sequence $\{x(n)\}$. The range of $\tilde{\mu}$ in (12.312) is $0 \leq \tilde{\mu} \leq \frac{2}{M}$ [3].

Over the years, several variable step-size algorithms were proposed in the literature. Some of these algorithms are in the papers [153–167] and in their references. Besides the variable step-size approaches, an alternative scheme to improve the tradeoff between convergence speed and steady-sate EMSE is constituted by combinations of adaptive algorithms, described in Section 1.12.5.8.1.

Another class of variable step-size algorithms was developed specifically to accelerate the convergence of filters whose optimum weight vector $\boldsymbol{w}_o$ is sparse, that is, has only a small fraction of nonzero elements. The primary application for this is echo cancellation, mainly for echo due to reflections in the telephone connection. The algorithms are known as *proportionate* NLMS (PNLMS) algorithms, and have usually very good performance for approximating sparse weight vectors [168–172].

Table 12.13 Fourth- and Sixth-Order Moments of Different Distributions

Distribution	$\frac{\mathrm{E}\{(x-\mu_x)^4\}}{\sigma_x^4}$	$\frac{\mathrm{E}\{(x-\mu_x)^6\}}{\sigma_x^6}$
Binary	1	1
Uniform	9/5	27/7
Gaussian	3	15
Exponential	6	90

1.12.5.4 Non-Gaussian noise and robust filters

Consider a problem in which we know that the regressor $\boldsymbol{\phi}(n)$ and the desired signal $d(n)$ are related through a model (we assume for simplicity that all variables are real)

$$d(n) = \boldsymbol{w}_o^T \boldsymbol{\phi}(n) + v(n), \tag{12.314}$$

in which $v(n)$ is independent of $\boldsymbol{\phi}(n)$. It is shown in [173] that minimization of the *fourth* power of the error may lead to an adaptive filter with better compromise between convergence rate and excess mean-square error than LMS, if $v(n)$ is *sub-Gaussian*. Let us explain what this means.

Recall that for any random variable x, it holds that $0 \le \sigma_x^2 = \mathrm{E}\{x^2\} - \big(\mathrm{E}\{x\}\big)^2$. Similarly, if we define $\mu_x = \mathrm{E}\{x\}$,

$$\mathrm{E}\left\{(x-\mu_x)^4\right\} \ge \left(\mathrm{E}\left\{(x-\mu_x)^2\right\}\right)^2 = \sigma_x^4, \quad \mathrm{E}\left\{(x-\mu_x)^6\right\} \ge \left(\mathrm{E}\left\{(x-\mu_x)^2\right\}\right)^3 = \sigma_x^6.$$

In general, we can write $\mathrm{E}\{(x-\mu_x)^4\} = \alpha\sigma_x^4$, with $\alpha \ge 1$. For example, consider the distributions in Table 12.13. Distributions for which $\frac{\mathrm{E}\{(x-\mu_x)^4\}}{\sigma_x^4} < 3$, such as binary and uniform, are called *sub-Gaussian*, and those for which $\frac{\mathrm{E}\{x^4\}}{\left(\mathrm{E}\{(x-\mu_x)^2\}\right)^2} > 3$, such as the exponential, are called *super-Gaussian*. These relations are usually given in terms of the *kurtosis* γ_4 of a distribution

$$\gamma_4 = \frac{\mathrm{E}\left\{(x-\mu_x)^4\right\}}{\sigma_x^4} - 3.$$

Therefore, a distribution is sub-Gaussian if its kurtosis is negative, and super-Gaussian if its kurtosis is positive. A distribution with positive kurtosis is such that its probability density function (pdf) $f(x)$ decays more slowly than the Gaussian as $|x| \to \infty$.

The authors of [173] show that when the noise $v(n)$ is sub-Gaussian, an algorithm based on the minimization of

$$J_{\mathrm{LMP}} = \mathrm{E}\{|e(n)|^p\}, \tag{12.315}$$

with $p > 2$ results in lower steady-state excess mean-square error (EMSE) than LMS. They study particularly the case $p = 4$, for which the stochastic gradient algorithm is of the form

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e^3(n)\boldsymbol{\phi}(n). \tag{12.316}$$

This algorithm is known as *least-mean fourth* (*LMF*) algorithm. Analyses of general algorithms with error nonlinearities can be found in [174], and a unified analysis is available in [175].

The LMF algorithm can be understood intuitively as a variable-step version of LMS:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \left(\mu e^2(n)\right) e(n)\boldsymbol{\phi}(n).$$

When the error is large, the "equivalent" step-size $\mu e^2(n)$ is large, and the filter converges quickly. On the other hand, when the error is small, the "equivalent" step-size is reduced, which consequently reduces the EMSE.

This idea works well if the distributions of $\boldsymbol{\phi}(n)$ and $v(n)$ are indeed sub-Gaussian, but it was soon noticed that the algorithm becomes sensitive to outliers (large but rare values of the input signals). This is also easy to understand: if the error is too large, the "equivalent" step-size may be such that the recursion becomes unstable. References [176,177] proposed changing the cost function to a mixture of quadratic and quartic, in an attempt to reduce this sensitivity. In [178, p. 313], a different modification of the cost function is proposed, with the same purpose. The new cost function is the following

$$J_{\text{LMF}, \text{mod}} = \text{E}\left\{f\left(e(n)\right)\right\}, \quad f(e) = \begin{cases} e^4, & \text{if } |e| \le \gamma, \\ e^2, & \text{if } |e| > \gamma, \end{cases} \tag{12.317}$$

in which γ is a constant. References [45,59] show that when the regressor is Gaussian, LMF has a non-zero probability of diverging. From these results one can see that the solution proposed in [178] indeed stabilizes the algorithm.

Since the LMF recursion contains a cubic nonlinearity, it can be used as a simpler platform to understand the behavior of blind-equalization algorithms such as CMA (see Section 1.12.5.5), which also contains a cubic nonlinearity.

Set-membership algorithms [1], which are explained in more detail in Section 1.12.5.10, can be seen as an extreme version of (12.317), in which $f(e)$ is reduced to simply 0 when $|e| \le \gamma$. The corresponding cost function is

$$J_{\text{SM}} = \text{E}\left\{f\left(e(n)\right)\right\}, \quad f(e) = \begin{cases} 0, & \text{if } |e| \le \gamma, \\ e^2, & \text{if } |e| > \gamma. \end{cases} \tag{12.318}$$

Set-membership algorithms are particularly well-suited to bounded noise, i.e., to the case in which one knows that $|v| \le \gamma$.

All these methods are suited to the case in which the noise $v(n)$ is sub-Gaussian. When the noise is super-Gaussian, there is a small but significant probability that $v(n)$ will assume a very large value. This may make the performance of even LMS become poor. The solution is to give less weight to large errors, and one choice is to use $p < 2$ in (12.315). This will give rise to several *robust* adaptive filters, that is, algorithms that are insensitive to outliers. It should be noticed that LMS itself is quite robust in terms of energy (quadratic) relations: in fact, [179] shows that LMS is optimal in the $\mathcal{H}_\infty$ sense (see also [178, ch. 17]). However, for noise distributions with higher kurtosis, it is convenient to use a smaller value of p [180–183].

One important algorithm of this class is the *sign-LMS* algorithm [98,184], which is obtained from (12.315) with $p = 1$, and which we already saw in Section 1.12.5.1.1.

For very impulsive signals, methods based on *order statistics* (such as the median, for example), may produce better results, although their complexity tends to be higher [185].

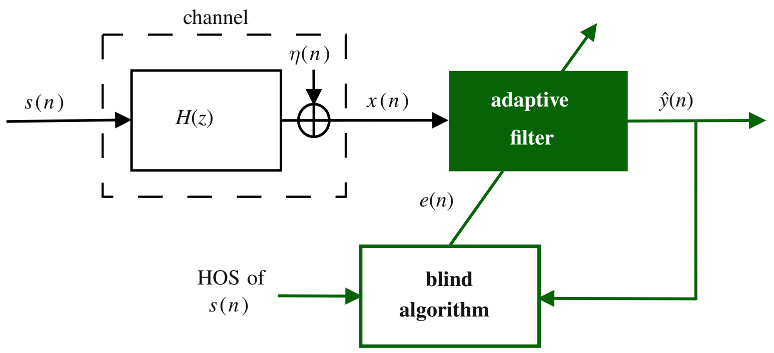


FIGURE 12.36

Simplified communications system with a blind adaptive equalizer.

1.12.5.5 Blind equalization

As described in Section 1.12.1.3, the objective of equalization is to mitigate the intersymbol interference (ISI) introduced by dispersive channels in order to recover the transmitted sequence. Assuming that the equalizer is an adaptive FIR filter, its coefficients may be updated in two different ways:

i. using a supervised algorithm in the training and decision-directed modes, as explained in Section 1.12.1.3; or

ii. using a blind equalization algorithm, which uses higher-order statistics (HOS) of the transmitted signal instead of a training sequence, as shown in Figure 12.36. In this case, the available bandwidth is used in an more efficient manner due to the absence of a training sequence.

The literature contains many blind equalization algorithms based on HOS, but the most popular is the constant modulus algorithm (CMA), proposed independently by Godard [186] and Treichler and Agee [187] in the 1980s. CMA seeks to minimize the constant-modulus cost function defined as

$$J_{\text{CM}} = \text{E}\left\{(r - |\hat{y}(n)|^2)^2\right\}, \qquad (12.319)$$

where $r = \text{E}\{|s(n)|^4\}/\text{E}\{|s(n)|^2\}$ is a dispersion constant, which contains information about higher-order statistics of the transmitted signal $s(n)$ and $\hat{y}(n) = \boldsymbol{w}^H\boldsymbol{x}(n)$ is the equalizer output. This function penalizes deviations in the modulus of the equalized signal away from the dispersion constant r. Different from the mean-square-error cost function used in supervised adaptive filtering, J_{CM} is not convex in relation to the coefficients of the equalizer (see the definition of convex functions in Box 5). It has local minima, and constant-modulus-based algorithms can get stuck at these suboptimal solutions.

CMA is obtained from an instantaneous approximation for the gradient of J_{CM} in relation to $\boldsymbol{w}$. Defining the estimation error

$$e(n) = (r - |\hat{y}(n)|^2)\hat{y}(n), \qquad (12.320)$$

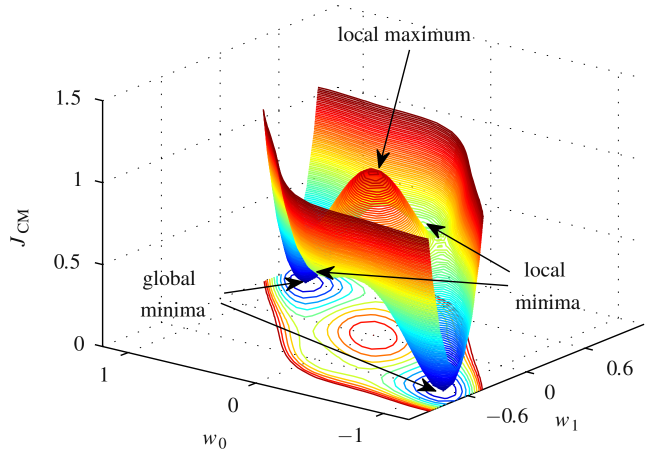


FIGURE 12.37

Constant-modulus cost function as a function of the equalizer weights.

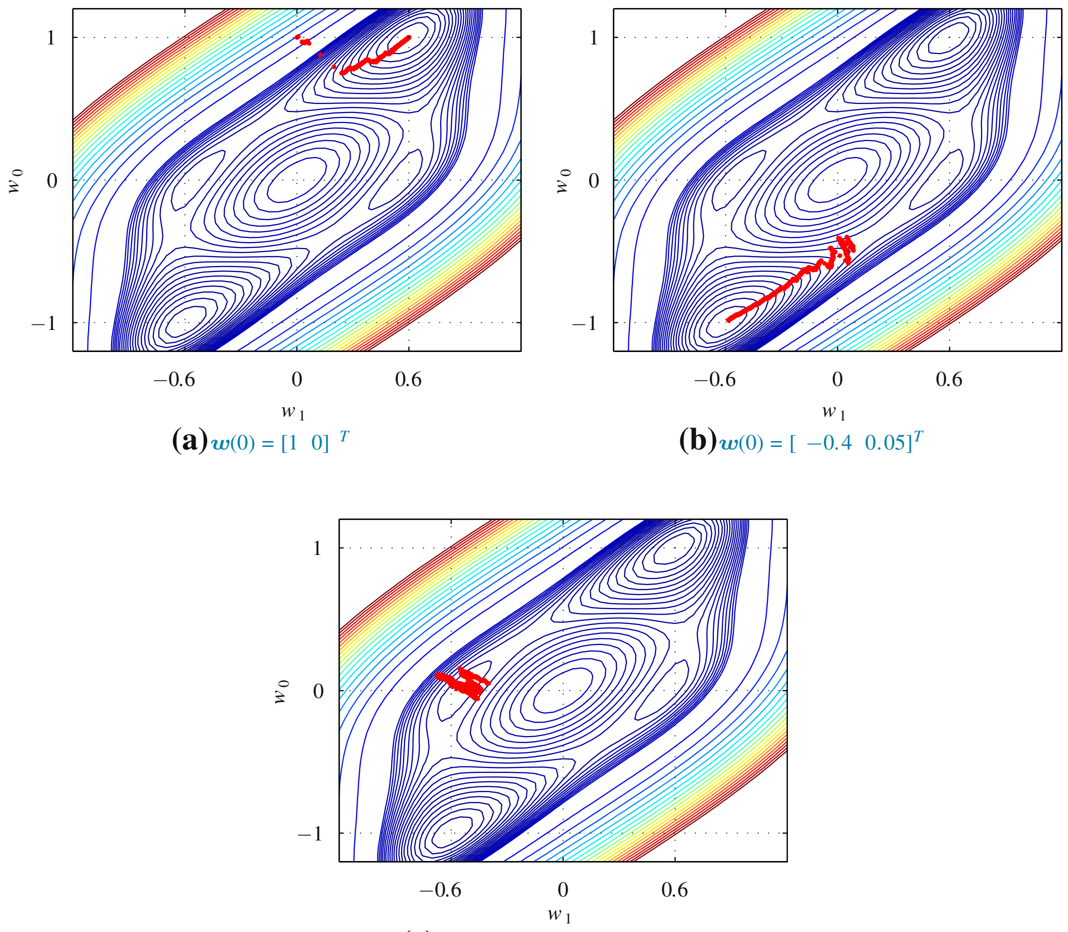
the update equation of CMA can be written as

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e^*(n)\boldsymbol{x}(n), \tag{12.321}$$

where μ stands for a step-size. Since CMA is a stochastic-gradient type algorithm, the similarity of (12.321) to the LMS update equation is not surprising. Due to this similarity, CMA is sometimes interpreted as a blind version of the LMS algorithm [188]. Thus, as in LMS, small step-sizes lead to a small misadjustment and slow convergence. However, the similarity to LMS stops here. The multimodality of the CM cost function makes the prediction of the behavior of CMA a hard task.

To illustrate the constant-modulus cost function, we assume the transmission of a binary signal $\{-1, +1\}$ through the IIR channel $H(z) = 1/(1 + 0.6z^{-1})$ in the absence of noise [189]. Considering an FIR equalizer with $M = 2$ coefficients, we can obtain a 3-D plot of J_{CM} as a function of w_0 and w_1 as shown in Figure 12.37. As indicated in the figure, J_{CM} contains two global minima at $\pm[1\ 0.6]^T$ and two local minima. Note that if $\boldsymbol{w}$ converges to one of the global minima, the transmitted sequence or its inverse will be recovered. These two possibilities occur due the fact that CMA seeks to recover only the modulus of the transmitted signal and does not solve phase ambiguities introduced by the channel. The phase ambiguities can be corrected by using differential code, which is based on the change of sign between two successive samples.

The animations in Figure 12.38 illustrate the behavior of CMA initialized at the different points. Initializing at $\boldsymbol{w}(0) = [1\ 0]^T$, the weight vector converges to $[+1\ \ +0.6]^T$. Now, considering $\boldsymbol{w}(0) = [-0.4\ 0.05]^T$, we get $\boldsymbol{w}(n) \approx [-1\ \ -0.6]^T$ at the steady-state. On the other hand, if CMA is initialized at $[0.05\ \ -0.4]^T$, the algorithm gets stuck at a local minimum. We should notice that in the presence of noise or depending on the step-size, CMA also can escape from a minimum and drift to another solution.



(a) $\boldsymbol{w}(0) = [1 \ \ 0]^T$

(b) $\boldsymbol{w}(0) = [-0.4 \ \ 0.05]^T$

(c) $\boldsymbol{w}(0) = [0.05 \ \ -0.4]^T$

FIGURE 12.38

CMA initialized at different points, $\mu = 0.016$, channel $H(z) = 1/(1 + 0.6z^{-1})$ in the absence of noise. Click on each caption to see animations on your browser.

Despite the name of the algorithm, CMA also works for non-constant modulus signals. However, as shown analytically in [80,190], its best performance occurs for constant-modulus signals since the variability in the transmitted signal modulus plays a role similar to the measurement noise in supervised algorithms.

The stochastic analysis of constant-modulus based algorithms is a hard task. Due to the nonlinearity and multimodality of the CM cost function, additional assumptions are necessary. A good review of the main results in the analysis of CMA can be found in [191], but many results were obtained after its publication. These results address different issues of constant-modulus based algorithms as the steady-state and transient behavior [77,80,190,192–196], convergence and stability [58,60,197–200], phase rotation [201–203], solutions for non-constant modulus signals [204–207], among others.

1.12.5.6 Subband and transform-domain adaptive filters

In transform-domain adaptive filters, the input signals are transformed (using the FFT, the DCT or other convenient transform), and the adaptation is performed in the transform domain. The gains are twofold: first, using block processing, one can obtain large reductions in computational cost. Second, by using the decorrelation properties of the transforms, it is possible to use different step-sizes for each tap in the transformed domain, thereby reducing the equivalent eigenvalue spread and increasing the convergence rate [208–216].

A related approach is that of subband adaptive filters, which are also useful to both reduce the computational cost and increase the convergence speed of long filters, in particular for acoustic echo. The main idea is to split the input signal in several parts using an *analysis filter bank*. Each part, relative to a particular band of the full spectrum, is processed separately, and finally all parts are joined together again by a *synthesis* filter bank [217]. Since each subsignal has a smaller bandwidth, the sampling rate can be reduced. The whole process must be made carefully to avoid as much as possible a deterioration of the signals due to aliasing, while still gaining in computational cost, in comparison with standard full-band filters [218–224]. It is important to point out that it is possible to avoid the introduction of extra delay (compared with a full-band filter) [225–228].

Good introductions to the subject can be found in [1,3,150].

1.12.5.7 Affine projections algorithm

The *affine projections* algorithm (APA) is an extension of NLMS, designed to obtain a convergence rate closer to that of RLS, but with a low computational cost, as close as possible to that of NLMS. We saw in Section 1.12.3.2 that NLMS can be derived as a method for choosing the step-size so that the current a posteriori estimation error is zeroed (12.130). The APA extends this idea, by canceling a vector of a posteriori errors, obtained from data from times $n-K$ to n, and the weight vector at time $n+1$:

$$\boldsymbol{\xi}(n) = \boldsymbol{d}(n) - \boldsymbol{\Phi}(n)\boldsymbol{w}^*(n+1), \tag{12.322}$$

in which

$$\boldsymbol{d}(n) = \begin{bmatrix} d(n) \\ d(n-1) \\ \vdots \\ d(n-K-1) \end{bmatrix}, \quad \boldsymbol{\Phi}(n) = \begin{bmatrix} \boldsymbol{\phi}^T(n) \\ \boldsymbol{\phi}^T(n-1) \\ \vdots \\ \boldsymbol{\phi}^T(n-K-1) \end{bmatrix}. \tag{12.323}$$

The affine projections algorithm chooses as next weight estimate the vector $\boldsymbol{w}(n+1)$ closest to the current estimate $\boldsymbol{w}(n)$ such that $\boldsymbol{\xi}(n) = \boldsymbol{0}$ [229–234]:

$$\boldsymbol{w}(n+1) = \arg \min_{\substack{\boldsymbol{w} \\ \text{s.t. } \boldsymbol{\xi}(n)=\boldsymbol{0}}} |\boldsymbol{w} - \boldsymbol{w}(n)|^2. \tag{12.324}$$

The solution to this problem is a recursion

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \boldsymbol{\Phi}^T(n)\left(\boldsymbol{\Phi}^*(n)\boldsymbol{\Phi}^T(n)\right)^{-1}\boldsymbol{e}(n), \tag{12.325}$$

where

$$\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{\Phi}(n)\boldsymbol{w}^*(n). \tag{12.326}$$

Note that (12.325) reduces to NLMS when $K = 1$. Similarly to what is done for NLMS, it is convenient to introduce a regularization term $\varepsilon \boldsymbol{I}$, with $\varepsilon > 0$ and a step-size $0 < \mu \leq 1$ in (12.325), leading to

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\boldsymbol{\Phi}^T(n)\left(\varepsilon\boldsymbol{I} + \boldsymbol{\Phi}^*(n)\boldsymbol{\Phi}^T(n)\right)^{-1}\boldsymbol{e}(n). \tag{12.327}$$

The advantage of this algorithm is that $\boldsymbol{\Phi}^*(n)\boldsymbol{\Phi}^T(n)$ is $K \times K$ (as opposed to the $M \times M$ matrix that must be recursively inverted for RLS), and the value of K does not need to be large, which keeps the computational cost low. In general, a value of two or three already gives a good improvement to the convergence rate even for regressors whose covariance matrix has large spreading factor.

In order to implement (12.327), it is useful to remember that computing the inverse explicitly is never a good idea: it is much better to solve the linear system

$$\left(\varepsilon\boldsymbol{I} + \boldsymbol{\Phi}^*(n)\boldsymbol{\Phi}^T(n)\right)\boldsymbol{a} = \boldsymbol{e}(n), \tag{12.328}$$

and then compute $\boldsymbol{\Phi}^T(n)\boldsymbol{a}$. From a numerical point of view, the best option would be to use the QR decomposition of $\boldsymbol{\Phi}(n)$ and avoid forming $\boldsymbol{\Phi}^*(n)\boldsymbol{\Phi}^T(n)$, but the computational cost would be larger. When the regressor is a tapped-delay line and K is large, one can use a fast version of APA, in which the solution of (12.328) is obtained in $\mathcal{O}(K)$ operations, instead of the usual $\mathcal{O}(K^3)$ [235–237].

Recent works focusing in the analysis of the APA are [238–241]. The problem of regularization (choice of ε) is considered in [242,243]. Applications to echo cancellation can be found in [244,245].

A related approach is the sliding-window RLS algorithm, a version of RLS in which only a finite memory is considered, instead of a forgetting factor as used in Section 1.12.3.3 [246,247].

1.12.5.8 Cooperative estimation

In this section, we briefly describe important advances in adaptive filtering through cooperative estimation. We first describe combinations of adaptive filters and in the sequel, we focus on distributed adaptive filtering.

1.12.5.8.1 Combinations of adaptive filters

Combinations of adaptive filters have received considerable attention lately, since they decrease the sensitivity of the filter to choices of parameters such as the step-size, forgetting factor or filter length. The idea is to combine the outputs of two (or several) different independently-run adaptive algorithms to achieve better performance than that of a single filter. In general, this approach is more robust than variable parameter schemes [190].

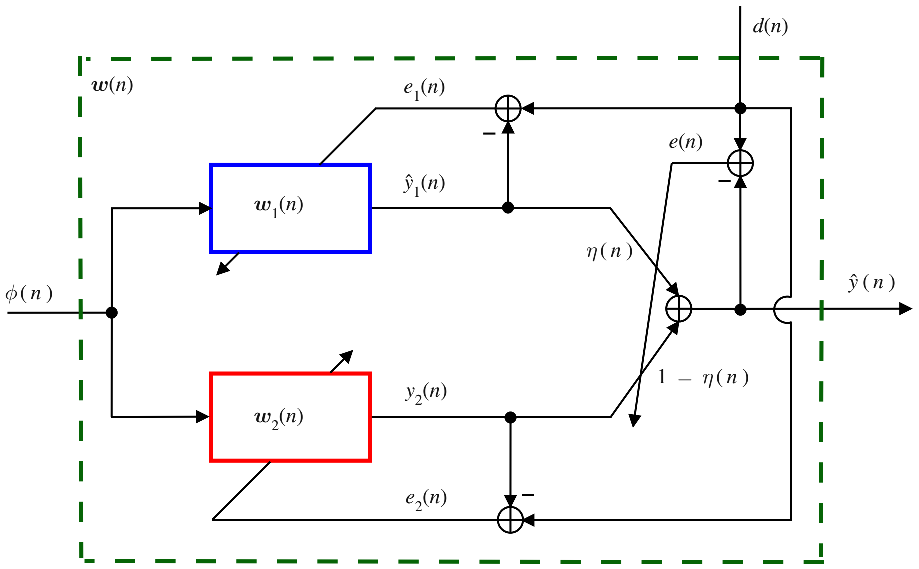


FIGURE 12.39

Convex combination of two transversal adaptive filters.

The first combined scheme that attracted attention was the convex combination of adaptive filters due to its relative simplicity and the proof that the optimum combination is universal, i.e., the combined estimate is at least as good as the best of the component filters in steady-state, for stationary inputs [248]. This scheme was proposed in [249] and further extended and analyzed in [248,250]. The original idea was to combine one fast and one slow LMS filter to obtain an overall filter with fast convergence and low misadjustment, but it was extended to other algorithms to take advantages of different characteristics, as explained below.

Figure 12.39 shows the convex combination of two adaptive filters, in which the output of the overall filter is computed as

$$\hat{y}(n) = \eta(n)\hat{y}_1(n) + [1-\eta(n)]\hat{y}_2(n), \tag{12.329}$$

where $\eta(n) \in [0,1]$ is the mixing parameter, $\hat{y}_i(n) = \boldsymbol{w}_i^H(n)\boldsymbol{\phi}(n), i = 1,2$, are the outputs of the transversal filters, $\boldsymbol{\phi}(n)$ is the common regressor vector, and $\boldsymbol{w}_i(n)$ are the weight vectors of the component filters. Note that the weight vector and the estimation error of the overall filter are given respectively by

$$\boldsymbol{w}(n) = \eta(n)\boldsymbol{w}_1(n) + [1-\eta(n)]\boldsymbol{w}_2(n) \tag{12.330}$$

and

$$e(n) = d(n) - \hat{y}(n) = \eta(n)e_1(n) + [1-\eta(n)]e_2(n), \tag{12.331}$$

where $e_i(n) = d(n) - \hat{y}_i(n)$ are the estimation errors of each component filter.

In order to restrict the mixing parameter to the interval [0, 1] and to reduce gradient noise when $\eta(n) \approx 0$ or $\eta(n) \approx 1$, a nonlinear transformation and an auxiliary variable $a(n)$ are used, i.e.,

$$\eta(n) = \text{sgm}[a(n)] \triangleq \frac{1}{1 + \mathrm{e}^{-a(n)}}, \tag{12.332}$$

where $a(n)$ is updated to minimize the overall squared error $e^2(n)$, i.e.,

$$a(n+1) = a(n) - \frac{\mu_a}{2}\nabla_a e^2(n) = a(n) + \mu_a\left[\hat{y}_1(n) - \hat{y}_2(n)\right]e(n)\eta(n)\left[1 - \eta(n)\right]. \tag{12.333}$$

In practice, $a(n)$ must be restricted by saturation of (12.333) to an interval $[-a_+, a_+]$, since the factor $\eta(n)\left[1 - \eta(n)\right]$ would virtually stop adaptation if $a(n)$ were allowed to grow too much. The correct adjustment of the step-size μ_a depends on the input signal and additive noise powers and also on the step-sizes of the component filters. To make the choice of μ_a independent of the filtering scenario, a normalized scheme was proposed in [251].

The combination of one fast and one slow LMS (with $\mu_1 > \mu_2$) operates in a very intuitive way. When fast changes appear, the fast filter outperforms the slow one, making $\eta(n) \approx 1$. In stationary situations, the slow filter gets a lower quadratic error and $\eta(n) \approx 0$, which allows the overall filter to achieve the misadjustment of the slow LMS filter. There are situations in which the combination outperforms each component filter and in these cases, $0 < \eta(n) < 1$. This behavior can be observed in the simulation results shown in Figure 12.40, where the convex combination of two LMS filters with different step-sizes ($\mu_1 = 0.1$ and $\mu_2 = 0.01$) was used to identity the system (from [248])

$$\begin{bmatrix} 0.9003 & -0.5377 & 0.2137 & -0.0280 & 0.7826 & 0.5242 & -0.0871 \end{bmatrix}.$$

The regressor $\boldsymbol{\phi}(n)$ is obtained from a process $x(n)$ generated with a first-order autoregressive model, whose transfer function is $\sqrt{1-b^2}/(1-bz^{-1})$. This model is fed with an iid Gaussian random process, whose variance is such that $\text{Tr}(\boldsymbol{R}_{\boldsymbol{\phi}}) = 1$. Moreover, additive iid noise $v(n)$ with variance $\sigma_v^2 = 0.01$ is added to form the desired signal. Figure 12.40a shows the EMSE curves estimated from the ensemble-average of 500 independent runs and filtered by a moving-average filter with 128 coefficients to facilitate the visualization. The combined filter acquires the faster convergence of μ_1-LMS and attains the EMSE of μ_2-LMS. Figure 12.40b shows the time evolution of the mixing parameter. We can observe that it rapidly changes toward $\text{sgm}[a_+]$ during the initial convergence, while its steady-state value is $1 - \text{sgm}[a_+]$.

After the publication of [249], many papers on combinations of adaptive filters appeared in the literature. Apparently, the idea of combining the outputs of several different independently-run adaptive algorithms was first proposed in [252], and later improved in [253,254]. The algorithms proposed in [252–254] are based on a Bayesian argument, and construct an overall (combined) filter through a linear combination of the outputs of several independent adaptive filters. The weights are the *a posteriori* probabilities that the underlying models used to describe each individual algorithm are "true." Since the weights add up to one, in a sense these first papers also proposed "convex" combinations of algorithms. Unconstrained linear combinations of adaptive filters were also proposed in [255]. However, the method of [248,249] has received more attention due to its relative simplicity and the proof that the optimum combination is universal. By this, we mean that if the combination uses at every instant the optimum

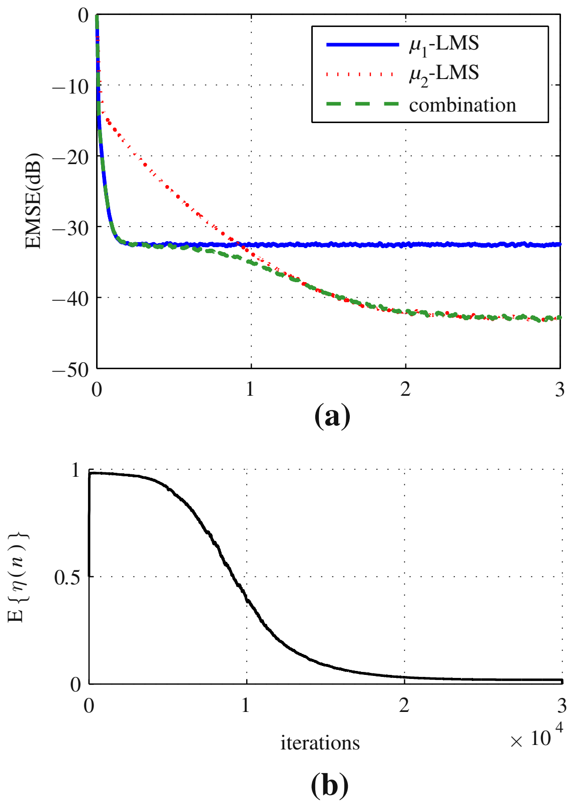


FIGURE 12.40

(a) EMSE for μ_1-LMS, μ_2-LMS, and their convex combination; (b) ensemble-average of $\eta(n)$; $\mu_1 = 0.1$, $\mu_2 = 0.01$, $\mu_a = 100$, $a^+ = 4$, $b = 0.8$; mean of 500 independent runs.

value of η, then the combined filter is always at least as good as the best component filter. However, in practice η must also be estimated, so the performance of the combination may be slightly worse than that of the best filter.

In the sequel, we briefly describe some of the most important recent contributions in this area.

In [250], the convergence of the overall filter is greatly improved by transferring a part of the fast filter $\boldsymbol{w}_1$ to the slow filter $\boldsymbol{w}_2$ when $\lambda(n) \geq \gamma$, that is,

$$\boldsymbol{w}_2(n+1) = \alpha \left[\boldsymbol{w}_2(n) + \mu_2 e_2(n)\boldsymbol{x}(n)\right] + (1-\alpha)\boldsymbol{w}_1(n), \tag{12.334}$$

where α is a parameter close to 1 and γ is a threshold close to the maximum value that can be reached by $\lambda(n)$.

A steady-state analysis of the combined filter was presented in [248]. This analysis shows that the convex combination is universal if the optimum η is used, a property that was exploited to design filters with improved tracking performance. Due to the nonlinear function (12.332), a transient analysis of the convex combination needs additional simplifications. This issue was addressed in [256] and [257], which provide models for the transient of the convex combination based on first and second-order Taylor series approximations.

The convex combination was used in [258] to improve the performance of a variable tap-length LMS algorithm in a low signal-to-noise environment (SNR $\leq$ 0 dB). The adaptation of the tap-length in the variable tap-length LMS algorithm is highly affected by the parameter choice and the noise level. Combination approaches improve such adaptation by exploiting advantages of parallel adaptive filters with different parameters.

The convex combination was exploited in [190] to improve the tracking performance of adaptive filters by combining filters with different tracking capabilities, as are the cases of LMS and RLS (see Section 1.12.4.4). The combination of algorithms of different families was also addressed in [259], where it was shown that a combination of two filters from the same family (i.e., two LMS or two RLS filters) cannot improve the performance over that of a single filter of the same type with optimal selection of the step-size (or forgetting factor). However, combining LMS and RLS filters, it is possible to simultaneously outperform the optimum LMS and RLS filters. In other words, combination schemes can achieve smaller errors than optimally adjusted individual filters.

Convex combinations were used for sparse echo cancellation in [260], considering the combination of two improved proportionate normalized least-mean-square (IPNLMS) filters. The combined scheme increases the IPNLMS robustness to channels with different degrees of sparsity and also alleviates the tradeoff between rate of convergence versus steady-state misadjustment imposed by the selection of the step-size. Still considering the echo cancellation application, [261] proposes a combination of adaptive Volterra kernels that presents a similar behavior to that of the complete Volterra filters, but with a reduction in the computational load. This scheme is robust regardless of the level of nonlinear distortion, which is a desired property for nonlinear echo cancellation.

In [262], the convex combination was also used as a scheme for adaptively biasing the weights of adaptive filters using an output multiplicative factor. Interpreting the biased estimator as the combination of the original filter and a filter with constant output equal to zero, practical schemes to adaptively adjust the multiplicative factor were proposed. This scheme provides a convenient bias versus variance tradeoff, leading to reductions in the filter mean-square error, especially in situations with a low signal-to-noise ratio.

Extending [248], the authors of [263] proposed an affine combination of two LMS algorithms, where the condition on the mixing parameter is relaxed, allowing it to be negative. Thus, this scheme can be interpreted as a generalization of the convex combination since the mixing parameter is not restricted to the interval [0, 1]. This approach allows for smaller EMSE in theory, but suffers from larger gradient noise in some situations. Under certain conditions, the optimum mixing parameter was proved to be negative in steady-state. Although the optimal linear combiner is unrealizable, two realizable algorithms were introduced. One is based on a stochastic gradient search and the other is based on the ratio of the average error powers from each individual adaptive filter. Under some circumstances, both algorithms present performance close to the optimum.

Similarly to the convex combination, the correct adjustment of the step-size for the updating of the mixing parameter in the affine combination (using the stochastic algorithm of [263]) depends on characteristics of the filtering scenario. This issue was addressed in [193], where a transient analysis for

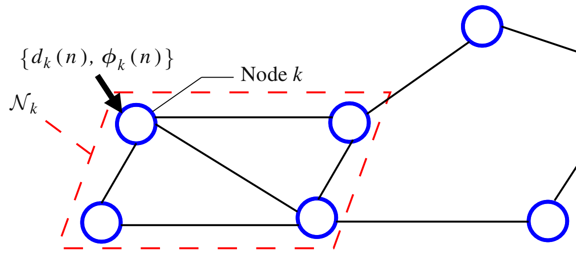


FIGURE 12.41

Diffusion network with $N = 7$ nodes: at time n, every node k takes a measurement $\{d_k(n), \boldsymbol{\phi}_k(n)\}$.

the affine combination of two adaptive filters was presented. The results of this analysis were used to facilitate the adjustment of the free parameters of the scheme and to propose two normalized algorithms to update the mixing parameter. Recently, the scheme based on the ratio of error powers of the two filters proposed in [263] to update the mixing parameter was analyzed in [264].

Finally, Kozat et al. [265] studied different mixing strategies in which the overall output is formed as the weighted linear combination (not necessarily constrained to convex or affine, as before) of the outputs of several component algorithms for stationary and certain nonstationary data.

1.12.5.8.2 ***Distributed adaptive filtering***

Adaptive networks use the information from data collected at nodes distributed over a certain region. Each node collects noisy observations related to a parameter of interest and interacts with its neighbors considering a certain network topology. The objective is to obtain an estimate of the parameter of interest as accurate as the one that would be obtained if each node had access to the information across the entire network [266,267]. Distributed estimation algorithms are useful in several contexts, including wireless and sensor networks, where scalability, robustness, and low power consumption are desirable. They also find applications in precision agriculture and environmental monitoring and transportation [268].

Figure 12.41 shows a network composed by N nodes distributed over some geographical area. At time instant n, every node k takes a measurement $\{d_k(n), \boldsymbol{\phi}_k(n)\}$ to estimate some parameter vector $\boldsymbol{w}_o$. The scalar measurement $d_k(n)$ represents the desired signal and $\boldsymbol{\phi}_k(n)$ denotes the length-M input regressor vector for node k. There are different solutions to the problem of estimating $\boldsymbol{w}_o$. In the *centralized* solution, every node transmits its data $\{d_k(n), \boldsymbol{\phi}_k(n)\}$ to a fusion center for processing. This solution is non-robust to failure in the fusion center. In *distributed* solutions, every node exchanges information with a subset of its neighboring nodes, and the processing is distributed among all nodes in the network. The set of nodes connected to node k (including k itself) is denoted by $\mathcal{N}_k$ and is called the neighborhood of node k.

There are three main network topologies used in distributed estimation: incremental, diffusion, and probabilistic diffusion [267]. In incremental networks, the nodes are connected so that information flows sequentially from node to node, in a cyclic manner (Figure 12.42a). In diffusion networks, each node communicates with its entire neighborhood at each instant (Figure 12.42b). Finally, in probabilistic

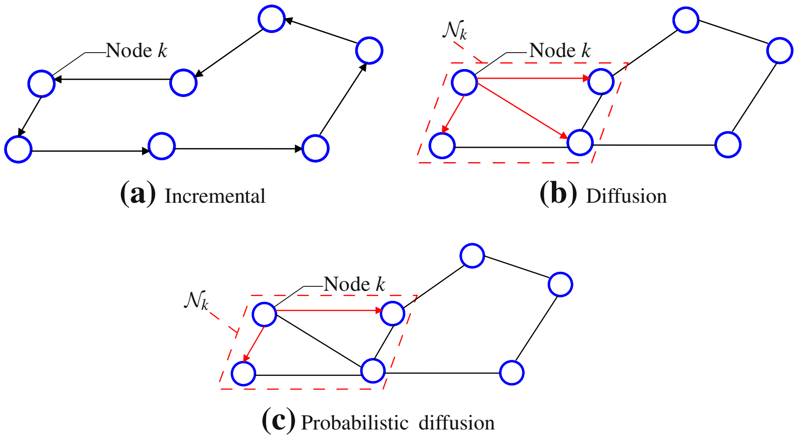


FIGURE 12.42

Different network topologies.

diffusion networks each node communicates with a (possibly random) subset of its neighbors at each instant (Figure 12.42c).

Several estimation algorithms have been proposed in the context of distributed adaptive filtering, such as incremental LMS, incremental RLS, diffusion LMS, diffusion RLS, diffusion Kalman filtering, and smoothing algorithms. In these algorithms, the nodes share information with their neighbors, and perform local adaptation and merging of information using convex combinations of available estimates. Distributed algorithms only require local communications between neighboring nodes, and can attain good estimation performance compared to centralized solutions. This is a current area of intense research [268–280].

Recently, adaptive distributed algorithms have also been used to model biological systems due to their the self-organization property. Examples include fish joining together in schools, bacterial motility, and bird flight formations [281–286].

1.12.5.9 Adaptive IIR filters

Adaptive infinite impulse response (IIR) filters represent an advantageous alternative in relation to adaptive FIR filters, due to their capacity of providing long impulse responses with a small number of coefficients. Their use may be desirable in applications requiring hundreds or thousands of taps, such as satellite-channel and mobile-radio equalizers. The advantages of adaptive IIR filters have a price: slow convergence, possible instability, error surface with local minima or biased global minimum [1,287–290]. In addition, not all long impulse responses will necessarily be well approximated by an IIR filter with low order.

The nonlinear relation between the adaptive-filter coefficients and the internal signals makes the gradient computation and convergence analysis more complicated than those of adaptive FIR filters [1]. Over the years, problems related to local minima, stability, and the effect of poles close to the unit

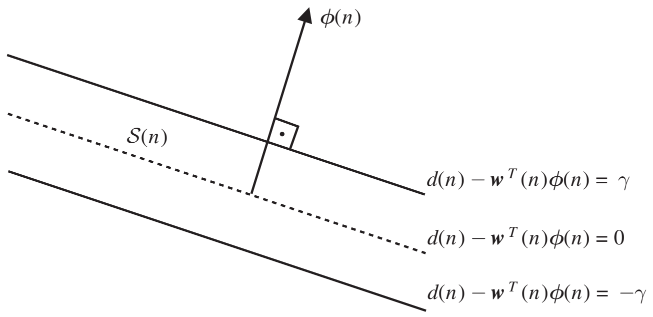

$\phi(n)$
$\mathcal{S}(n)$
$d(n) - \boldsymbol{w}^T(n)\phi(n) = \gamma$
$d(n) - \boldsymbol{w}^T(n)\phi(n) = 0$
$d(n) - \boldsymbol{w}^T(n)\phi(n) = -\gamma$

FIGURE 12.43

Constraint imposed by data at time *n*.

circle have been addressed by several authors leading to different adaptive algorithms and realization structures [291–300].

Another difficulty with adaptive IIR filters is that the performance of simple constant-gain algorithms may rapidly degrade as the order of the filter grows. Even in the absence of local minima, the adaptation of filters with order greater than two can remain almost stopped in regions where the mean square error is relatively high. This issue was addressed in [287], which was able to obtain a large acceleration in the convergence rate of IIR filters.

1.12.5.10 Set-membership and projections on convex sets

When the desired signal and the regressor are related as

$$d(n) = \boldsymbol{w}_o^H \boldsymbol{\phi}(n) + v(n),$$

where $v(n)$ is *bounded*, that is, $|v(n)| \leq \gamma$ for all n, it is natural to consider that each pair $\big(d(n), \boldsymbol{\phi}(n)\big)$ defines a region in which the true solution $\boldsymbol{w}_*$ must lie:

$$\boldsymbol{w}_o \in \mathcal{S}(n) = \left\{\boldsymbol{w} \in \mathcal{C}^M : \left|d(n) - \boldsymbol{w}^H \boldsymbol{\phi}(n)\right| \leq \gamma \right\}. \tag{12.335}$$

For example, if all vectors are two-dimensional and real, the region $\mathcal{S}(n)$ would be as in Figure 12.43. *Set-membership* algorithms aim to find the set of possible solutions, by tracking the intersection

$$\mathcal{T}(n) = \bigcap_{k=0}^{n} \mathcal{S}(k) \tag{12.336}$$

of all $\mathcal{S}(n)$ [301]. In the control literature and earlier algorithms, this is made by approximating $\mathcal{T}(n)$ by ellipsoids [301–303]. However, very good performance is obtained if one uses just any point contained in $\mathcal{T}(n)$ as a point estimate of $\boldsymbol{w}_o$ at each instant. In fact, it is more interesting to restrict the memory, so that the filter can track variations of $\boldsymbol{w}_\mathrm{o}$. In this case, only the last K constraint sets $\mathcal{S}(n)$ are used:

$$\mathcal{T}_K(n) = \bigcap_{k=n-K+1}^{n} \mathcal{S}(k). \tag{12.337}$$

This approach has a very nice property: close to steady-state, if the weight estimates are already inside $\mathcal{T}_K(n)$, it is not necessary to perform any update. This may reduce considerably the average complexity of the filter [304–310].

More recently, the approach has been extended to a more general setting, using projections onto convex sets and subgradients. This allows the extension of the results to non-differentiable cost functions, with different descriptions for the constraints (not restricted to (12.335)), also allowing the design of kernel adaptive filters [6,311–313] (an introduction to kernel adaptive filters can be found in [314]).

1.12.5.11 Adaptive filters with constraints

In some applications, one wishes to minimize a cost function given certain constraints: a typical case is beamforming. In the simplest example, an array of antennas or microphones receives a signal coming from a certain (known) direction, and interference from other (unknown) directions. The output of the array is a linear combination of the signals $\boldsymbol{\phi}_i(n)$ received at each antenna, e.g.,

$$\hat{y}(n) = w_0^*\boldsymbol{\phi}_0(n) + w_1^*\boldsymbol{\phi}_1(n) + \cdots + w_{M-1}^*\boldsymbol{\phi}_{M-1}(n).$$

One way of improving the signal-to-noise ratio at the output is to minimize the output power, under the restriction that the gain of the array in the direction of the desired signal is one. This is described by a constraint of the kind

$$\boldsymbol{c}^H\boldsymbol{w} = 1,$$

where $\boldsymbol{w} = [w_0 \ \dots \ w_{M-1}]$ is the weight vector, and $\boldsymbol{c} \in \mathcal{C}^M$ is a vector describing the constraint. The goal of the filter would then be to find the optimum $\boldsymbol{w}_\mathrm{o}$ that is the solution of

$$\boldsymbol{w}_\mathrm{o} = \arg \min_{\substack{w \\ \text{s.t. } \boldsymbol{c}^H\boldsymbol{w}=1}} \left|\hat{y}(n)\right|^2.$$

This solution is known as *minimum-variance distortionless response* (MVDR), or Capon beamformer, since it was originally proposed in [315]. There is an extensive literature on this subject [312,316–325]. Introductions to the topic can be found in [326–328].

1.12.5.12 Reduced-rank adaptive filters

In applications in which large filters are necessary, such as acoustic echo cancellation and nonlinear models using Volterra expansions, it may be useful to reduce the order of the problem, by working in a smaller subspace of the original problem. This reduces the number of parameters to be estimated, increasing the convergence speed and reducing the excess mean-square error [329]. Several methods have been proposed, such as [330,331], based on eigen-decompositions, the multi-stage Wiener filter of [332–334], the auxiliary vector filtering algorithm [335], and the joint iterative optimization method of [336–339]. This last method has the advantage of having low computational cost.

Box 8: [Fixed-point arithmetic]

In fixed-point arithmetic, all variables are restricted to lie in a fixed interval $[-a, a)$, and are stored as a sequence of B bits $b_0 b_1 \ldots b_{B-1}$. Assuming that two's complement arithmetic is used and that $a = 1$, then b_0 represents the signal of the variable and the sequence of bits represents the number

$$-b_0 + \sum_{m=1}^{B-1} b_m 2^{-m}.$$

Acknowledgment

The authors thank Prof. Wallace A. Martins (Federal University of Rio de Janeiro) for his careful reading and comments on the text.

Relevant Theory: Signal Processing Theory and Machine Learning

See this Volume, Chapter 3 Discrete-Time Signals and Systems
See this Volume, Chapter 4 Random Signals and Stochastic Processes
See this Volume, Chapter 6 Digital Filter Structures and Their Implementation
See this Volume, Chapter 11 Parametric Estimation
See this Volume, Chapter 25 A Tutorial on Model Selection

References

[1] P.S.R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation, third ed., Springer, 2008.
[2] S. Haykin, Adaptive Filter Theory, fourth ed., Prentice Hall, 2001.
[3] A.H. Sayed, Adaptive Filters, Wiley-IEEE Press, 2008.
[4] B. Widrow, S.D. Stearns, Adaptive Signal Processing, Prentice Hall, Englewood Cliffs, 1985.
[5] M.D. Miranda, M. Gerken, M.T.M. Silva, Efficient implementation of error-feedback LSL algorithm, Electron. Lett. 35 (16) (1999) 1308–1309.
[6] S. Theodoridis, K. Slavakis, I. Yamada, Adaptive learning in a world of projections, IEEE Signal Process. Mag. 28 (1) (2011) 97–123.
[7] B. Widrow, M. Hoff, Adaptive switching circuits, in: IRE WESCON Conv. Rec., pt. 4, 1960, pp. 96–104.
[8] B. Widrow, Thinking about thinking: the discovery of the LMS algorithm, IEEE Signal Process. Mag. 22 (1) (2005) 100–106.
[9] A.V. Oppenheim, A.S. Willsky, Signals and Systems, Prentice-Hall, 1996.
[10] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, 1987.
[11] R.A. Horn, C.R. Johnson, Topics in Matrix Analysis, Cambridge University Press, Cambridge, MA, 1991.
[12] A.J. Laub, Matrix Analysis for Scientists and Engineers, Society for Industrial and Applied Mathematics, PA, December 2005.
[13] C.D. Meyer, Matrix Analysis and Applied Linear Algebra, SIAM, Philadelphia, USA, 2000.
[14] G.H. Golub, C.F.V. Loan, Matrix Computations, third ed., Johns Hopkins, 1996.

[15] K.J. Åström, B. Wittenmark, Adaptive Control, Addison-Wesley, 1995.
[16] A. Leon-Garcia. Probability, Statistics, and Random Processes for Electrical Engineering, third ed., Prentice Hall, January 2008.
[17] T. Kailath, A.H. Sayed, B. Hassibi, Linear Estimation, Prentice Hall, NJ, 2000.
[18] P. Ioannou, B. Fidan, Adaptive Control Tutorial, SIAM, Society for Industrial and Applied Mathematics, December 2006.
[19] T. Gansler, J. Benesty, New insights into the stereophonic acoustic echo cancellation problem and an adaptive nonlinearity solution, IEEE Trans. Speech Audio Process. 10 (5) (2002) 257–267.
[20] H.I.K. Rao, B. Farhang-Boroujeny, Fast LMS/Newton algorithms for stereophonic acoustic echo cancelation, IEEE Trans. Signal Process. 57 (8) (2009) 2919–2930.
[21] C.H. Hansen, S.D. Snyder, Active Control of Noise and Vibration, Taylor & Francis, 1997.
[22] S.M. Kuo, D.R. Morgan, Active Noise Control Systems – Algorithms and DSP Implementations, Wiley Series in Telecommunications and Signal Processing, Wiley-Interscience, 1996.
[23] P.A. Nelson, S.J. Elliot, Active Control of Sound, Academic Press, 1992.
[24] S.D. Snyder, Active Noise Control Primer, Springer-Verlag, 2000.
[25] F.G. de Almeida Neto, V.H. Nascimento, M.T.M. Silva, Reduced-complexity widely linear adaptive estimation, in: 7th International Symposium on Wireless Communication Systems, September 2010, pp. 399–403.
[26] B. Picinbono, Wide-sense linear mean square estimation and prediction, in: International Conference on Acoustics, Speech, and Signal Processing, 1995, ICASSP-95, vol. 3, May 1995, pp. 2032–2035.
[27] B. Picinbono, P. Bondon, Second-order statistics of complex signals, IEEE Trans. Signal Process. 45 (2) (1997) 411–420.
[28] T. Adali, H. Li, R. Aloysius, On properties of the widely linear MSE filter and its LMS implementation, in: 43rd Annual Conference on Information Sciences and Systems, March 2009, pp. 876–881.
[29] A. Chinatto, C. Junqueira, J.M.T. Romano, Interference mitigation using widely linear arrays, in: 17th European Signal Processing Conference, September 2009, pp. 353–357.
[30] S.C. Douglas, D.P. Mandic, Performance analysis of the conventional complex LMS and augmented complex LMS algorithms, in: Proc. IEEE Int. Acoustics Speech and Signal Processing (ICASSP) Conf., 2010, pp. 3794–3797.
[31] D.P. Mandic, S. Still, S.C. Douglas, Duality between widely linear and dual channel adaptive filtering, in: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, ICASSP 2009, 2009, pp. 1729–1732.
[32] P.M. Djuric, J.H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M.F. Bugallo, J. Miguez, Particle filtering, IEEE Signal Process. Mag. 20 (5) (2003) 19–38.
[33] W.R. LePage, Complex Variables and the Laplace Transform for Engineers, Dover, 1980.
[34] K. Kreutz-Delgado, The complex gradient operator and the CR-Calculus, 0906.4835, June 2009.
[35] W.A. Sethares, The least mean square family, in: N. Kalouptsidis, S. Theodoridis (Eds.), Adaptive System Identification and Signal Processing Algorithms, Prentice Hall, 1993.
[36] V. Solo, Averaging analysis of the LMS algorithm, in: C.T. Leondes (Ed.), Control and Dynamic Systems, Stochastic Techniques in Digital Signal Processing Systems, Part 2 of 2, vol. 65, Academic Press, 1994, pp. 379–397.
[37] B.D.O. Anderson, R.R. Bitmead, C.R.J. Johnson Jr., P.V. Kokotovic, R.L. Kosut, I.M.Y. Mareels, L. Praly, B.D. Riedle, Stability of adaptive systems: passivity and averaging analysis, MIT Press, Cambridge, MA, 1986
[38] A.H. Sayed, T. Kailath, A state-space approach to adaptive RLS filtering, IEEE Signal Process. Mag. 11 (3) (1994) 18–60.
[39] H.J. Kushner, G.G. Yin, Stochastic Approximation Algorithms and Applications, Applications of Mathematics, Springer, 1997.

[40] P.A. Regalia, Numerical stability properties of a QR-based fast least squares algorithm, IEEE Trans. Signal Process. 41 (6) (1993) 2096–2109.

[41] D.T.M. Slock, The backward consistency concept and round-off error propagation dynamics in recursive least-squares filtering algorithms, Opt. Eng. 31 (1992) 1153–1169.

[42] G.E. Bottomley, S.T. Alexander, A novel approach for stabilizing recursive least squares filters, IEEE Trans. Signal Process. 39 (8) (1991) 1770–1779.

[43] J.A. Apolinário Jr. (Ed.), QRD-RLS Adaptive Filtering, Springer, NY, 2009.

[44] Y. Zakharov, G. White, J. Liu, Low-complexity RLS algorithms using dichotomous coordinate descent iterations, IEEE Trans. Signal Process. 56 (7) (2008) 3150–3161.

[45] V.H. Nascimento, J.C.M. Bermudez, Probability of divergence for the least-mean fourth (LMF) algorithm, IEEE Trans. Signal Process. 54 (4) (2006) 1376–1385.

[46] V.H. Nascimento, A.H. Sayed, On the learning mechanism of adaptive filters, IEEE Trans. Signal Process. 48 (6) (2000) 1609–1625.

[47] S. Chen, Low complexity concurrent constant modulus algorithm and soft directed scheme for blind equalization, IEE Proc. – Vis. Image Signal Process. 150 (2003) 312–320.

[48] H.J. Butterweck, Iterative analysis of the steady-state weight fluctuations in LMS-type adaptive filters, Technical Report EUT 96-E-299, Eindhoven University of Technology, Netherlands, June 1996.

[49] H.J. Butterweck, A wave theory of long adaptive filters, IEEE Trans. Circ. Syst. I 48 (6) (2001) 739–747.

[50] E. Eweda, O. Macchi, Convergence of an adaptive linear estimation algorithm, IEEE Trans. Autom. Control AC-19 (2) (1984) 119–127.

[51] O. Macchi, E. Eweda, Convergence analysis of self-adaptive equalizers, IEEE Trans. Inform. Theory IT-30 (2) (1984) 161–176.

[52] J.E. Mazo, On the independence theory of equalizer convergence, Bell Syst. Tech. J. 58 (1979) 963–993.

[53] V. Solo, X. Kong, Adaptive Signal Processing Algorithms, Prentice Hall, Englewood Cliffs, NJ, 1995.

[54] S.C. Douglas, W. Pan, Exact expectation analysis of the LMS adaptive filter, IEEE Trans. Signal Process. 43 (12) (1995) 2863–2871.

[55] A. Feuer, E. Weinstein, Convergence analysis of LMS filters with uncorrelated Gaussian data, IEEE Trans. Acoust. Speech Signal Process. ASSP-33 (1) (1985) 222–229.

[56] S. Florian, A. Feuer, Performance analysis of the LMS algorithm with a tapped delay line (two-dimensional case), IEEE Trans. Acoust. Speech Signal Process. ASSP-34 (6) (1986) 1542–1549.

[57] A.H. Sayed, V.H. Nascimento, Energy conservation and the learning ability of LMS adaptive filters, in: S. Haykin, B. Widrow (Eds.), Least-Mean-Square Adaptive Filters, Wiley, 2003.

[58] O. Dabeer, E. Masry, Convergence analysis of the constant modulus algorithm, IEEE Trans. Inform. Theory 49 (6) (2003) 1447–1464.

[59] P.I. Hübscher, J.C.M. Bermudez, V.H. Nascimento, A mean-square stability analysis of the least mean fourth (LMF) adaptive algorithm, IEEE Trans. Signal Process. 55 (8) (2007) 4018–4028.

[60] V.H. Nascimento, M.T.M. Silva, Stochastic stability analysis for the constant-modulus algorithm, IEEE Trans. Signal Process. 56 (10) (2008) 4984–4989.

[61] T.Y. Al-Naffouri, M. Moinuddin, Exact performance analysis of the ϵ-NLMS algorithm for colored circular Gaussian inputs, IEEE Trans. Signal Process. 58 (10) (2010) 5080–5090.

[62] T.Y. Al-Naffouri, M. Moinuddin, M.S. Sohail, Mean weight behavior of the NLMS algorithm for correlated Gaussian inputs, IEEE Signal Process. Lett. 18 (1) (2011) 7–10.

[63] N.J. Bershad, Analysis of the normalized LMS algorithm with Gaussian inputs, IEEE Trans. Acoust. Speech Signal Process. ASSP-34 (1986) 793–806.

[64] N.J. Bershad, Behavior of the ϵ-normalized LMS algorithm with Gaussian inputs, IEEE Trans. Acoust. Speech Signal Process. ASSP-35, May 1987, pp. 636–644.

[65] M.H. Costa, J.C.M. Bermudez, An improved model for the normalized LMS algorithm with Gaussian inputs and large number of coefficients, in: Proc. ICASSP 2002, vol. 2, pp. 1385–1388.

[66] V.H. Nascimento, A simple model for the effect of normalization on the convergence rate of adaptive filters, in: Proc. of the 2004 IEEE International Conference on Acoustics, Speech and, Signal Processing, 2004, pp. II-453–II-456.

[67] D.T.M. Slock, On the convergence behavior of the LMS and the normalized LMS algorithms, IEEE Trans. Signal Process. 41 (9) (1993) 2811–2825.

[68] M. Tarrab, A. Feuer, Convergence and performance analysis of the normalized LMS algorithm with uncorrelated Gaussian data, IEEE Trans. Inform. Theory 34 (4) (1988) 680–691.

[69] A. Carini, E. Mumolo, G.L. Sicuranza, V-vector algebra and its application to Volterra-adaptive filtering, IEEE Trans. Circ. Syst. II 46 (5) (1999) 585–598.

[70] F. Kuech, W. Kellermann, Orthogonalized power filters for nonlinear acoustic echo cancellation, Signal Process. 86 (6) (2006) 1168–1181.

[71] V.J. Mathews, G.L. Sicuranza, Polynomial Signal Processing, Wiley-Interscience, New York, 2000.

[72] A. Stenger, W. Kellermann, Adaptation of a memoryless preprocessor for nonlinear acoustic echo cancelling, Signal Process. 80 (9) (2000) 1747–1760.

[73] G.L. Sicuranza, A. Carini, A. Fermo, Nonlinear adaptive filters for acoustic echo cancellation in mobile terminals, in: K.E. Barner, G.R. Arce (Eds.), Nonlinear Signal and Image Processing: Theory, Methods, and Applications, The Electrical Engineering and Applied Signal Processing Series, CRC Press, 2003, pp. 223–255 (Chapter 7).

[74] E. Eleftheriou, D. Falconer, Tracking properties and steady-state performance of RLS adaptive filter algorithms, IEEE Trans. Acoust. Speech Signal Process. 34 (5) (1986) 1097–1110.

[75] G.V. Moustakides, Study of the transient phase of the forgetting factor RLS, IEEE Trans. Signal Process. 45 (10) (1997) 2468–2476.

[76] A.H. Sayed, M. Rupp, Error-energy bounds for adaptive gradient algorithms, IEEE Trans. Signal Process. 44 (8) (1996) 1982–1989.

[77] N. Yousef, A.H. Sayed, A unified approach to the steady-state and tracking analysis of adaptive filters, IEEE Trans. Signal Process. 49 (2) (2001) 314–324.

[78] M. Rupp, A.H. Sayed, A time-domain feedback analysis of filtered-error adaptive gradient algorithms, IEEE Trans. Signal Process. 44 (1996) 1428–1439.

[79] C. Samsom, V.U. Reddy, Fixed point error analysis of the normalized ladder algorithm, IEEE Trans. Acoust. Speech, Signal Process. 31 (5) (1983) 1177–1191.

[80] J.Y. Mai, A.H. Sayed, A feedback approach to the steady-state performance of fractionally spaced blind adaptive equalizers, IEEE Trans. Signal Process. 48 (1) (2000) 80–91.

[81] E. Eweda, Comparison of RLS, LMS, and sign algorithms for tracking randomly time-varying channels, IEEE Trans. Signal Process. 42 (11) (1994) 2937–2944.

[82] J.H. Wilkinson, Rounding Errors in Algebraic Processes, Her Majesty's Stationery Office, 1963.

[83] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, London, UK, 1965.

[84] S.T. Alexander, Transient weight misadjustment properties for the finite precision LMS algorithm, IEEE Trans. Acoust. Speech Signal Process. ASSP-35 (9) (1987) 1250–1258.

[85] S. Ardalan, Floating-point error analysis of recursive least-squares and least-mean-squares adaptive filters, IEEE Trans. Circ. Syst. 33 (12) (1986) 1192–1208.

[86] C. Caraiscos, B. Liu, A roundoff error analysis of the LMS adaptive algorithm, IEEE Trans. Acoust. Speech Signal Process. ASSP-32 (1) (1984) 34–41.

[87] R.D. Gitlin, J.E. Mazo, M.G. Taylor, On the design of gradient algorithms for digitally implemented adaptive filters, IEEE Trans. Circ. Theory, CT-20 (2) (1973) 125–136.

[88] J.C.M. Bermudez, N.J. Bershad, A nonlinear analytical model for the quantized LMS algorithm the arbitrary step size case, IEEE Trans. Signal Process. 44 (5) (1996) 1175–1183.

[89] J.C.M. Bermudez, N.J. Bershad, Transient and tracking performance analysis of the quantized LMS algorithm for time-varying system identification, IEEE Trans. Signal Process. 44 (8) (1996) 1990–1997.

[90] N.J. Bershad, J.C. Bermudez, New insights on the transient and steady-state behavior of the quantized LMS algorithm, IEEE Trans. Signal Process. 44 (10) (1996) 2623–2625.

[91] N.J. Bershad, J.C.M. Bermudez, A nonlinear analytical model for the quantized LMS algorithm-the power-of-two step size case, IEEE Trans. Signal Process. 44 (11) (1996) 2895–2900.

[92] J.M. Cioffi, J.J. Werner, The tap-drifting problem in digitally implemented data-driven echo cancellers, Bell Syst. Tech. J. 64 (1) (1985) 115–138.

[93] R.D. Gitlin, H.C. Meadors Jr., S.B. Weinstein, The tap-leakage algorithm: an algorithm for the stable operation of a digitally implemented, fractionally spaced adaptive equalizer, Bell Syst. Tech. J. 61 (8) (1982) 1817–1839.

[94] V.H. Nascimento, A.H. Sayed, Unbiased and stable leakage-based adaptive filters, IEEE Trans. Signal Process. 47 (12) (1999) 3261–3276.

[95] E. Eweda, Almost sure convergence of a decreasing gain sign algorithm for adaptive filtering, IEEE Trans. Acoust. Speech Signal Process. 36 (10) (1988) 1669–1671.

[96] E. Eweda, A tight upper bound of the average absolute error in a constant step-size sign algorithm, IEEE Trans. Acoust. Speech Signal Process. 37 (11) (1989) 1774–1776.

[97] E. Eweda, Convergence analysis of the sign algorithm without the independence and Gaussian assumptions, IEEE Trans. Signal Process. 48 (9) (2000) 2535–2544.

[98] A. Gersho, Adaptive filtering with binary reinforcement, IEEE Trans. Inform. Theory 30 (2) (1984) 191–199.

[99] P. Xue, B. Liu, Adaptive equalizer using finite-bit power-of-two quantizer, IEEE Trans. Acoust. Speech Signal Process. 34 (6) (1986) 1603–1611.

[100] S. Ljung, L. Ljung, Error propagation properties of recursive least-squares adaptation algorithms, Automatica 21 (2) (1985) 157–167.

[101] J. Cioffi, The fast adaptive ROTOR's RLS algorithm, IEEE Trans. Acoust. Speech Signal Process. 38 (4) (1990) 631–653.

[102] S. Alexander, A. Ghimikar, A method for recursive least squares filtering based upon an inverse QR decomposition, IEEE Trans. Signal Process. 41 (1) (1993) 20.

[103] S.T. Alexander, A.L. Ghimikar, A method for recursive least squares filtering based upon an inverse QR decomposition, IEEE Trans. Signal Process. 41 (1) (1993).

[104] J.A. Apolinário Jr., P. Diniz, A new fast QR algorithm based on a priori errors, IEEE Signal Process. Lett. 4 (11) (1997) 307–309.

[105] M.D. Miranda, L. Aguayo, M. Gerken. Performance of the a priori and a posteriori QR-LSL algorithms in a limited precision environment, in: Proc. IEEE Int Acoustics, Speech, and Signal Processing ICASSP-97 Conf., vol. 3, 1997, pp. 2337–2340.

[106] M.D. Miranda, M. Gerken, A hybrid QR-lattice least squares algorithm using a priori errors, in: Proc. 38th Midwest Symp. Circuits and Systems, Proceedings, vol. 2, 1995, pp. 983–986.

[107] M.D. Miranda, M. Gerken, A hybrid least squares QR-lattice algorithm using a priori errors, IEEE Trans. Signal Process. 45 (12) (1997) 2900–2911.

[108] I.K. Proudler, J.G. McWhirter, T.J. Shepherd, Computationally efficient QR decomposition approach to least squares adaptive filtering, IEE Proc. F Radar Signal Process. 138 (4) (1991) 341–353.

[109] P.A. Regalia, M.G. Bellanger, On the duality between fast QR methods and lattice methods in least-squares adaptive filtering, IEEE Trans. Signal Process. 39 (1991) 879–891.

[110] A. Rontogiannis, S. Theodoridis, On inverse factorization adaptive LS algorithms, Signal Process. 52 (1997) 35–47.

[111] A.A. Rontogiannis, S. Theodoridis, New fast inverse QR least squares adaptive algorithms, in: Proc. Int. Acoustics, Speech, and Signal Processing, vol. 2, 1995, pp. 1412–1415.

[112] A.A. Rontogiannis, S. Theodoridis, New multichannel fast QRD-LS adaptive algorithms, in: Proc. 13th Int. Conf. Digital Signal Processing, vol. 1, 1997, pp. 45–48.

[113] A.A. Rontogiannis, S. Theodoridis, Multichannel fast QRD-LS adaptive filtering: new technique and algorithms, IEEE Trans. Signal Process. 46 (11) (1998) 2862–2876.

[114] A.A. Rontogiannis, S. Theodoridis, New fast QR decomposition least squares adaptive algorithms, IEEE Trans. Signal Process. 46 (8) (1998) 2113–2121.

[115] M. Shoaib, S. Werner, J.A. Apolinário, Multichannel fast QR-decomposition algorithms: weight extraction method and its applications, IEEE Trans. Signal Process. 58 (1) (2010) 175–188.

[116] M. Shoaib, S. Werner, J.A. Apolinário, T.I. Laakso, Solution to the weight extraction problem in fast QR-decomposition rls algorithms, in: Proc. IEEE Int. Acoustics, Speech and Signal Processing Conf., ICASSP 2006, vol. 3, 2006.

[117] B. Friedlander, Lattice filters for adaptive processing, IEEE Trans. Signal Process. 70 (8) (1982) 829–867.

[118] D. Lee, M. Morf, B. Friedlander, Recursive least squares ladder estimation algorithms, IEEE Trans. Circ. Syst. 28 (6) (1981) 467–481.

[119] H. Lev-Ari, T. Kailath, J. Cioffi, Least-squares adaptive lattice and transversal filters: a unified geometric theory, IEEE Trans. Inform. Theory 30 (2) (1984) 222–236.

[120] F. Ling, D. Manolakis, J. Proakis, Numerically robust least-squares lattice-ladder algorithms with direct updating of the reflection coefficients, IEEE Trans. Acoust. Speech Signal Process. 34 (4) (1986) 837–845.

[121] V.J. Mathews, Z. Xie, Fixed-point error analysis of stochastic gradient adaptive lattice filters, IEEE Trans. Acoust. Speech Signal Process. 38 (1) (1990) 70–80.

[122] R.C. North, J.R. Zeidler, W.H. Ku, T.R. Albert, A floating-point arithmetic error analysis of direct and indirect coefficient updating techniques for adaptive lattice filters, IEEE Trans. Signal Process. 41 (5) (1993) 1809–1823.

[123] M.J. Reed, B. Liu, Analysis of simplified gradient adaptive lattice algorithms using power-of-two quantization, in: Proc. IEEE Int. Circuits and Systems Symp., 1990, pp. 792–795.

[124] C. Samson, V. Reddy, Fixed point error analysis of the normalized ladder algorithm, IEEE Trans. Acoust. Speech Signal Process. 31 (5) (1983) 1177–1191.

[125] R. Merched, Extended RLS lattice adaptive filters, IEEE Trans. Signal Process. 51 (9) (2003) 2294–2309.

[126] R. Merched, A.H. Sayed, Order-recursive RLS laguerre adaptive filtering, IEEE Trans. Signal Process. 48 (11) (2000) 3000–3010.

[127] R. Merched, A.H. Sayed, RLS-Laguerre lattice adaptive filtering: error-feedback, normalized, and array-based algorithms, IEEE Trans. Signal Process. 49 (11) (2001) 2565–2576.

[128] J.-L. Botto, G.V. Moustakides, Stabilizing the fast Kalman algorithms, IEEE Trans. Acoust. Speech Signal Process. 37 (9) (1989) 1342–1348.

[129] J. Cioffi, T. Kailath, Fast, recursive-least-squares transversal filters for adaptive filtering, IEEE Trans. Acoust. Speech Signal Process. 32 (2) (1984) 304–337.

[130] J. Cioffi, T. Kailath, Windowed fast transversal filters adaptive algorithms with normalization, IEEE Trans. Acoust. Speech Signal Process. 33 (3) (1985) 607–625.

[131] D. Falconer, L. Ljung, Application of fast Kalman estimation to adaptive equalization, IEEE Trans. Commun. 26 (10) (1978) 1439–1446.

[132] G.V. Moustakides, S. Theodorides, Fast Newton transversal filters a new class of adaptive estimation algorithms, IEEE Trans. Signal Process. 39 (10) (1991) 2184–2193.

[133] D.T.M. Slock, T. Kailath, Fast transversal filters with data sequence weighting, IEEE Trans. Acoust., Speech Signal Process. 37 (3) (1989) 346–359.

[134] D.T.M. Slock, T. Kailath, Numerically stable fast transversal filters for recursive least squares adaptive filtering, IEEE Trans. Signal Process. 39 (1) (1991) 92–114.

[135] Y. Zakharov, T. Tozer, Multiplication-free iterative algorithm for LS problem, Electron. Lett. 40 (9) (2004) 567–569.

[136] W.A. Sethares, D.A. Lawrence, C.R. Johnson Jr., R.R. Bitmead, Parameter drift in LMS adaptive filters, IEEE Trans. Acoust. Speech Signal Process. ASSP-34 (1986) 868–878.

[137] K. Mayyas, T. Aboulnasr, The leaky LMS algorithm: MSE analysis for Gaussian data, IEEE Trans. Signal Process. 45 (4) (1997) 927–934.

[138] C.S. Ludovico, J.C.M. Bermudez, A recursive least squares algorithm robust to low-power excitation, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004, Proceedings (ICASSP'04), 2004.

[139] J. Liu, Y. Zakharov, Low complexity dynamically regularised RLS algorithm, Electron. Lett. 44 (14) (2008) 885–886.

[140] M.C. Tsakiris, C.G. Lopes, V.H. Nascimento, An array recursive least-squares algorithm with generic non-fading regularization matrix, IEEE Signal Process. Lett. 17 (12) (2010) 1001–1004.

[141] N. Jablon, Adaptive beamforming with the generalized sidelobe canceller in the presence of array imperfections, IEEE Trans. Antennas Propag. 34 (8) (1986) 996–1012.

[142] J.C.M. Bermudez, M.H. Costa, Optimum leakage factor for the MOV-LMS algorithm in nonlinear modeling and control systems, in: Proc. IEEE Int Acoustics, Speech, and Signal Processing (ICASSP) Conf., vol. 2, 2002.

[143] E.J. Candès, M.B. Wakin, People hearing without listening: an introduction to compressive sampling, IEEE Signal Process. Mag. 25 (2) (2008) 21–30.

[144] D. Angelosante, J.A. Bazerque, G.B. Giannakis, Online adaptive estimation of sparse signals: where RLS meets the ℓ_1-norm, IEEE Trans. Signal Process. 58 (7) (2010) 3436–3447.

[145] D. Angelosante, G.B. Giannakis, RLS-weighted Lasso for adaptive estimation of sparse signals, in: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, ICASSP 2009, 2009, pp. 3245–3248.

[146] Y. Chen, Y. Gu, A.O. Hero, Sparse LMS for system identification, in: Proc. ICASSP 2009, April 2009, pp. 3125–3128.

[147] Y. Kopsinis, K. Slavakis, S. Theodoridis, Online sparse system identification and signal reconstruction using projections onto weighted ℓ_1 balls, IEEE Trans. Signal Process. 59 (3) (2011) 936–952.

[148] G. Mileounis, B. Babadi, N. Kalouptsidis, V. Tarokh, An adaptive greedy algorithm with application to nonlinear communications, IEEE Trans. Signal Process. 58 (6) (2010) 2998–3007.

[149] R. Harris, D. Chabries, F. Bishop, A variable step (VS) adaptive filter algorithm, IEEE Trans. Acoust. Speech Signal Process. 34 (2) (1986) 309–316.

[150] B. Farhang-Boroujeny, Adaptive Filters – Theory and Applications, John Wiley & Sons, West Sussex, 1998.

[151] J. Nagumo, A. Noda, A learning method for system identification, IEEE Trans. Autom. Control 12 (3) (1967) 282–287.

[152] A.E. Albert, L.S. Gardner Jr., Stochastic Approximation and Nonlinear Regression, MIT Press, 1967.

[153] T. Aboulnasr, K. Mayyas, A robust variable step-size LMS-type algorithm, IEEE Trans. Signal Process. 45 (3) (1997) 631–639.

[154] W.-P. Ang, B. Farhang-Boroujeny, A new class of gradient adaptive step-size LMS algorithms, IEEE Trans. Signal Process. 49 (4) (2001) 805–810.

[155] J. Benesty, H. Rey, L. Vega, S. Tressens, A nonparametric VSS NLMS algorithm, IEEE Signal Process. Lett. 13 (10) (2006) 581–584.

[156] N. Bershad, On the optimum gain parameter in LMS adaptation, IEEE Trans. Acoust. Speech Signal Process. 35 (7) (1987) 1065–1068.

[157] R.C. Bilcu, P. Kuosmanen, K. Egiazarian, A transform domain LMS adaptive filter with variable step-size, IEEE Signal Process. Lett. 9 (2) (2002) 51–53.
[158] J.B. Evans, P. Xue, B. Liu, Analysis and implementation of variable step size adaptive algorithms, IEEE Trans. Signal Process. 41 (8) (1993) 2517–2535.
[159] B. Farhang-Boroujeny, Variable-step-size LMS algorithm: new developments and experiments, IEE Proc. Vis. Image Signal Process. 141 (5) (1994) 311–317.
[160] A. Gupta, S. Joshi, Variable step-size LMS algorithm for fractal signals, IEEE Trans. Signal Process. 56 (4) (2008) 1411–1420.
[161] R.H. Kwong, E.W. Johnston, A variable step size LMS algorithm, IEEE Trans. Signal Process. 40 (7) (1992) 1633–1642.
[162] V.J. Mathews, Z. Xie, A stochastic gradient adaptive filter with gradient adaptive step size, IEEE Trans. Signal Process. 41 (6) (1993) 2075–2087.
[163] C. Paleologu, J. Benesty, S. Ciochina, A robust variable forgetting factor recursive least-squares algorithm for system identification, IEEE Signal Process. Lett. 15 (2008) 597–600.
[164] C. Paleologu, S. Ciochina, J. Benesty, Variable step-size NLMS algorithm for under-modeling acoustic echo cancellation, IEEE Signal Process. Lett. 15 (2008) 5–8.
[165] D.I. Pazaitis, A.G. Constantinides, A novel kurtosis driven variable step-size adaptive algorithm, IEEE Trans. Signal Process. 47 (3) (1999) 864–872.
[166] H.-C. Shin, A.H. Sayed, W.-J. Song, Variable step-size NLMS and affine projection algorithms, IEEE Signal Process. Lett. 11 (2) (2004) 132–135.
[167] L. Vega, H. Rey, J. Benesty, S. Tressens, A new robust variable step-size NLMS algorithm, IEEE Trans. Signal Process. 56 (5) (2008) 1878–1893.
[168] J. Benesty, S.L. Gay, An improved PNLMS algorithm, in: International Conference on Acoustics, Speech and Signal Processing, vol. 2, IEEE, 2002, pp. 1881–1884.
[169] F. das Chagas de Souza, O.J. Tobias, R. Seara, D.R. Morgan, A PNLMS algorithm with individual activation factors, IEEE Trans. Signal Process. 58 (4) (2010) 2036–2047.
[170] F. das Chagas de Souza, O.J. Tobias, R. Seara, D.R. Morgan, Stochastic model for the mean weight evolution of the IAF-PNLMS algorithm, IEEE Trans. Signal Process. 58 (11) (2010) 5895–5901.
[171] D.L. Duttweiler, Proportionate normalized least-mean-squares adaptation in echo cancellers, IEEE Trans. Speech Audio Process. 8 (5) (2000) 508–518.
[172] T. Gansler, S.L. Gay, M.M. Sondhi, J. Benesty, Double-talk robust fast converging algorithms for network echo cancellation, IEEE Trans. Speech Audio Process. 8 (6) (2000) 656–663.
[173] E. Walach, B. Widrow, The least mean fourth (LMF) adaptive algorithm and its family, IEEE Trans. Inform. Theory IT-30 (2) (1984) 275–283.
[174] R. Sharma, W.A. Sethares, J.A. Bucklew, Asymptotic analysis of stochastic gradient-based adaptive filtering algorithms with general cost functions, IEEE Trans. Signal Process. 44 (9) (1996) 2186–2194.
[175] T.Y. Al-Naffouri, A.H. Sayed, Transient analysis of adaptive filters with error nonlinearities, IEEE Trans. Signal Process. 51 (3) (2003) 653–663.
[176] J.A. Chambers, O. Tanrikulu, A.G. Constantinides, Least mean mixed-norm adaptive filtering, Electron. Lett. 30 (19) (1994).
[177] O. Tanrikulu, J.A. Chambers, Convergence and steady-state properties of the least-mean mixed-norm (LMMN) adaptive algorithm, IEEE Trans. Signal Process. 143 (3) (1996) 137–142.
[178] A.H. Sayed, Fundamentals of Adaptive Filtering, Wiley-Interscience, 2003.
[179] B. Hassibi, A.H. Sayed, T. Kailath, LMS is H^{∞}-optimal, in: Proc. Conference on Decision and Control, San Antonio, TX, vol. 1, December 1993, pp. 74–79.
[180] G. Aydin, O. Arikan, A.E. Cetin, Robust adaptive filtering algorithms for α-stable random processes, IEEE Trans. Circ. Syst. II 46 (2) (1999) 198–202.

[181] S.R. Kim, A. Efron, Adaptive robust impulse noise filtering, IEEE Trans. Signal Process. 43 (8) (1995) 1855–1866.

[182] E. Masry, Alpha-stable signals and adaptive filtering, IEEE Trans. Signal Process. 48 (11) (2000) 3011–3016.

[183] M. Shao, C.L. Nikias, Signal processing with fractional lower order moments: stable processes and their applications, Proc. IEEE 81 (7) (1993) 986–1010.

[184] D. Hirsch, W. Wolf, A simple adaptive equalizer for efficient data transmission, IEEE Trans. Commun. Technol. 18 (1) (1970) 5–12.

[185] A. Flaig, G.R. Arce, K.E. Barner, Affine order-statistic filters: medianization of linear FIR filters, IEEE Trans. Signal Process. 46 (8) (1998) 2101–2112.

[186] D.N. Godard, Self-recovering equalization and carrier tracking in two-dimensional data communication systems, IEEE Trans. Commun. 28 (11) (1980) 1867–1875.

[187] J.R. Treichler, B. Agee, A new approach to multipath correction of constant modulus signals, IEEE Trans. Acoust. Speech Signal Process. ASSP-28 (1983) 334–358.

[188] C.B. Papadias, D.T.M. Slock, Normalized sliding window constant modulus and decision-direct algorithms: a link between blind equalization and classical adaptive filtering, IEEE Trans. Signal Process. 45 (1997) 231–235.

[189] Z. Ding, Y. Li, Blind Equalization and Identification, Marcel Dekke, 2001.

[190] M.T.M. Silva, V.H. Nascimento, Improving the tracking capability of adaptive filters via convex combination, IEEE Trans. Signal Process. 56 (7) (2008) 3137–3149.

[191] C.R. Johnson Jr., P. Schniter, T.J. Endres, J.D. Behm, D.R. Brown, R.A. Casas, Blind equalization using the constant modulus criterion: a review, Proc. IEEE 86 (10) (1998) 1927–1950.

[192] R. Candido, M.T.M. Silva, M.D. Miranda, V.H. Nascimento, A statistical analysis of the dual-mode CMA, in: Proc. IEEE Int. Circuits and Systems (ISCAS) Symp., 2010, pp. 2510–2513.

[193] R. Candido, M.T.M. Silva, V.H. Nascimento, Transient and steady-state analysis of the affine combination of two adaptive filters, IEEE Trans. Signal Process. 58 (8) (2010) 4064–4078.

[194] I. Fijalkow, C.E. Manlove, C.R. Johnson Jr., Adaptive fractionally spaced blind CMA equalization: excess MSE, IEEE Trans. Signal Process. 46 (1) (1998) 227–231.

[195] M.T.M. Silva, M.D. Miranda, Tracking issues of some blind equalization algorithms, IEEE Signal Process. Lett. 11 (9) (2004) 760–763.

[196] M.T.M. Silva, V.H. Nascimento, Tracking analysis of the constant modulus algorithm, in: Proc., ICASSP 2008, Las Vegas, NV, USA, pp. 3561–3564.

[197] M.D. Miranda, M.T.M. Silva, V.H. Nascimento, Avoiding divergence in the constant modulus algorithm, in: Proc., ICASSP 2008, Las Vegas, NV, USA, 2008, pp. 3565–3568.

[198] M.D. Miranda, M.T.M. Silva, V.H. Nascimento, Avoiding divergence in the Shalvi-Weinstein algorithm, IEEE Trans. Signal Process. 56 (11) (2008) 5403–5413.

[199] P.A. Regalia, M. Mboup, Undermodeled equalization: a characterization of stationary points for a family of blind criteria, IEEE Trans. Signal Process. 47 (1999) 760–770.

[200] R. Suyama, R.R.F. Attux, J.M.T. Romano, M. Bellanger, On the relationship between least squares and constant modulus criteria for adaptive filtering, in: The 37th Asilomar Conference on Signals, Systems and Computers, vol. 2, 2003, pp. 1293–1297.

[201] J. Yang, J.-J. Werner, G.A. Dumont, The multimodulus blind equalization and its generalized algorithms, IEEE J. Sel. Areas Commun. 20 (2002) 997–1015.

[202] J.-T. Yuan, T.-C. Lin, Equalization and carrier phase recovery of CMA and MMA in blind adaptive receivers, IEEE Trans. Signal Process. 58 (6) (2010) 3206–3217.

[203] J.-T. Yuan, K.-D. Tsai, Analysis of the multimodulus blind equalization algorithm in QAM communication systems, IEEE Trans. Commun. 53 (2005) 1427–1431.

[204] S. Abrar, A. Nandi, Blind equalization of square-QAM signals: a multimodulus approach, IEEE Trans. Commun. 58 (6) (2010) 1674–1685.
[205] K. Banovic, E. Abdel-Raheem, M.A.S. Khalid, A novel radius-adjusted approach for blind adaptive equalization, IEEE Signal Process. Lett. 13 (2006) 37–40.
[206] J. Mendes Filho, M.T.M. Silva, M.D. Miranda, A family of algorithms for blind equalization of QAM signals, in: Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process, Prague, Czech Republic, 2011, pp. 3388–3391.
[207] J. Mendes Filho, M.T.M. Silva, M.D. Miranda, V.H. Nascimento, A region-based algorithm for blind equalization of QAM signals, in: Proc. of the IEEE/SP 15th Workshop on Statistical Signal Processing, Cardiff, UK, 2009, pp. 685–688.
[208] F. Beaufays, Transform-domain adaptive filters: an analytical approach, IEEE Trans. Signal Process. 43 (2) (1995) 422–431.
[209] G. Clark, S. Mitra, S. Parker, Block implementation of adaptive digital filters, IEEE Trans. Acoust. Speech Signal Process. 29 (3) (1981) 744–752.
[210] B. Farhang-Boroujeny, Analysis and efficient implementation of partitioned block LMS adaptive filters, IEEE Trans. Signal Process. 44 (11) (1996) 2865–2868.
[211] E. Ferrara, Fast implementations of LMS adaptive filters, IEEE Trans. Acoust. Speech Signal Process. 28 (4) (1980) 474–475.
[212] R. Merched, A.H. Sayed, An embedding approach to frequency-domain and subband adaptive filtering, IEEE Trans. Signal Process. 48 (9) (2000) 2607–2619.
[213] S. Narayan, A. Peterson, M. Narasimha, Transform domain LMS algorithm, IEEE Trans. Acoust. Speech Signal Process. 31 (3) (1983) 609–615.
[214] S.S. Narayan, A.M. Peterson, Frequency domain least-mean-square algorithm, Proc. IEEE 69 (1) (1981) 124–126.
[215] J.J. Shynk, Frequency-domain and multirate adaptive filtering, IEEE Signal Process. Mag. 9 (1) (1992) 14–37.
[216] J.-S. Soo, K.K. Pang, Multidelay block frequency domain adaptive filter, IEEE Trans. Acoust. Speech Signal Process. 38 (2) (1990) 373–376.
[217] P.P. Vaidyanathan, Multirate Systems and Filter Banks, Prentice Hall, Englewood Cliffs, NJ, 1993.
[218] I. Furukawa, A design of canceller of broadband acoustic echo, in: Proc. Int. Teleconference Symposium, Tokyo, Japan, 1984, pp. 1–8.
[219] A. Gilloire, M. Vetterli, Adaptive filtering in subbands with critical sampling: analysis, experiments, and application to acoustic echo cancellation, IEEE Trans. Signal Process. 40 (8) (1992) 1862–1875.
[220] W. Kellermann, Kompensation akustischer Echos in Frequenzteilbändern, Frequenz 39 (1985) 209–215.
[221] M.R. Petraglia, R.G. Alves, P.S.R. Diniz, New structures for adaptive filtering in subbands with critical sampling, IEEE Trans. Signal Process. 48 (12) (2000) 3316–3327.
[222] M.R. Petraglia, P.B. Batalheiro, Nonuniform subband adaptive filtering with critical sampling, IEEE Trans. Signal Process. 56 (2) (2008) 565–575.
[223] M.R. Petraglia, S.K. Mitra, Adaptive fir filter structure based on the generalized subband decomposition of fir filters, IEEE Trans. Circ. Syst. II 40 (6) (1993) 354–362.
[224] S. Weiss, R.W. Stewart, R. Rabenstein, Steady-state performance limitations of subband adaptive filters, IEEE Trans. Signal Process. 49 (9) (2001) 1982–1991.
[225] N. Hirayama, H. Sakai, S. Miyagi, Delayless subband adaptive filtering using the Hadamard transform, IEEE Trans. Signal Process. 47 (6) (1999) 1731–1734.
[226] R. Merched, M. Petraglia, P.S.R. Diniz, A delayless alias-free subband adaptive filter structure, IEEE Trans. Signal Process. 47 (6) (1999) 1580–1591.

[227] D.R. Morgan, J.C. Thi, A delayless subband adaptive filter architecture, IEEE Trans. Signal Process. 43 (8) (1995) 1819–1830.

[228] K. Nishikawa, H. Kiya, Conditions for convergence of a delayless subband adaptive filter and its efficient implementation, IEEE Trans. Signal Process. 46 (4) (1998) 1158–1167.

[229] J. Apolinário Jr., M.L.R. Campos, P.S.R. Diniz, Convergence analysis of the binormalized data-reusing LMS algorithm, IEEE Trans. Signal Process. 48 (11) (2000) 3235–3242.

[230] S.G. Kratzer, D.R. Morgan, The partial-rank algorithm for adaptive beamforming, in: Proc. SPIE, vol. 564, 1985, pp. 9–14.

[231] D.R. Morgan, S.G. Kratzer, On a class of computationally efficient, rapidly converging, generalized NLMS algorithms, IEEE Signal Process. Lett. 3 (8) (1996) 245–247.

[232] K. Ozeki, T. Umeda, An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties, Electron. Commun. Jpn. 67-A (1984) 19–27.

[233] S. Roy, J.J. Shynk, Analysis of the data-reusing LMS algorithm, in: Proc. 32nd Midwest Symp. Circuits and Systems, 1989, pp. 1127–1130.

[234] M. Rupp, A family of adaptive filter algorithms with decorrelating properties, IEEE Trans. Signal Process. 46 (3) (1998) 771–775.

[235] S.L. Gay, S. Tavathia, The fast affine projection algorithm, in: Proc. Int. Acoustics, Speech, and Signal Processing ICASSP-95 Conf., vol. 5, 1995, pp. 3023–3026.

[236] G. Rombouts, M. Moonen, A fast exact frequency domain implementation of the exponentially windowed affine projection algorithm, in: Proc. and Control Symp Adaptive Systems for Signal Processing, Communications 2000, AS-SPCC, The IEEE 2000, 2000, pp. 342–346.

[237] M. Tanaka, S. Makino, J. Kojima, A block exact fast affine projection algorithm, IEEE Trans. Speech Audio Process. 7 (1) (1999) 79–86.

[238] S.J.M. de Almeida, J.C.M. Bermudez, N.J. Bershad, A stochastic model for a pseudo affine projection algorithm, IEEE Trans. Signal Process. 57 (1) (2009) 107–118.

[239] S.J.M. de Almeida, J.C.M. Bermudez, N.J. Bershad, M.H. Costa, A statistical analysis of the affine projection algorithm for unity step size and autoregressive inputs, IEEE Trans. Circ. Syst. I 52 (7) (2005) 1394–1405.

[240] S.G. Sankaran, A.A.L. Beex, Convergence behavior of affine projection algorithms, IEEE Trans. Signal Process. 48 (4) (2000) 1086–1096.

[241] H.-C. Shin, A.H. Sayed, Mean-square performance of a family of affine projection algorithms, IEEE Trans. Signal Process. 52 (1) (2004) 90–102.

[242] C. Paleologu, J. Benesty, S. Ciochina, Regularization of the affine projection algorithm, IEEE Trans. Circ. Syst. II 58 (6) (2011) 366–370.

[243] H. Rey, L.R. Vega, S. Tressens, J. Benesty, Variable explicit regularization in affine projection algorithm: robustness issues and optimal choice, IEEE Trans. Signal Process. 55 (5) (2007) 2096–2109.

[244] J. Benesty, P. Duhamel, Y. Grenier, A multichannel affine projection algorithm with applications to multichannel acoustic echo cancellation, IEEE Signal Process. Lett. 3 (2) (1996) 35–37.

[245] C. Paleologu, J. Benesty, S. Ciochina, A variable step-size affine projection algorithm designed for acoustic echo cancellation, IEEE Trans. Audio Speech Lang. Process. 16 (8) (2008) 1466–1478.

[246] D. Manolakis, F. Ling, J. Proakis, Efficient time-recursive least-squares algorithms for finite-memory adaptive filtering, IEEE Trans. Circ. Syst. 34 (4) (1987) 400–408.

[247] K. Maouche, D.T.M. Slock, Performance analysis and FTF version of the generalized sliding window recursive least-squares (GSWRLS) algorithm, in: Proc. Conf. Signals, Systems and Computers Record of the 29th Asilomar Conf., vol. 1, 1995, pp. 685–689.

[248] J. Arenas-García, A.R. Figueiras-Vidal, A.H. Sayed, Mean-square performance of a convex combination of two adaptive filters, IEEE Trans. Signal Process. 54 (3) (2006) 1078–1090.

[249] M. Martinez-Ramon, J. Arenas-García, A. Navia-Vazquez, A.R. Figueiras-Vidal, An adaptive combination of adaptive filters for plant identification, in: 14th International Conference on Digital Signal Processing (DSP 2002), 2002, pp. 1195–1198.

[250] J. Arenas-García, M. Martinez-Ramón, A. Navia-Vázquez, A.R. Figueiras-Vidal, Plant identification via adaptive combination of transversal filters, Signal Process. 86 (2006) 2430–2438.

[251] L.A. Azpicueta-Ruiz, A.R. Figueiras-Vidal, J. Arenas-García, A normalized adaptation scheme for the convex combination of two adaptive filters, in: Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process., Las Vegas, NV, 2008, pp. 3301–3304.

[252] P. Andersson, Adaptive forgetting in recursive identification through multiple models, Int. J. Control 42 (1985) 1175–1193.

[253] M. Niedźwiecki, Identification of nonstationary stochastic systems using parallel estimation schemes, IEEE Trans. Autom. Control 35 (3) (1990) 329–334.

[254] M. Niedźwiecki, Multiple-model approach to finite memory adaptive filtering, IEEE Trans. Signal Process. 40 (2) (1992) 470–473.

[255] S.S. Kozat, A.C. Singer, Multi-stage adaptive signal processing algorithms, in: Proc. SAM Signal Process. Workshop, 2000, pp. 380–384.

[256] V.H. Nascimento, M.T. Silva, R. Candido, J. Arenas-García, A transient analysis for the convex combination of adaptive filters, in: Proc. IEEE Workshop on Statistical Signal Process., Cardiff, UK, 2009, pp. 53–56.

[257] M.T.M. Silva, V.H. Nascimento, J. Arenas-García, A transient analysis for the convex combination of two adaptive filters with transfer of coefficients, in: Proc., ICASSP 2010, Dallas, USA, pp. 3842–3845.

[258] Y. Zhang, J. Chambers, Convex combination of adaptive filters for a variable tap-length LMS algorithm, IEEE Signal Process. Lett. 13 (10) (2006) 628–631.

[259] V.H. Nascimento, M.T.M. Silva, L.A. Azpicueta-Ruiz, J. Arenas-García, On the tracking performance of combinations of least mean squares and recursive least squares adaptive filters, in: Proc. IEEE Int. Acoustics Speech and Signal Processing (ICASSP) Conf., 2010, pp. 3710–3713.

[260] J. Arenas-García, A.R. Figueiras-Vidal, Adaptive combination of proportionate filters for sparse echo cancellation, IEEE Trans. Audio Speech Lang. Process. 17 (2009) 1087–1098.

[261] L.A. Azpicueta-Ruiz, M. Zeller, A.R. Figueiras-Vidal, J. Arenas-García, W. Kellermann, Adaptive combination of Volterra kernels and its application to nonlinear acoustic echo cancellation, IEEE Trans. Acoust. Speech Signal Process. 19 (2011) 97–110.

[262] M. Lazaro-Gredilla, L.A. Azpicueta-Ruiz, A.R. Figueiras-Vidal, J. Arenas-García, Adaptively biasing the weights of adaptive filters, IEEE Trans. Signal Process. 58 (7) (2010) 3890–3895.

[263] N.J. Bershad, J.C.M. Bermudez, J.-Y. Tourneret, An affine combination of two LMS adaptive filters—transient mean-square analysis, IEEE Trans. Signal Process. 56 (5) (2008) 1853–1864.

[264] J.C.M. Bermudez, N.J. Bershad, J.-Y. Tourneret, Stochastic analysis of an error power ratio scheme applied to the affine combination of two LMS adaptive filters, Signal Process. 91 (11) (2011) 2615–2622.

[265] S.S. Kozat, A.T. Erdogan, A.C. Singer, A.H. Sayed, Steady-state MSE performance analysis of mixture approaches to adaptive filtering, IEEE Trans. Signal Process. 58 (8) (2010) 4050–4063.

[266] C.G. Lopes, A.H. Sayed, Incremental adaptive strategies over distributed networks, IEEE Trans. Signal Process. 55 (2007) 4064–4077.

[267] C.G. Lopes, A.H. Sayed, Diffusion least-mean squares over adaptive networks: formulation and performance analysis, IEEE Trans. Signal Process. 56 (7) (2008) 3122–3136.

[268] F.S. Cattivelli, A.H. Sayed, Diffusion LMS strategies for distributed estimation, IEEE Trans. Signal Process. 58 (3) (2010) 1035–1048.

[269] F.S. Cattivelli, C.G. Lopes, A.H. Sayed, Diffusion recursive least-squares for distributed estimation over adaptive networks, IEEE Trans. Signal Process. 56 (5) (2008) 1865–1877.

[270] F.S. Cattivelli, A.H. Sayed, Diffusion strategies for distributed Kalman filtering and smoothing, IEEE Trans. Autom. Control 55 (9) (2010) 2069–2084.

[271] F.S. Cattivelli, A.H. Sayed, Analysis of spatial and incremental LMS processing for distributed estimation, IEEE Trans. Signal Process. 59 (4) (2011) 1465–1480.

[272] F.S. Cattivelli, A.H. Sayed, Distributed detection over adaptive networks using diffusion adaptation, IEEE Trans. Signal Process. 59 (5) (2011) 1917–1932.

[273] R.L.G. Cavalcante, I. Yamada, B. Mulgrew, An adaptive projected subgradient approach to learning in diffusion networks, IEEE Trans. Signal Process. 57 (7) (2009) 2762–2774.

[274] S. Chouvardas, K. Slavakis, S. Theodoridis, Adaptive robust distributed learning in diffusion sensor networks, IEEE Trans. Signal Process. 59 (10) (2011) 4692–4707.

[275] L. Li, J. Chambers, A new incremental affine projection-based adaptive algorithm for distributed networks, Signal Process. 88 (10) (2008) 2599–2603.

[276] G. Mateos, I.D. Schizas, G.B. Giannakis, Closed-form MSE performance of the distributed LMS algorithm, in: Proc. IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop DSP/SPE 2009, 2009, pp. 66–71.

[277] G. Mateos, I.D. Schizas, G.B. Giannakis, Distributed recursive least-squares for consensus-based in-network adaptive estimation, IEEE Trans. Signal Process. 57 (11) (2009) 4583–4588.

[278] V.H. Nascimento, A.H. Sayed, Continuous-time distributed estimation, in: Proc. of the 45th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, November 2011, pp. 1–5

[279] I.D. Schizas, G. Mateos, G.B. Giannakis, Distributed LMS for consensus-based in-network adaptive processing, IEEE Trans. Signal Process. 57 (6) (2009) 2365–2382.

[280] N. Takahashi, I. Yamada, A.H. Sayed, Diffusion least-mean squares with adaptive combiners: formulation and performance analysis, IEEE Trans. Signal Process. 58 (9) (2010) 4795–4810.

[281] F.S. Cattivelli, A.H. Sayed, Modeling bird flight formations using diffusion adaptation, IEEE Trans. Signal Process. 59 (5) (2011) 2038–2051.

[282] J. Chen, A.H. Sayed, Bio-inspired cooperative optimization with application to bacteria motility, in: Proc. IEEE Int. Acoustics, Speech and Signal Processing (ICASSP) Conf., 2011, pp. 5788–5791.

[283] J. Chen, X. Zhao, A.H. Sayed, Bacterial motility via diffusion adaptation, in: Proc. Conf. Signals, Systems and Computers (ASILOMAR) Record of the 44th Asilomar Conf., 2010, pp. 1930–1934.

[284] S.-Y. Tu, A.H. Sayed, Foraging behavior of fish schools via diffusion adaptation, in: Proc. 2nd Int. Cognitive Information Processing (CIP), Workshop, 2010, pp. 63–68.

[285] S.-Y. Tu, A.H. Sayed, Tracking behavior of mobile adaptive networks, in: Proc. Conf. Signals, Systems and Computers (ASILOMAR) Record of the 44th Asilomar Conf., 2010, pp. 698–702.

[286] S.-Y. Tu, A.H. Sayed, Cooperative prey herding based on diffusion adaptation, in: Proc. IEEE Int. Acoustics, Speech and Signal Processing (ICASSP) Conf., 2011, pp. 3752–3755.

[287] P.M.S. Burt, P. Regalia, A new framework for convergence analysis and algorithm development of adaptive IIR filters, IEEE Trans. Signal Process. 53 (2005) 3129–3140.

[288] S.L. Netto, P.S.R. Diniz, P. Agathoklis, Adaptive IIR filtering algorithms for system identification: a general framework, IEEE Trans. Educ. 38 (1) (1995) 54–66.

[289] P.A. Regalia, Adaptive IIR Filtering in Signal Processing and Control, Marcel Dekker, 1995.

[290] J.J. Shynk, Adaptive IIR filtering, IEEE ASSP Mag. 6 (2) (1989) 4–21.

[291] P.M.S. Burt, M. Gerken, A polyphase IIR adaptive filter: error surface analysis and application, in: Proc. IEEE Int. Conf. Acoustics, Speech, and, Signal Process., vol. 3, April 1997, pp. 2285–2288.

[292] P.M.S. Burt, M. Gerken, A polyphase IIR adaptive filter, IEEE Trans. Circ. Syst. II 49 (5) (2002) 356–359.

[293] J.E. Cousseau, P.S. Diniz, A general consistent equation-error algorithm for adaptive IIR filtering, Signal Process. 56 (2) (1997) 121–134.

[294] J.E. Cousseau, P.S.R. Diniz, New adaptive IIR filtering algorithms based on the Steiglitz-McBride method, IEEE Trans. Signal Process. 45 (5) (1997) 1367–1371.
[295] P.S.R. Diniz, J.E. Cousseau, A. Antoniou, Fast parallel realization of IIR adaptive filters, IEEE Trans. Circ. Syst. II 41 (8) (1994) 561–567.
[296] H. Fan, A structural view of asymptotic convergence speed of adaptive IIR filtering algorithms. I. Infinite precision implementation, IEEE Trans. Signal Process. 41 (4) (1993) 1493–1517.
[297] C.R. Johnson Jr., Adaptive IIR filtering: current results and open issues, IEEE Trans. Inform. Theory 30 (2) (1984) 237–250.
[298] R. Lopez-Valcarce, F. Perez-Gonzalez, Adaptive lattice IIR filtering revisited: convergence issues and new algorithms with improved stability properties, IEEE Trans. Signal Process. 49 (4) (2001) 811–821.
[299] P.A. Regalia, Stable and efficient lattice algorithms for adaptive IIR filtering, IEEE Trans. Signal Process. 40 (2) (1992) 375–388.
[300] S. Stearns, Error surfaces of recursive adaptive filters, IEEE Trans. Acoust. Speech Signal Process. 29 (3) (1981) 763–766.
[301] D.P. Bertsekas, I.B. Rhodes, Recursive state estimation for a set-membership description of uncertainty, IEEE Trans. Autom. Control AC-16 (2) (1971) 117–128.
[302] S. Dasgupta, Y.-F. Huang, Asymptotically convergent modified recursive least-squares with data-dependent updating and forgetting factor for systems with bounded noise, IEEE Trans. Inform. Theory 33 (3) (1987) 383–392.
[303] J.R. Deller Jr., M. Nayeri, S.F. Odeh, Least-square identification with error bounds for real-time signal processing and control, Proc. IEEE 81 (6) (1993) 815–849.
[304] T. Aboulnasr, K. Mayyas, Complexity reduction of the NLMS algorithm via selective coefficient update, IEEE Trans. Signal Process. 47 (5) (1999) 1421–1424.
[305] R.C. de Lamare, P. Diniz, Set-membership adaptive algorithms based on time-varying error bounds for CDMA interference suppression, IEEE Trans. Veh. Technol. 58 (2) (2009) 644–654.
[306] P.S.R. Diniz, S. Werner, Set-membership binormalized data-reusing LMS algorithms, IEEE Trans. Signal Process. 51 (1) (2003) 124–134.
[307] K. Dogancay, O. Tanrikulu, Adaptive filtering algorithms with selective partial updates, IEEE Trans. Circ. Syst. II 48 (8) (2001) 762–769.
[308] S.C. Douglas, Adaptive filters employing partial updates, IEEE Trans. Circ. Syst. II 44 (3) (1997) 209–216.
[309] S. Werner, J.A. Apolinário, M.L. de Campos, P.S. Diniz, Low-complexity constrained affine-projection algorithms, IEEE Trans. Signal Process. 53 (12) (2005) 4545–4555.
[310] S. Werner, M.L.R. de Campos, P.S.R. Diniz, Partial-update NLMS algorithms with data-selective updating, IEEE Trans. Signal Process. 52 (4) (2004) 938–949.
[311] K. Slavakis, S. Theodoridis, I. Yamada, Online kernel-based classification using adaptive projection algorithms, IEEE Trans. Signal Process. 56 (7) (2008) 2781–2796.
[312] K. Slavakis, S. Theodoridis, I. Yamada, Adaptive constrained learning in reproducing kernel hilbert spaces: the robust beamforming case, IEEE Trans. Signal Process. 57 (12) (2009) 4744–4764.
[313] I. Yamada, K. Slavakis, K. Yamada, An efficient robust adaptive filtering algorithm based on parallel subgradient projection techniques, IEEE Trans. Signal Process. 50 (5) (2002) 1091–1101.
[314] W. Liu, J. Principe, S. Haykin, Kernel Adaptive Filtering: A Comprehensive Introduction, Wiley Publishing, 2010.
[315] J. Capon, High-resolution frequency-wavenumber spectrum analysis, Proc. IEEE 57 (8) (1969) 1408–1418.
[316] S. Applebaum, Adaptive arrays, IEEE Trans. Antennas Propag. 24 (5) (1976) 585–598.
[317] N. Bornhorst, M. Pesavento, A.B. Gershman, Distributed beamforming for multi-group multicasting relay networks, IEEE Trans. Signal Process. 60 (1) (2012) 221–232.

[318] H. Cox, R. Zeskind, M. Owen, Robust adaptive beamforming, IEEE Trans. Acoust. Speech, Signal Process. 35 (10) (1987) 1365–1376.

[319] O.L. Frost III, An algorithm for linearly constrained adaptive array processing, Proc. IEEE 60 (8) (1972) 926–935.

[320] G.-O. Glentis, K. Berberidis, S. Theodoridis, Efficient least squares adaptive algorithms for FIR transversal filtering, IEEE Signal Process. Mag. 16 (4) (1999) 13–41.

[321] L. Lei, J.P. Lie, A.B. Gershman, C.M.S. See, Robust adaptive beamforming in partly calibrated sparse sensor arrays, IEEE Trans. Signal Process. 58 (3) (2010) 1661–1667.

[322] B.D. Van Veen, An analysis of several partially adaptive beamformer designs, IEEE Trans. Acoust. Speech Signal Process. 37 (2) (1989) 192–203.

[323] B.D. van Veen, Minimum variance beamforming with soft response constraints, IEEE Trans. Signal Process. 39 (9) (1991) 1964–1972.

[324] S.A. Vorobyov, H. Chen, A.B. Gershman, On the relationship between robust minimum variance beamformers with probabilistic and worst-case distortionless response constraints, IEEE Trans. Signal Process. 56 (11) (2008) 5719–5724.

[325] S.A. Vorobyov, A.B. Gershman, Z.-Q. Luo, N. Ma, Adaptive beamforming with joint robustness against mismatched signal steering vector and interference nonstationarity, IEEE Signal Process. Lett. 11 (2) (2004) 108–111.

[326] H. Krim, M. Viberg, Two decades of array signal processing research: the parametric approach, IEEE Signal Process. Mag. 13 (4) (1996) 67–94.

[327] H.L.V. Trees, Optimum Array Processing, Wiley, 2002.

[328] B.D. Van Veen, K.M. Buckley, Beamforming: a versatile approach to spatial filtering, IEEE ASSP Mag. 5 (2) (1988) 4–24.

[329] L. Scharf, D. Tufts, Rank reduction for modeling stationary signals, IEEE Trans. Acoust. Speech Signal Process. 35 (3) (1987) 350–355.

[330] A.M. Haimovich, Y. Bar-Ness, An eigenanalysis interference canceler, IEEE Trans. Signal Process. 39 (1) (1991) 76–84.

[331] Y. Song, S. Roy, Blind adaptive reduced-rank detection for DS-CDMA signals in multipath channels, IEEE J. Sel. Areas Commun. 17 (11) (1999) 1960–1970.

[332] J.S. Goldstein, I.S. Reed, L.L. Scharf, A multistage representation of the Wiener filter based on orthogonal projections, IEEE Trans. Inform. Theory 44 (7) (1998) 2943–2959.

[333] M.L. Honig, J.S. Goldstein, Adaptive reduced-rank interference suppression based on the multistage Wiener filter, IEEE Trans. Commun. 50 (6) (2002) 986–994.

[334] L.L. Scharf, E.K.P. Chong, M.D. Zoltowski, J.S. Goldstein, I.S. Reed, Subspace expansion and the equivalence of conjugate direction and multistage Wiener filters, IEEE Trans. Signal Process. 56 (10) (2008) 5013–5019.

[335] D.A. Pados, G.N. Karystinos, An iterative algorithm for the computation of the MVDR filter, IEEE Trans. Signal Process. 49 (2) (2001) 290–300.

[336] R.C. de Lamare, R. Sampaio-Neto, Reduced-rank interference suppression for DS-CDMA based on interpolated fir filters, IEEE Commun. Lett. 9 (3) (2005) 213–215.

[337] R.C. de Lamare, R. Sampaio-Neto, Reduced-rank adaptive filtering based on joint iterative optimization of adaptive filters, IEEE Signal Process. Lett. 14 (12) (2007) 980–983.

[338] R.C. de Lamare, R. Sampaio-Neto, Adaptive reduced-rank processing based on joint and iterative interpolation, decimation, and filtering, IEEE Trans. Signal Process. 57 (7) (2009) 2503–2514.

[339] M. Yukawa, R. de Lamare, R. Sampaio-Neto, Efficient acoustic echo cancellation with reduced-rank adaptive filtering based on selective decimation and adaptive interpolation, IEEE Trans. Speech Audio Process. 16 (4) (2008) 696–710.