Chapter 7
Adaptive Audio Processing

7.1 Introduction

Adaptive filters in audio scenarios have a wide range of applications that, in general, can be reduced to only two main families:

- *Direct acoustic modeling*: room impulse response estimation, acoustic echo cancellation, parameters determination of complex acoustic dynamical systems, methods for computational analysis of the acoustic scene, ...
- *Inverse acoustic modeling*: equalization of transduction devices such as loudspeakers and microphones, multipoint acoustic room correction, active noise cancellation, systems calibration for 3D audio, crosstalk elimination, ...

In other words, from complex PA systems to low-cost noise-canceling headphones, modern audio systems are almost always equipped with "intelligent" calibration systems. In this ever-changing environment, adaptive signal processing is increasingly becoming an indispensable theoretical and applicative tool in modern DASP.

7.2 Adaptation with Stochastic and Approximate Stocastic Optimization

Adaptive Filters (AFs) are defined as information processing systems capable of autonomously adjusting their parameters in response to external stimuli. In other words, the system learns independently and adapts its parameters to achieve a certain processing goal such as extracting the useful information from an acquired signal and the removal of disturbances due to noise or other sources interfering or, more generally, the adaptive filter provides the elimination of the redundant information. A linear transversal AF, illustrated in Fig. 7.1, is a simple FIR filter equipped with an adaptation or learning algorithm for determining its parameters according to a predefined criterion that consists in minimizing a certain *cost-* or *loss-function* [1]-[11].

For a compact representation, it is possible to use a vector notation as in Eqn. (4.3). We define the weight vector, containing the coefficients of the filter impulse

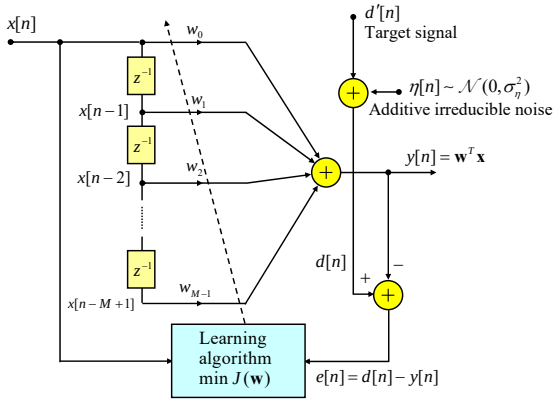


Fig. 7.1 A linear adaptive FIR filter or simply adaptive filter (AF) composed by convolver circuit, i.e. an adaptive linear combiner feeded by a delay-line, plus a learning algorithm that minimize a given cost function (CF) $J(\mathbf{w})$. Note that the desired signal $d[n]$, is a "target sequence" with added Gaussian noise.

response at time n, as $\mathbf{w}_n \in \mathbb{R}^{M\times 1}$. The input signal is defined as $\mathbf{x}_n \in \mathbb{R}^{M\times 1} = [x[n]\ x[n-1]\ \cdots\ x[n-M+1]]^T$, which contains the window signal along the input delay line of the filter. However, in the absence of the index n, is worth $\mathbf{x} \to \mathbf{x}_n$ and $\mathbf{w} \to \mathbf{w}_n$. Thus, as in Eqn. (4.3), we can write the convolution as the inner (or dot) product between input and weight vectors $y[n] = \mathbf{x}^T\mathbf{w} = \mathbf{w}^T\mathbf{x}$.

Hence, indicating by $\Delta\mathbf{w}_n$ the coefficients' variation (calculated according to some law described below) at time defined by the index n, we can write the adaptation rule as $\mathbf{w}_n = \mathbf{w}_{n-1} + \Delta\mathbf{w}_n$.

Let $d[n]$ be the *reference signal*, also denoted as *target* or *desired signal*, the weight vector $\mathbf{w}$ is determined by minimizing a certain *cost function* (CF) or *performance function* of the *a priori* error that is defined as

$$e[n] = d[n] - y[n] = d[n] - \mathbf{w}^T\mathbf{x}. \tag{7.1}$$

If the stochastic nature of the input signals $\mathbf{x}$ and $d[n]$ are *a priori* known, the CF is some statistic function of the error signal. In these cases it is usual to consider the *statistical expectation*[1] of the squared error. Such quantity indicate as *mean-squared error* (MSE) is defined as

$$J(\mathbf{w}) = \mathbb{E}\{|e[n]|^2\}. \tag{7.2}$$

The minimization of Eqn. (7.2) is used to determine a closed form solution of a given problem, by a stochastic optimization criterion called *minimum mean squared error* (MMSE).

If the statistic of the input signals are unknown it is possible to proceed numerically starting from an empirical estimate of the MSE. In this case, we more properly speak of *sum of squared error* (SSE) and the CF can be written as

$$J(\mathbf{w}) = \hat{\mathbb{E}}\{|e[n]|^2\} = \frac{1}{N}\sum_n |e[n]|^2. \tag{7.3}$$

[1] If X is a *random variable* with *probability density function* $p_X(x)$, its *expected vaule* is defined as $\mathbb{E}(X) = \int_{-\infty}^{\infty} xp(x)dx$.

The minimization of Eqn. (7.3) is performed by the class of *least squares* (LS) algorithms. The LS criterion can be considered as an approximation of the stochastic MSE criterion where the expectation operator, in practice, is replaced directly by an empirical time average operator. Therefore is the following approximation is considered $\mathbb{E}\{|e[n]|^2\} \approx \hat{\mathbb{E}}\{|e[n]|^2\}$. If this approximation is true the stochastic process $e[n]$ is defined as an *ergodic process*.

A schematic representation of the learning paradigms based on the error minimization is illustrated in Fig. 7.2.

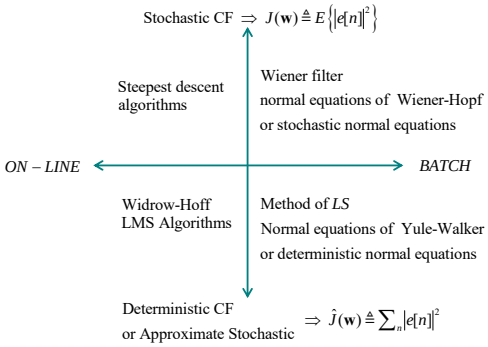


Fig. 7.2 Schematic representation of adaptation with stochastic vs. approximate stochastic optimization and the batch vs. online learning algorithms. (Courtesy of [11]).

While the stochast approach it is intrinsically a batch method, the approximate stochastic optimization algorithms can be derived in a recursive or non-recursive or batch formulation. In batch, formulation the fundamental hypothesis is to know the entire signals (i.e. the training set), or a portion, acquired by direct, usually noisy, measures. In these cases, when it is possible to consider ergodic and stationary input processes, the expectation can be replaced with its time average calculated over N signal samples.

7.2.1 Normal Equations in the Discrete Wiener-Hopf Notation

In the Wiener's optimal filtering theory [1]-[3], the performance function is the expectation of the squared error. From the error definition in (7.1) we can write

$$J(\mathbf{w}) = \mathbb{E}\left\{e^2[n]\right\} = \mathbb{E}\left\{d^2[n] - 2\mathbf{w}^T\mathbf{x}_n d[n] + \mathbf{w}^T\mathbf{x}_n\mathbf{x}_n^T\mathbf{w}\right\} \tag{7.4}$$

then, if $\mathbf{x}_n$ is a zero-mean stationary stochastic process its covariance/correlation matrix is defined $\mathbf{R} \in \mathbb{R}^{M\times M} = \mathbb{E}\{\mathbf{x}_n\mathbf{x}_n^T\}$. Again, if $d[n]$ is a zero-mean stationary stochastic process, the cross-correlation vector between $\mathbf{x}_n$ and $d[n]$ is defined as $\mathbf{g} \in \mathbb{R}^{M\times 1} = \mathbb{E}\{\mathbf{x}_n^T d[n]\}$, so the above equation can be rewritten as

$$J(\mathbf{w}) = \mathbb{E}\left\{d^2[n]\right\} - 2\mathbf{w}^T\mathbf{g} + \mathbf{w}^T\mathbf{R}\mathbf{w} \tag{7.5}$$

that is a quadratic function of the tap-weight vector $\mathbf{w}$ with a single global minimum. The Wiener's solution can be found by solving the following optimization problem

$$\mathbf{w}_{\text{opt}} \therefore \underset{\mathbf{w}\in\mathbb{R}^{M\times 1}}{\arg\min} \{J(\mathbf{w})\}. \tag{7.6}$$

The gradient vector of the performance function (7.5) can be written as

$$\nabla J(\mathbf{w}) \in \mathbb{R}^{M\times 1} = \frac{\partial\left[\mathbb{E}\left\{d^2[n]\right\} - 2\mathbf{w}^T\mathbf{g} + \mathbf{w}^T\mathbf{R}\mathbf{w}\right]}{\partial\mathbf{w}} = 2(\mathbf{R}\mathbf{w} - \mathbf{g}) \tag{7.7}$$

while the second derivative, i.e. the Hessian matrix, is

$$\nabla^2 J(\mathbf{w}) \in \mathbb{R}^{M\times M} = \frac{\partial^2\left[\mathbb{E}\left\{d^2[n]\right\} - 2\mathbf{w}^T\mathbf{g} + \mathbf{w}^T\mathbf{R}\mathbf{w}\right]}{\partial\mathbf{w}^2} = \mathbf{R}. \tag{7.8}$$

Thus, as $\nabla J(\mathbf{w}) \to \mathbf{0}$, the optimal solution of Eqn. (7.5), can be obtained solving the so-called *Wiener-Hopf normal equations*

$$\mathbf{R}\mathbf{w} = \mathbf{g} \tag{7.9}$$

that is

$$\mathbf{w}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{g}. \tag{7.10}$$

Moreover, replacing $\mathbf{w}$ by $\mathbf{w}_{\text{opt}}$ and $\mathbf{R}\mathbf{w}_{\text{opt}}$ by $\mathbf{g}$ in Eqn. (7.5) we obtain *minimum error energy* that can be written in the following equivalent forms

$$\begin{aligned} J_{\min}(\mathbf{w}) &= \mathbb{E}\left\{d^2[n]\right\} - \mathbf{w}_{\text{opt}}^T\mathbf{g} \\ &= \mathbb{E}\left\{d^2[n]\right\} - \mathbf{w}_{\text{opt}}^T\mathbf{R}\mathbf{w}_{\text{opt}} \\ &= \mathbb{E}\left\{d^2[n]\right\} - \mathbf{g}^T\mathbf{R}^{-1}\mathbf{g}. \end{aligned} \tag{7.11}$$

Property 7.1. Note that, let $\mathbf{h}$ be a "target model", i.e. such that $d[n] = \mathbf{h}^T\mathbf{x} + \eta[n]$, in the case of independent noise $\eta[n]$ with variance σ_η^2, at the optimal solution (i.e. for $\mathbf{w}_{\text{opt}} = \mathbf{h}$) we have that

$$\mathbb{E}\{d^2[n]\} = \mathbb{E}\{(\mathbf{w}_{\text{opt}}^T\mathbf{x} + \eta[n])^2\} = \sigma_\eta^2 + \mathbf{w}_{\text{opt}}^T\mathbf{R}\mathbf{w}_{\text{opt}}.$$

So, from the Eqn. (7.11), the minimum error energy is equal to

$$J_{\min}(\mathbf{w}) = \sigma_\eta^2 \tag{7.12}$$

which is the reason why $\eta[n]$ is referred to as *irreducible noise* and σ_η^2 represents the *lower bound* for the linear adaptive filter learning algorithm performance.

Remark 7.1. In the Wiener's optimal filtering theory, the filter's inputs are considered as a stochastic processes (SPs) described in terms of their *a priori* known second-order statistics (i.e. its correlation matrix and its cross-correlation vector). The vector of the filter weights is considered as deterministic unknown and the calculation of optimal filter coefficients $\mathbf{w}_{\text{opt}}$, is made minimizing the statistical CF defined

by the MSE (7.4). However, note that many authors (see e.g. [6]-[11]) define the adaptive filter (AF), the filter whose parameters are iteratively adjusted based on the new signal samples that gradually flows to its input.

7.2.2 Normal Equations in the Yule-Walker Notation

Although the Wiener theory has great relevance in obtaining results in a closed form, in adaptive filtering the optimal solution is determined by minimizing a deterministic CF $\hat{\mathbb{E}}\{e^2[n]\}$, where $\hat{\mathbb{E}}$ is an empirical approximation of the error expectation (7.2).

In the LS method, the characteristic of the data-block (batch or mini-batch), defined by the N-length window, is determined considering the nature of the problem. The following conventions are assumed:

- the measurement interval, also denoted analysis window, is limited: $n \in [0, N-1]$;
- the signal is zero outside the analysis window.

So, explicitly writing the error (7.1) at time instants 0, 1 , ..., $N-1$, we have that

$$\begin{aligned} e[0] &= d[0] - \mathbf{x}_0^T \mathbf{w} \\ e[1] &= d[1] - \mathbf{x}_1^T \mathbf{w} \\ &\vdots \\ e[N-1] &= d[N-1] - \mathbf{x}_{N-1}^T \mathbf{w}. \end{aligned} \tag{7.13}$$

The set of equations (7.13) can be written in compact form as

$$\mathbf{e} = \mathbf{d} - \mathbf{X}\mathbf{w} \tag{7.14}$$

denoted as *ordinary linear least squares* (OLS) or simply *least squares* (LS) model, where $\mathbf{X} \in \mathbb{R}^{N \times M}$ is defined as *data matrix* that contains all available data, $\mathbf{d} \in \mathbb{R}^{N \times 1}$ is the desired signal vector and $\mathbf{e} \in \mathbb{R}^{N \times 1}$ is the error vector

Moreover, the CF (7.3) can written as

$$\begin{aligned} J(\mathbf{w}) &= \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} (d_i - \mathbf{x}_i^T \mathbf{w})^2 = \mathbf{e}^T \mathbf{e} \\ &= [\mathbf{d} - \mathbf{X}\mathbf{w}]^T [\mathbf{d} - \mathbf{X}\mathbf{w}] \end{aligned} \tag{7.15}$$

developing the product, we get the following quadratic form

$$J(\mathbf{w}) = \mathbf{d}^T \mathbf{d} - 2\mathbf{w}^T \mathbf{X}^T \mathbf{d} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w}. \tag{7.16}$$

For the gradient of $J(\mathbf{w})$, we have that

$$\nabla J(\mathbf{w}) \triangleq \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{X}^T \mathbf{d} + 2\mathbf{X}^T \mathbf{X}\mathbf{w} \tag{7.17}$$

and the minimum of the LF is when the gradient is null $\nabla J(\mathbf{w}) \to 0$; i.e. for

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{d}. \tag{7.18}$$

Note that, the previous expression are the *Yule-Walker normal equations*. Thus, solving for $\mathbf{w}$, we obtain the LS solution defined as

$$\mathbf{w}_{\text{LS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{d}. \tag{7.19}$$

Remark 7.2. Note that from statistical interpretation, comparing expressions of the Wiener and LS quadratic CFs, we have that $\mathbf{R}_{xx} \in \mathbb{R}^{M\times M} = \mathbf{X}^T\mathbf{X}$ is an empirical estimation of the covariance matrix $\mathbf{R}$; and that $\mathbf{R}_{xd} \in \mathbb{R}^{M\times 1} = \mathbf{X}^T\mathbf{d}$ is an empirical estimation of the cross-covariance vector $\mathbf{g}$. Moreover, the OLS is a *maximum likelihood estimation* (MLE).

In addition, due to its nature, the matrix $\mathbf{X}^T\mathbf{X}$ may be ill-conditioned and the equations system has no solution, and in this case the problem is *hill-posed*, A simple "trick", in order to have reliable and "more regular" solution, and also to avoid numerical errors, it is prefered to solve the system of equations by adding a diagonal matrix such as

$$\mathbf{w}_{\text{LS}} = (\mathbf{X}^T\mathbf{X} + \delta\mathbf{I})^{-1}\mathbf{X}^T\mathbf{d} \tag{7.20}$$

where δ is a "small" constant defined as *regularization parameter*.

Remark 7.3. Observe that for $N > M$, the term $\mathbf{X}^{\#} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, is the generalized inverse, also called pseudo-inverse or Moore–Penrose inverse, of the matrix $\mathbf{X}$. In other terms, the LS solution is the solution of a linear equation system: $\mathbf{w}_{\text{LS}} = \mathbf{X}^{\#}\mathbf{d}$.

7.3 First Order Adaptive Algorithm

The simple adaptive method, developed in 1959 by Widrow-Hoff [5], denoted as *Widrow-Hoff delta rule* or *Least Mean Squares* (LMS) algorithm, can be derived considering the most simple approximation of the CF (7.2).

7.3.1 Least Mean Squares Algorithm

In order to introduce the LMS, consider a simple online adaptation procedure where the recursive estimator has a form of $\mathbf{w}_k = \mathbf{w}_{k-1} + \Delta\mathbf{w}_k$, where $k = 0,\ 1\ ,\ ...;$ is the iteration index and where the adaptation parameter is defined as $\Delta\mathbf{w}_k \propto -\nabla J(\mathbf{w})$. So, the adaptation rule can be written as

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \tfrac{1}{2}\mu\{-\nabla J(\mathbf{w}_k)\}, \quad \text{for} \quad k = 0, 1, ... \tag{7.21}$$

where μ is a "small constant" defined as *step size* or *learning rate*, and $\frac{1}{2}$ has been introduced for a further simplification.

Thus, let's define $\mathbf{w}_{-1}$ the initial condition (i.c.), i.e. starting from a "small" values random configuration (of a given distribution), the estimated results is obtained after a "sufficient" number of iterations.

Table 7.1 LMS algorithm for adaptive filtering

```
Given input and desired sequences [x[n] d[n]]_{n=1}^{N}
Initialization w, μ
Delay-line definition x_n ← 0
    x_n = [x[n]; x_n(1 : M − 1)];   // DL update: shift and load a new input sample
    y[n] = w^T x_n;   // compute the filter output
    e[n] = d[n] − y[n];   // compute the a priori error
    w = w + μ · e[n] · x_n;   // LMS adaptation rule
}
```

In LMS in order to determine the "best" AF weigts we minimize directly the square of the error $e^2[n]$ in other words, in LMS the CF is defined as

$$J(\mathbf{w}) = e^2[n]. \tag{7.22}$$

The gradient of (7.22), evaluated at the index k assume the form

$$\nabla J(\mathbf{w}) \triangleq \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -2(d_k - \mathbf{w}^T \mathbf{x}_k)\mathbf{x}_k \tag{7.23}$$

and, as the error is: $e_k = d_k - \mathbf{w}^T \mathbf{x}_k$, we can write

$$\nabla J(\mathbf{w}) = -e_k \mathbf{x}_k \tag{7.24}$$

finally, the recursive formula (7.21) for each adaptation step, can be written as

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \mu e_k \mathbf{x}_k, \quad \text{for} \quad k = 0, 1, 2, \dots . \tag{7.25}$$

This algorithm is the well known *least mean squares* (LMS) and is also known as the Widrow-Hoff learning rule [5] that can be formulated as reported in Table 7.3.1

Remark 7.4. Note that, in adaptive algorithms the CF's gradient is not *a priori* known and the quantity $\nabla J(\mathbf{w})$ represents a local estimate of the true gradient vector. This class is referred to as *sthocastic gradient algorithms* (SGA) and the most widespread family derived from that class, is the LMS algorithm (7.25), that, in addition can be also derived from the recursive solution of the normal equations of Yule-Walker.

7.3.2 Adaptive Algorithm Convergence Analysis

Adaptive algorithms as the LMS in (7.25), generalized as $\mathbf{w}_k = \mathbf{w}_{k-1} + f(e_k, \mathbf{x}_k)$, can be considered as a dynamical systems and their performance analysis can be evaluated based on the fundamental aspects of: *stability*, *convergence speed*, *error at steady state*, *transient behavior*, and *tracking capability*; expressed in terms of opportune statistical

functions of the error signal. Here, for brevity, we will deal only with the first three aspects: *stability*, *convergence speed* and *error at steady state*.

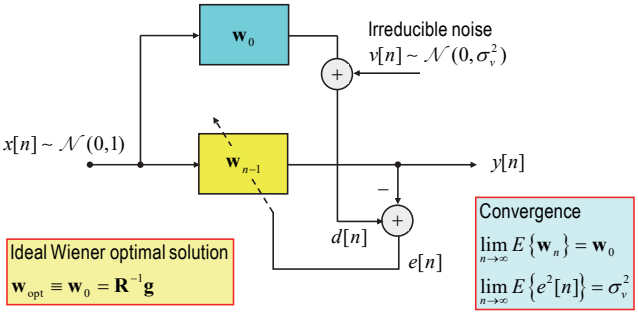


Fig. 7.3 The performance analysis model is an identification model used for the statistical analysis of the performance of adaptive algorithms. Ideally, for a high number of iterations, the adaptive filter weights tend on average to the true model, while the error variance tends to the variance of the irreducible noise.

For both theoretical and experimental statistical analysis of adaptive algorithms performance, is considered a simple *performance analysis model*, that consists in a simple identification problem as illustrated in Fig. 7.3. The weights $\mathbf{w}_0$ represents the target model (usually random generated and for tracking capability measure are time-variant), while the irreducible noise is a zero-mean Gaussian noise with *a priori* known variance. The input $x[n]$ is usually white or colored noise with unitary variance.

7.3.2.1 LMS Weak-Convergence Statistical Analysis

From Eqn. (7.25), subtracting the target $\mathbf{w}_0 = \mathbf{w}_{\text{opt}}$ from both members, and defining the *error vector* as $\mathbf{u}_n = \mathbf{w}_n - \mathbf{w}_0$, the adaptation rule can be written as

$$\mathbf{u}_n = \mathbf{u}_{n-1} + \mu e[n]\mathbf{x}_n. \tag{7.26}$$

Let $v[n] = d[n] - \mathbf{w}_0^T\mathbf{x}_n$, be the *error at optimal solution*, we can express the *a priori* error as $e[n] = v[n] - \mathbf{u}_{n-1}\mathbf{x}_n$, that substitute in (7.26) allows to write

$$\mathbf{u}_n = (\mathbf{I} - \mu\mathbf{x}_n\mathbf{x}_n^T)\mathbf{u}_{n-1} + \mu v[n]\mathbf{x}_n \tag{7.27}$$

where the quantity $\mathbf{u}_n$, $v[n]$ and $\mathbf{x}_n$ can be considered discrete stochastic processes (SP), and for this reason the Eqn. (7.27) can be considered as time-varying *stochastic difference equation* (SDE). The forcing term $\mu v[n]\mathbf{x}_n$ is due to irreducible noise $v[n]$ whose causes are due to the measurement error, the quantization effects and other noise sources.

Remark 7.5. The determination of the statistical solution of a SDE is very complex because it requires the calculation of both sides first and second order moments. For example, taking the expectation of (7.27) we note the presence of third-order moment $\mathbb{E}\{\mathbf{x}_n\mathbf{x}_n^T\mathbf{u}_{n-1}\}$. This causes some mathematical-statistical difficulties and, for this reason, is preferred to refer the assumption of independence[2].

[2] For independence is $\mathbb{E}\{\mathbf{u}_n\mathbf{v}_n\} = \mathbb{E}\{\mathbf{u}_n\}\mathbb{E}\{\mathbf{v}_n\}$.

A first simplification for the study can be made by considering a very small adaptation step size $\mu \ll 1$ such that term appearing in (7.27) can be simplified. With this condition the approximation $(\mathbf{I}-\mu\mathbf{x}_n\mathbf{x}_n^H) \sim (\mathbf{I}-\mu\mathbf{R})$ holds, so Eqn. (7.27) can be rewritten as

$$\mathbf{u}_n = (\mathbf{I}-\mu\mathbf{R})\mathbf{u}_{n-1} + \mu v[n]\mathbf{x}_n.$$

A second simplification for the statistical study of LMS convergence can be done by the *Direct-Averaging Method* (DAM)) [14], i.e. considering the (7.27) as a normal ordinary difference equation (ODE) respect to the averaged quantity. Thus, taking the expectation of both side of previous equation we can analyze the first order solution. For the independence of the quantity $\mathbf{x}_n$ and $v[n]$, we have $\mathbb{E}\{\mu v[n]\mathbf{x}_n\} = 0$, so

$$\mathbb{E}\{\mathbf{u}_n\} = (\mathbf{I}-\mu\mathbf{R})\mathbb{E}\{\mathbf{u}_{n-1}\}.$$

Decomposing the correlation matrix $\mathbf{R}$ with the unitary similarity transformation $\mathbf{\Lambda} = \mathbf{Q}^T\mathbf{R}\mathbf{Q}$, posing $\hat{\mathbf{u}}_n = \mathbf{Q}^T\mathbf{u}_n$, we can write

$$\mathbb{E}\{\hat{\mathbf{u}}_n\} = (\mathbf{I}-\mu\mathbf{\Lambda})\mathbb{E}\{\hat{\mathbf{u}}_{n-1}\} \tag{7.28}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{M-1})$ is the so called *spectral matrix* defined as diagonal matrix formed with the eigenvalues of $\mathbf{R}$ (any autocorrelation matrix can be factorized in this way). The (7.28) corresponds to a set of M disjoint first order finite difference equations, with time index n, of the type

$$\mathbb{E}\{\hat{u}_n(i)\} = (1-\mu\lambda_i)\mathbb{E}\{\hat{u}_{n-1}(i)\}, \quad n \geq 0, \quad i = 0, 1, \ldots, M-1. \tag{7.29}$$

Let $\hat{u}_{-1}(i)$ be the i.c. of the above iterative equation, the solution can be found by back substitution which is

$$\mathbb{E}\{\hat{u}_n(i)\} = (1-\mu\lambda_i)^n \hat{u}_{-1}(i) \tag{7.30}$$

which is stable if the following condition is satisfied

$$|1-\mu\lambda_i| < 1 \qquad \text{or} \qquad 0 < \mu < \frac{2}{\lambda_i}. \tag{7.31}$$

It follows that $\lim_{n\to\infty} \mathbb{E}\{\hat{u}_n(i)\} = 0$, $\forall i \in [0, M-1]$, we can then write

$$\lim_{n\to\infty} \mathbb{E}\{\hat{\mathbf{u}}_n\} = \mathbf{0}$$

or, equivalently

$$\lim_{n\to\infty} \mathbb{E}\{\mathbf{w}_n\} = \mathbf{w}_0.$$

It can be said then that, for $n \to \infty$, the vector $\mathbf{w}_n$ converges in average to the Wiener optimal solution. □

7.3.2.2 Convergence Speed and Condition Number of Hessian Matrix

From Eqn. (7.30), each coefficient $\hat{u}_n(i)$ decays to zero, i.e. $w_n(i) \to w_{\text{opt}}(i)$, with a speed that depends on μ and on the value of its relative eigenvalue λ_i. So if the eigenvalues are very different from each other, i.e. high *eigenvalue spread* or *eigenspread*, the convergence of the filter is conditioned by the smallest eigenvalue (i.e. the slowest mode). In this case it is said that *convergence is not uniform*.

For example, in Fig. 7.4 there is an example of convergence to the optimal values for a 2 coefficients filter ($M = 2$), considering two different step-size values. In right part of the figure is reported the trajectory of the weights superimposed on the contour plot of the quadratic performance surface $J(\mathbf{w})$. It is observed that the trajectories of the two cases are similar, but the convergence speed is different.

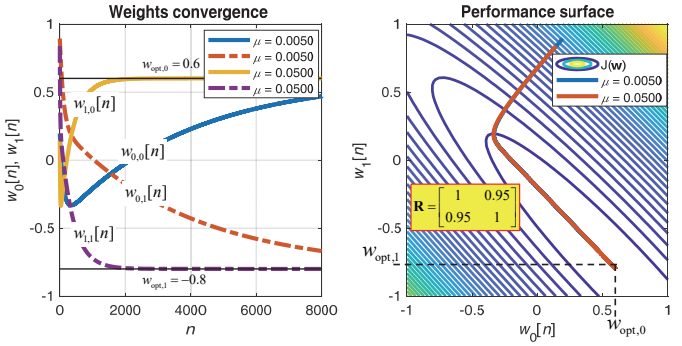


Fig. 7.4 Asymptotic convergence of weights $\mathbf{w}_n$ (for $M = 2$) towards the optimal value $\mathbf{w}_{\text{opt}} = [0.6 - 0.8]^T$. In right part of the figure is reported the trajectory of the weights superimposed on the contour plot of the quadratic function $J(\mathbf{w})$. Note that for high values of μ the convergence is faster.

Remark 7.6. Note that, the convergence speed depends on the shape of the performance surface (7.5), and that $\mathbf{R}$ (that also is the Hessian matrix of the surface $J(\mathbf{w})$), precisely describes the shape of the contour of $J(\mathbf{w})$.

Thus, the most influential effect for the convergence speed, is determined by the *condition number* of the covariance matrix defined as

$$\mathcal{X}(\mathbf{R}) \triangleq \lambda_{\max}/\lambda_{\min}.$$

Moreover, from basic geometry remember that the eigenvectors corresponding to the eigenvalues $\lambda_{\max}$ and $\lambda_{\min}$ are pointing, respectively, to the directions of maximum and minimum curvature of $J(\mathbf{w})$.

Remark 7.7. Observe that the convergence slows down if the surface is more eccentric (as that in Fig. 7.4), i.e. if the eigenspread is very high $\lambda_{\max}/\lambda_{\min} \gg 1$. For a circular contour of $J(\mathbf{w})$, is $\mathcal{X}(\mathbf{R}) = 1$ and the convergence to the optimum point can be obtained (theoretically) in one adaptation step.

Thus, from Eqn. (7.30) the LMS algorithm converges in probability on average to the Wiener optimal solution (i.e. $\Pr\{\lim_{n\to\infty} \mathbb{E}\{\mathbf{w}_n\} = \mathbf{w}_{\text{opt}}\} = 1$) when the learning rate

satisfy the stability condition (7.31). Thus, the fastest convergence of the dominant mode is obtained for $\mu = \frac{1}{\lambda_{\max}}$.

However, since for large M the estimate of $\lambda_{\max}$ is difficult and complex, the stability constraint is difficult to apply. Therefore to ensure convergence we need to consider a simpler and more practical criteria than that in Eqn. (7.31). So, considering the *trace* of the covariance matrix we have that $\text{tr}(\mathbf{R}) > \sum_{i=0}^{M-1} \lambda_i > \lambda_{\max}$ and, as $\text{tr}(\mathbf{R}) = M \cdot r[0]$ (where $r[0] = \mathbb{E}\{x^2[n]\}$), in practice a limit to the step-size that better guarantees convergence can be written as

$$0 < \mu < \frac{2}{M \cdot \mathbb{E}\{x^2[n]\}}$$

which assure the mean convergence of the weights. The convergence of weights variance $\lim_{n\to\infty} \mathbb{E}\{\mathbf{u}_n \mathbf{u}_n^T\} = 0$ (or small constant), requires more precautionary conditions that can be written as

$$0 < \mu < \frac{\mu_0}{\mu_1 + M \cdot \mathbb{E}\{x^2[n]\}} \tag{7.32}$$

where μ_1 avoids division by zero when the signal is zero, and μ_0 is determined experimentally (some authors, for Gaussian input suggest: $0.01 < \mu_0 < 0.1$).

A quantity often considered in the convergence analysis is the *excess of error*, defined as follows.

Definition 7.1. *Excess of steady-state error* - We can defined the *excess of error respect to steady-state MSE* (EMSE), denoted as J_{EMSE}, the quantity such that $J_{n\to\infty} = J_{\min} + J_{\text{EMSE}}$. So for (7.11) and (7.27) we can write $J_n = J_{\min} + \mathbb{E}\left\{\mathbf{u}_{n-1}^T \mathbf{R} \mathbf{u}_{n-1}\right\}$, i.e.

$$J_{\text{EMSE}} = \mathbb{E}\left\{\mathbf{u}_{n-1}^T \mathbf{R} \mathbf{u}_{n-1}\right\} = \text{tr}\left[\mathbf{R}\mathbf{K}_{n-1}\right] \tag{7.33}$$

where $\mathbf{K}_n \triangleq \mathbb{E}\{\mathbf{u}_n \mathbf{u}_n^T\}$ is *the covariance of the error vector.*

In other words, the excess MSE is the power of the additional error in the filter output due to the errors in the filter coefficients.

Definition 7.2. *Misadjustment* - An equivalent measure of the excess MSE in steady-state is the misadjustment, defined as

$$\mathcal{M} \triangleq \frac{J_{EMSE}}{J(\mathbf{w}_{\text{opt}})} = \frac{J_{EMSE}}{\sigma_v^2} \tag{7.34}$$

In addition, the following property is also valid.

Property 7.2. Indicating with $J_{\min}(\mathbf{w}) = \sigma_\eta^2$, the theoretical minimum level of the CF, considering the the excess of steady-state error and second-order statistical analysis, for brevity not reported (see for example [11]), it is possible to prove, that for a number of iterations that tends towards infinity (i.e. at steady-state), the following relation is valid

$$J_{n\to\infty}(\mathbf{w}_n) \approx \sigma_\eta^2 \left(1 + \frac{\mu}{2} \cdot M \cdot \mathbb{E}\{x^2[n]\}\right). \tag{7.35}$$

The previous relation is valid only for $\mu \ll 2/\lambda_{\max}$. Moreover, note that, when the input is $x[n] \sim \mathcal{N}(0,1)$ its correlation matrix is $\mathbf{R} \in \mathbb{R}^{M\times M} = \mathbf{I}$, i.e. $\lambda_{\max} = \lambda_{\min} = 1$ and thus, the Eqn. (7.29) convergence uniformly.

7.3.2.3 Experimental Performance Analysis and Convergence

To experimentally monitor the adaptation process, it is often useful to consider the plot of the *learning curve*, i.e. the MSE (or its log and smooth value), respect to the algorithm iterations $J(\mathbf{w}_n)$, for n=0,1, ... However, as the MSE is a statistical quantity, more properly, it should be plotted its ensemble average.

Practically, what we plot is a simple arithmetic average, sometime also smoothed by a zero-phase low-pass filter. The curve is evaluated on a certain number of different runs considering the performance analysis model in Fig. 7.3. The modeling experiment is repeated several time and averaged, starting from different weights i.c.

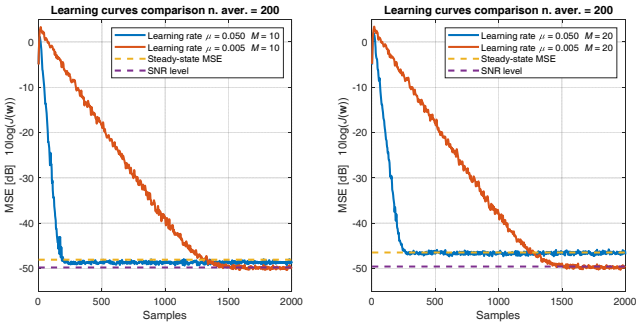


Fig. 7.5 Steady-state error evaluated with (7.35) for two different learning rate and filter length values reported in figure. For each experiment were considered the average of 200 runs with different i.c. (Modified from [11]).

For example, in Fig. 7.5 are reported the estimate learning curves averaged over 200 runs. In particular is plotted the MSE evaluated in decibel (i.e. $10 \cdot \log_{10} e^2[n]$) vs. the learning iterations n. Note that, for small learning rate $\mu = 0.005$, the curve reaches the lower limit due to the irreducible noise set at -50 dB (noise level in the figure). while for higher values $\mu = 0.05$, the steady-state error is approximately equal to the the lower limit value determined by the theoretical analysis in Eqn. (7.35) (also reported in the figure).

As a further experiments, in, in Fig. 7.6 are reported the comparison of LMS learning curve averaged over 200 runs for $M = 8$, when the input is a simple unit variance Gaussian white noise (WGN), and for unit-variance narrow-band colored process generated by a simple 1-st order moving average Markov process defined as

$$x[n] = bx[n-1] + \sqrt{1-b^2}\eta[n], \qquad \eta[n] \sim \mathcal{N}(0,1). \tag{7.36}$$

It should be noted that in the case of input colored noise, the covariance matrix of the input signal is no longer the unitary matrix but takes the form

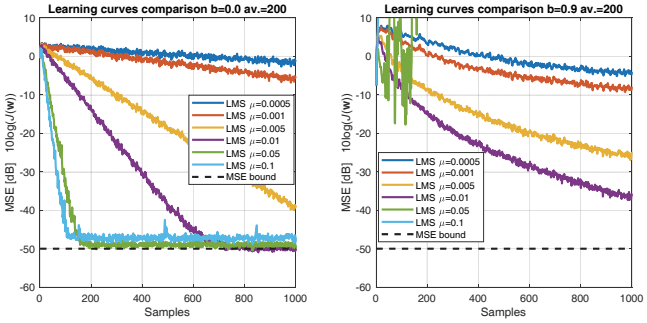


Fig. 7.6 Comparison of LMS learning curve, for different step-size μ averaged over 200 runs. left) white noise ($b = 0.0$) input; right) narrowband MA colored process ($b = 0.9$). (Modified from [11]).

$$\mathbf{R}_{xx} = \sigma_\eta^2 \begin{bmatrix} 1 & a & a^2 & \cdots & a^{M-1} \\ a & 1 & a & \cdots & a^{M-2} \\ a^2 & a & 1 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & a \\ a^{M-1} & a^{M-2} & \cdots & a & 1 \end{bmatrix}$$

for which the convergence conditions expressed by the (7.31) are more difficultly verified.

7.3.3 LMS Algorithm's Variants

In the literature there are many LMS algorithm variations. Generally, these variants are made to have a better convergence speed, or better steady-state performance, or others, tend to stabilize the weight trajectories, ect.

7.3.3.1 Normalized LMS Algorithm

The *normalized LMS* (NLMS) algorithm is a direct consequence of the expression (7.32), and represents a very used variant to accelerate the convergence speed at the expense of a modest increase in the computational cost. The NLMS is characterized by a variable learning rate according to the following law

$$\mu_n = \frac{\mu}{\delta + \|\mathbf{x}_n\|_2^2} \tag{7.37}$$

so, the update formula is

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu \frac{e[n]\mathbf{x}_n}{\delta + \mathbf{x}_n^T \mathbf{x}_n}. \tag{7.38}$$

With $\mu \in (0, 2]$ and $\delta > 0$. Note that δ is the *regularization parameter* which also ensures the computability of (7.38) in the case of zero input.

The Eqn.s (7.37)-(7.38) indicate that the step size is inversely proportional to the energy of the input signal. Given the implementative simplicity, the NLMS is one

of the most used algorithms in the audio equalization applications, echo cancelation, active noise control etc.

Remark 7.8. Observe that a simple way to reduce the number of multiplications for the $\|\mathbf{x}_n\|_2^2$ calculation, is obtained by observing that the vector $\mathbf{x}_n$ contains $M-1$ common values with the $\mathbf{x}_{n-1}$ vector. For this reason, the following relationship holds

$$\|\mathbf{x}_n\|_2^2 = \|\mathbf{x}_{n-1}\|_2^2 - |x[n-M]|^2 + |x[n]|^2 \tag{7.39}$$

7.3.3.2 Proportionate LMS Algorithms

The *proportionate NLMS* (PNLMS) algorithm, proposed in [43], is characterized by an adaptation rule of the type

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu \frac{\mathbf{g}_{n-1}\mathbf{x}_n e[n]}{\delta_p + \mathbf{x}_n^T \mathbf{G}_{n-1}\mathbf{x}_n} \tag{7.40}$$

where $0 < \mu < 1$ and $\mathbf{g}_n \in \mathbb{R}^{M\times M} = \text{diag}[\, g_n(0)\ g_n(1)\ \cdots\ g_n(M-1)\,]$ is a diagonal matrix identified in order to adjust the steps size of the filter weights in an individual mode. The $\mathbf{G}_n$ matrix is determined so as to have the step size proportional to the amplitude of the considered filter coefficient. In other words, the larger coefficients have a greater increase. Following this philosophy, a possible $\mathbf{G}_n$ matrix choice, is the following

$$\gamma_n[m] = \max\{\rho \cdot (\max[\delta_p,\ \ |w_n[0]|,\ \ ...,\ \ |w_n[M-1]|]),\ \ |w_n[m]|\} \tag{7.41}$$

$$g_n(m) = \frac{\gamma_n[m]}{\|\gamma_n\|_1}, \qquad m = 0,\ \ 1,\ \ ...,\ \ M-1 \tag{7.42}$$

where $\gamma_n \in \mathbb{R}^{M\times 1} = [\,\gamma_n[0]\ \cdots\ \gamma_n[M-1]\,]^T$ and $\delta_p, \rho \in \mathbb{R}^+$, called *precautionary constants*, have typical values $\rho = 0.01$ and $\delta_p = 0.01$. In practice δ_p is a regularization parameter that ensures the consistency of the (2.106), also for null taps, ρ serves to prevent stalling of the m-th coefficient $w_n[m]$ when its amplitude is lower than the amplitude of the maximum coefficient.

In the algorithm called *improved PNLMS* (IPNLMS) [44], a more elegant $\mathbf{G}_n$ matrix choice, is proposed

$$\gamma_n[m] = (1-\beta)\frac{\|\mathbf{w}_n\|_1}{M} + (1+\beta)\,|w_n[m]| \tag{7.43}$$

$$\begin{aligned} g_n(m) &= \frac{\gamma_n[m]}{\|\gamma_n\|_1} \\ &= \frac{(1-\beta)}{2M} + (1+\beta)\frac{|w_n[m]|}{2\,\|\mathbf{w}_n\|_1}, \quad m = 0,\ \ 1,\ \ ...,\ \ M-1 \end{aligned} \tag{7.44}$$

where ($-1 < \beta < 1$) represents the proportionality control parameter. Note that for $\beta = -1$, the IPNLMS coincides with the NLMS. As reported in [44], [45], a good choice of the parameter of proportionality β is –0.5 or 0. Furthermore, in the IPNMLS is usual, to choose the regularization parameter with in the form

$$\delta_p = \frac{(1-\beta)}{2M}\delta \tag{7.45}$$

where δ is the NLMS regularization parameter.

Remark 7.9. The proportional algorithms, are suitable in the case of systems identification with *sparse* impulse response. A simple definition of *sparsity* is the following: *an impulse response is called sparse if a large fraction of its energy is concentrated in a small fraction of its duration*. In more formal terms, a simple measure of an impulse response $\mathbf{w}$ sparseness is the following [45]

$$\xi(\mathbf{w}) \triangleq \frac{M}{M-\sqrt{M}}\left(1-\frac{\|\mathbf{w}\|_1}{\sqrt{M}\,\|\mathbf{w}\|_2}\right) \tag{7.46}$$

where, we remind the reader that the L_1 and L_2 norms are defined, respectively, as

$$\|\mathbf{w}\|_1 = \sum_{m=0}^{M-1}|w[m]|, \qquad \|\mathbf{w}\|_2 = \sqrt{\sum_{m=0}^{M-1}|w[m]|^2} \tag{7.47}$$

for which $0 \leq \xi(\mathbf{w}) < 1$ and for sparse $\mathbf{w}$, we have that $\xi(\mathbf{w}) \to 1$.

Remark 7.10. Note that, a typical example of sparse impulse response, is the one that refers to TF of an acoustic path in a reverberating environment as, for example, the impulse response shown in Fig. 3.7 (see §3.2.4.1).

7.3.4 Mini Batch or Block Adaptive Filter

In the block or mini batch algorithms class, represented schematically in Fig. 7.7, the input signal is stored in a L-length buffer (*block length*), to allow the output and weights update to be periodically calculated, with a period equal to L.

Said k the block index and $\mathbf{w}_k \in \mathbb{R}^{M\times 1}$ the filter weights vector, the parameter update is characterized by a relation of the type

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \frac{1}{L}\Delta\mathbf{w}_k \tag{7.48}$$

in which $\Delta\mathbf{w}_k$, defined as a *block update parameter*, is given by the sum of the instantaneous variations $\Delta\mathbf{w}_i$, i.e.

$$\Delta\mathbf{w}_k = \sum_{i=0}^{L-1}\Delta\mathbf{w}_i. \tag{7.49}$$

With this definition, said i the time index inside the block, the input sequence time index n is defined as

$$n = kL+i, \qquad \begin{matrix} i = 0,\ 1,\ \ldots,\ L-1 \\ k = 1,\ 2,\ \ldots\ . \end{matrix} \tag{7.50}$$

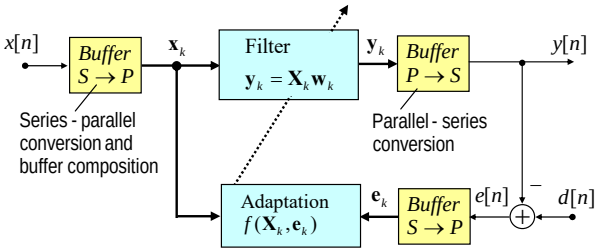


Fig. 7.7 General scheme of a *block adaptive filter*.

The term $\Delta\mathbf{w}_i$ is linked to the instantaneous estimate of the CF gradient ∇J_i and its calculation is performed for every block index, while keeping fixed filter coefficients.

The input signal, as in the LS methodology, is stored in a matrix $\mathbf{X}_k$, indicated as *block data matrix* or simply *block data*, such that the k-th block output can be calculated as the convolution sum expressed in terms of the following matrix-vector product

$$\mathbf{y}_k = \mathbf{X}_k \mathbf{w}_k. \tag{7.51}$$

Thus, the k-th block data matrix $\mathbf{X}_k \in \mathbb{R}^{L\times M}$ can defined, by row (or column composition) of the input signal, as

$$\mathbf{X}_k = \left[\, \mathbf{x}_{kL}\ \mathbf{x}_{kL-1}\ \cdots\ \mathbf{x}_{kL-L+1} \,\right]^T \tag{7.52}$$

where the signal vectors $\mathbf{x}_{kL} = [x[kL],\ x[kL-1]...,\ x[kL-M+1]]$. In addition, note that the matrix $\mathbf{X}_k$ contains the input signal samples arranged in columns/rows shifted of one sample. For example, in the case of $L = 4$ and $M = 3$ for the k-th index the (7.51) can be expressed as:

$$k \quad \rightarrow \begin{bmatrix} y[4k] \\ y[4k-1] \\ y[4k-2] \\ y[4k-3] \end{bmatrix} = \begin{bmatrix} x[4k] & x[4k-1] & x[4k-2] \\ x[4k-1] & x[4k-2] & x[4k-3] \\ x[4k-2] & x[4k-3] & x[4k-4] \\ x[4k-3] & x[4k-4] & x[4k-5] \end{bmatrix} \begin{bmatrix} w_k[0] \\ w_k[1] \\ w_k[2] \end{bmatrix}$$

while for the $k-1$-th index we have that

$$k-1 \rightarrow \begin{bmatrix} y[4k-4] \\ y[4k-5] \\ y[4k-6] \\ y[4k-7] \end{bmatrix} = \begin{bmatrix} x[4k-4] & x[4k-5] & x[4k-6] \\ x[4k-5] & x[4k-6] & x[4k-7] \\ x[4k-6] & x[4k-7] & x[4k-8] \\ x[4k-7] & x[4k-8] & x[4k-9] \end{bmatrix} \begin{bmatrix} w_{k-1}[0] \\ w_{k-1}[1] \\ w_{k-1}[2] \end{bmatrix}$$

note that for $L = M$, the matrix $\mathbf{X}_k$ is Toeplitz.

For other vectors, similar to LS algorithm in §7.2.2, we have the following definitions

$$\begin{aligned} \mathbf{d}_k \in \mathbb{R}^{L\times 1} &\triangleq \left[\, d[kL]\ d[kL-1]\ \cdots\ d[kL-L+1] \,\right]^T \\ \mathbf{y}_k \in \mathbb{R}^{L\times 1} &\triangleq \left[\, y[kL]\ y[kL-1]\ \cdots\ y[kL-L+1] \,\right]^T \\ \mathbf{e}_k \in \mathbb{R}^{L\times 1} &\triangleq \left[\, e[kL]\ e[kL-1]\ \cdots\ e[kL-L+1] \,\right]^T \end{aligned} \tag{7.53}$$

for which, for the error vector can be defined as

$$\mathbf{e}_k = \mathbf{d}_k - \mathbf{y}_k. \tag{7.54}$$

From (7.51), the filter coefficients $\mathbf{w}_k$ remain constant for all L output samples $\mathbf{y}_k$ and the convolution can be performed with a block algorithm.

As regards the block length, we can identify three distinct situations: $L = M$, $L < M$ and $L > M$. The most common choice is that in which the block length is equal to (or less) the filter length and, in this case, the possibility to compute the convolution in the frequency domain suggests filter lengths equal to powers of two.

7.3.4.1 Adaptive Filtering by Block LMS

The block (full batch or mini batch) algorithms, can be extended to adaptive filtering process if the data matrix is filled by the input sequence with a *sliding window* mechanism. For example, considering a N-samples sequence, the data matrix can be defined defining M-length window (for $M \leq N$) , which flows on the signal and fills the row of matrix $\mathbf{X}$ as shown in Fig. 7.8.

To derive the structure of the iterative LS algorithm, we can generalize in vector form the LMS algorithm. In fact, as the output vector can be expressed as $\mathbf{y} = \mathbf{X}\mathbf{w}_{n-1}$, the gradient of the quadratic cost function in Eqn. (7.17) can be written as

$$\nabla J(\mathbf{w}) \triangleq \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{X}^T(\mathbf{d} + \mathbf{y}) = -2\mathbf{X}^T\mathbf{e}. \tag{7.55}$$

So, incorporating all the scalar parameter in the learning rate μ_n, we can write

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu_n\left[-\nabla J(\mathbf{w})\right] \tag{7.56}$$

that is equivalent the following forms denoted as *block LMS algorithm* (BLMS)

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu_n \mathbf{X}_n^T \mathbf{e}_n \tag{7.57}$$

where note that the data-matrix $\mathbf{X}_n$ is a *Toeplitz matrix*, that is defined as a matrix where each descending diagonal from right to left (or left to right) is constant[3].

In addition, to ensure the algorithm stability, the parameter μ_n should be upper-bounded. In fact, note that the algorithm coincides with that of Landweber [12], which converges to the LS solution, when the learning rates, are such that $0 < (\mathbf{I} - \mu_n \mathbf{X}^T\mathbf{X} < 1$. In other words, similar to the condition (7.31), the learning rates are such that $0 < \mu_n < \frac{1}{\lambda_{\max}}$ (where $\lambda_{\max}$ is the maximum eigenvalue of $\mathbf{X}^T\mathbf{X}$). The algorithm converges quickly in case that μ_n is close to its upper limit.

Remark 7.11. Note that the expression (7.57) in the adaptive filtering is sometime denoted as the empirical *steepest descent algorithm* (SDA).

[3] Toeplitz symmetry allows the development of many robust inversion algorithms with a reduced computational complexity, as the Levinson algorithm (that is $O(M^2)$). A Toeplitz matrix can also be decomposed (i.e. factored) in $O(M^2)$ time for example by the LU decomposition that gives a quick and robust method for solving a LS system, and also for computing the determinant [11].

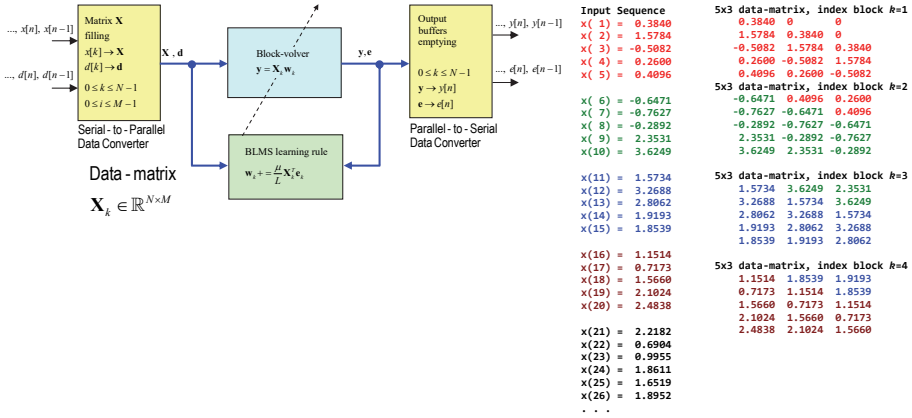


Fig. 7.8 Mini-batch LS adaptive filtering scheme and data matrix $\mathbf{X}$ filling mechanism

7.3.4.2 Summary of BLMS Algorithm

The BLMS algorithm is then defined by the following iterative procedure

$$\begin{aligned} \mathbf{y}_k &= \mathbf{X}_k \mathbf{w}_k \\ \mathbf{e}_k &= \mathbf{d}_k - \mathbf{y}_k \\ \mathbf{w}_{k+1} &= \mathbf{w}_k + \frac{\mu_B}{L} \mathbf{X}_k^T \mathbf{e}_k \end{aligned} \tag{7.58}$$

Remark 7.12. The first of (7.58) represents a convolution, while the third a cross-correlation. In order to obtain greater computational efficiency and, moreover, better convergence characteristics, as we shall see in the following, both expressions can be implemented in the frequency domain.

7.4 Second-Order Adaptive Filtering

This Section introduces the second-order algorithms for the solution of the Yule-Walker normal equations. Besides the gradient, in the second order algorithms is used the information of the second derivative, referred to as the Hessian matrix (i.e. the covariance), that takes into account the curvature of the performance surface.

In presence of colored signal the covariance presents a high eigenspread whereby, for quadratic cost functions, second-order algorithms converge in very few iterations, at the limit only one [6]-[11]. However, in the audio sector, in the presence of long impulsive responses, second-order algorithms must be used with great caution and low-cost variants are used.

7.4.1 Sequential Regression Algorithms

Let us consider the second order derivative $\nabla^2 J(\mathbf{w})$ that is a matrix, denoted as *Hessian matrix*, defined as:

$$\begin{aligned}\nabla^2 J(\mathbf{w}) &\triangleq \frac{\partial J(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T} = \frac{\partial}{\partial \mathbf{w}} \left[\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right]^T \\ &= \begin{bmatrix} \frac{\partial}{\partial w_1}\left[\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right]^T \\ \frac{\partial}{\partial w_2}\left[\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right]^T \\ \vdots \\ \frac{\partial J}{\partial w_M}\left[\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right]^T \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 J(\mathbf{w})}{\partial w_1^2} & \frac{\partial^2 J(\mathbf{w})}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 J(\mathbf{w})}{\partial w_1 \partial w_M} \\ \frac{\partial^2 J(\mathbf{w})}{\partial w_2 \partial w_1} & \frac{\partial^2 J(\mathbf{w})}{\partial w_2^2} & \cdots & \frac{\partial^2 J(\mathbf{w})}{\partial w_2 \partial w_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J(\mathbf{w})}{\partial w_M \partial w_1} & \frac{\partial^2 J(\mathbf{w})}{\partial w_M \partial w_2} & \cdots & \frac{\partial^2 J(\mathbf{w})}{\partial w_M^2} \end{bmatrix} \end{aligned} \tag{7.59}$$

If $\nabla^2 J(\mathbf{w})$ is semi-defined positive, that is $\mathbf{w}^T \cdot \nabla^2 J(\mathbf{w}) \cdot \mathbf{w} \geqslant 0$, $\forall \mathbf{w}$, the adaptation performance can be improved by using a second order update formula defined as

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \mu_k \left[\nabla^2 J(\mathbf{w})\right]^{-1} \{-\nabla J(\mathbf{w})\}. \tag{7.60}$$

Note that, in LMS the gradient determines the direction of the descent algorithm at point $\mathbf{w}_k$; while considering the Hessian matrix $\nabla^2 J(\mathbf{w})$, (that represents the information on the CF curvature), we can determine: 1) the adaptation step length; and 2) the optimal direction towards the CF minimum.

In other terms, let $\mathbf{v}_k = -\nabla J(\mathbf{w})$ be the vector of negative gradient direction, the expression (7.61) can be considered a special case of a more general formulation written as $\mathbf{w}_k = \mathbf{w}_{k-1} + \mu_k \mathbf{H}_k \mathbf{v}_k$, where $\mathbf{H}_k$ is a *weighing matrix* determinable as the inverse of the Hessian matrix $\mathbf{H}_k = [\nabla^2 J(\mathbf{w})]^{-1}$, or in various other ways. Thus, the matrix $\mathbf{H}_k$ can be interpreted as a "suitable" linear transformation to determine a optimum adaptation step (direction and length), such that the descent along the CF, can be performed in very few steps.

In the literature, there are numerous variations and specializations of the method (7.61). Some of the most common are below indicated.

7.4.1.1 The Levenberg-Marquardt Variants

In the Levenberg-Marquardt variant [12], [13], the Eqn. (7.61) is rewritten as

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \mu_k \left[\nabla^2 J(\mathbf{w}) + \delta \mathbf{I}\right]^{-1} \{-\nabla J(\mathbf{w})\} \tag{7.61}$$

in which the constant $\delta > 0$, denoted as *regularization term*, should be chosen considering two opposing requirements: possibly small to avoid biased solution, and sufficiently large such that the Hessian is always a positive definite matrix and to allow more "regular" and robust solutions (but affected by some bias).

7.4.1.2 Second Order Online LS Algorithm

In LS cost function the gradient is defined as $\nabla J(\mathbf{w}) \in \mathbb{R}^{M\times 1} = -\mathbf{X}\mathbf{e}$, while the Hessian can be approximated by th empirical covariance matrix is $\nabla^2 J(\mathbf{w}) \in \mathbb{R}^{M\times M} = \mathbf{X}^T\mathbf{X} = \mathbf{R}_{xx}$. So, considering also the Levenberg-Marquardt variant, for $\delta > 0$, the Eqn. (7.61) can be rewritten in the following equivalent forms of finite-difference equations (FDE) as

$$\begin{aligned}\mathbf{w}_n &= \mathbf{w}_{n-1} + \mu_n(\mathbf{X}^T\mathbf{X} + \delta\mathbf{I})^{-1}\mathbf{X}^T\mathbf{e} \\ &= \mathbf{w}_{n-1} + \mu_n(\mathbf{R}_{xx} + \delta\mathbf{I})^{-1}\mathbf{X}^T\mathbf{e} = \mathbf{w}_{n-1} + \mu_n\mathbf{X}^{\#}\mathbf{e}.\end{aligned} \tag{7.62}$$

In other words, the expression (7.62) take into account not only of the gradient but also of curvature of the performance surface carried out by the term $\left[\nabla^2 J(\mathbf{w})\right]^{-1}$. This formulation is the basic form of second-order adaptive algorithm class.

Remark 7.13. Observe that for $N = 1$, as $\mathbf{x}_n \in \mathbb{R}^{M\times 1}$, the expression (7.62) has the form

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu_n\mathbf{R}_{xx}^{-1}\mathbf{x}_n e[n] \tag{7.63}$$

denoted as *quasi-Newton LMS algorithm.*

Remark 7.14. Observe that the formulations (7.62) and (7.63), have only a theoretical meaning because in the algorithms used in real applications we tend to avoid the use of large matrices, let alone their inversion that usually is performed by appropriate recursive matrix factoring. Moreover, the recursive solution of the Yule-Walker equation is usually denoted as *Sequential Regression Algorithm* (SRA) that is similar but does not coincide with the so called *Recursive Least Square* (RLS), algorithm, and that can be considered as a particular case of the Kalman filter, which instead provides for a proper weighing of the normal equations by a parameter indicated as *forgetting factor* and a fast recursive formulation of the inverse covariance matrix estimation [6]-[11].

7.4.2 Affine Projection Algorithm

The formulation in Eqn. (7.62) is defined only for overdetermined linear systems ($N > M$). In the case of underdetermined system, very common in audio applications when impulse responses are very long, it is possible to use the following matrix identity

$$(\mathbf{X}_n^T\mathbf{X}_n + \delta\mathbf{I})^{-1}\mathbf{X}_n^T = \mathbf{X}_n^T(\mathbf{X}_n\mathbf{X}_n^T + \delta\mathbf{I})^{-1}. \tag{7.64}$$

Moreover, when the number of free parameter is very high, the data-matrix can be defined over a pre-specified time-depth (or by appropriately selecting a reduced number of non consecutive equations), as $\mathbf{X}_n^{K\times M}$, with $K \ll M$. Thus, from the above identity, the recurvise formulation (7.62) can be rewritten as

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu_n\mathbf{X}_n^T(\mathbf{X}_n\mathbf{X}^T + \delta\mathbf{I})^{-1}\mathbf{e}. \tag{7.65}$$

Table 7.2 Affine projection algorithm (APA) for adaptive filtering

Given input and desired sequences $[x[n]\ d[n]]_{n=1}^{N}$
Initialization $\mathbf{w},\ \mu,\ K,\ \delta$
Delay-lines and Data-matrix def. $\mathbf{x}_n \in \mathbb{R}^{M\times 1} \leftarrow \mathbf{0},\ \mathbf{d}_n \in \mathbb{R}^{K\times 1} \leftarrow \mathbf{0},\ \mathbf{X} \in R^{M\times K} \leftarrow \mathbf{0}$
for $n = 0, 1, \ldots$ { *// for each input samples' pair*
$\mathbf{x}_n = [x[n]; \mathbf{x}_n(1:M-1)]$; *// delay-line update: shift and load a new input sample*
$\mathbf{X} = [\mathbf{X}(2:K,:); \mathbf{x}_n^T]$; *// shift-up the data matrix and load new row vector*
$\mathbf{d}_n = [\mathbf{d}_n(2:K); d[n]]$; *// delay-line update: shift and load a new target sample*
$\mathbf{y} = \mathbf{X}\mathbf{w}$; *// compute the filter output*
$\mathbf{e} = \mathbf{d}_n - \mathbf{y}$; *// compute the a priori error vector*
$\mathbf{w} = \mathbf{w} + \mu\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \delta\mathbf{I})^{-1}\mathbf{e}$; *// APA adaptation rule*
}

In this class algorithms, denoted as *affine projection algorithms* (APAs), the matrix to be inverted has a size of $K \times K$ where the value of K can be chosen as a compromise between the desired performances and the computational resource available.

In Table 7.4.2 is reported the schematic APA algorithm for the adaptive filtering.

Remark 7.15. This class of APA algorithms, is widely used in adaptive audio applications since: 1) it has performances very close to the second order algorithms, with a computational overhead that can be chosen by the user; 2) has excellent performance in the presence of non-Gaussian input such as the speech signal.

7.4.3 Recursive Least Squares Algorithm

Known in the literature as *Recursive Least Squares* (RLS), the algorithm is different from the ordinary LS sequential regression form above described, as the inverse of correlation matrix is estimated recursively, assuming a certain *forgetting factor*, and avoid any matrix inversion.

The CF $\hat{J}_n(\mathbf{w})$ for this class of algorithms, has an expression of the type

$$J_n(\mathbf{w}) = \sum_{i=0}^{n} \lambda^{n-i} |e[i]|^2 \tag{7.66}$$

where the constant $0 \ll \lambda \leq 1$, defined as *forgetting factor* that takes into account the *algorithm memory*. In other words, the CF depends on instantaneous current error $\lambda^0 e[n]$ and to the past step errors: $\lambda e[n-1]$, $\lambda^2 e[n-2]$, $\lambda^3 e[n-3]$, ...; considering a weighting function defined by the forgetting factor. Note that for $\lambda = 1$ the influence of the past errors has same weight of the instantaneous error, in such case the algorithm is called *growing memory*.

For a complete discussion of the RLS, refer to specialized literature (e.g. [6]-[11]). Below is a brief summary of the standard RLS.

7.4.3.1 Updating with *a priori* Error

Consider the Newton LMS algorithm (see Eqn. (7.63))

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{P}_n \mathbf{x}_n e[n] \tag{7.67}$$

where $\mathbf{P}_n$ is an estimate of the inverse of the Hessian matrix. Defining the *Kalman gain vector* as $\mathbf{k}_n = \mathbf{P}_n \mathbf{x}_n$, the updating formula (7.67) is rewritten as

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{k}_n e[n]. \tag{7.68}$$

Table 7.3 Summary of the conventional RLS algorithm.

Given input and desired sequences $[x[n]\ d[n]]_{n=1}^{N}$

Initialization $\mathbf{w}_{-1} = \mathbf{0},\ \mathbf{x}_{-1} = \mathbf{0},\ \mathbf{P}_{-1} = \delta^{-1}\mathbf{I},\ \delta,\ \lambda$

for $n = 0, 1, \ldots$ { *// for each input samples' pair*

$\mathbf{x}_n = [x[n]; \mathbf{x}_n(1 : M-1)]$; *// shift and load a new input sample*

$y[n] = \mathbf{w}_n^H \mathbf{x}_n$; *// compute the filter output*

$e[n] = d[n] - y[n]$; *// compute the a priori error*

$\mathbf{k}_n = \dfrac{\lambda^{-1}\mathbf{P}_{n-1}\mathbf{x}_n}{1 + \lambda^{-1}\mathbf{x}_n^H \mathbf{P}_{n-1}\mathbf{x}_n}$; *// Kalman gain vector*

$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{k}_n e^*[n]$; *// weights update*

$\mathbf{P}_n = \lambda^{-1}(\mathbf{P}_{n-1} - \mathbf{k}_n \mathbf{x}_n^H \mathbf{P}_{n-1})$; *// Riccati equation*

}

In RLS the matrix $\mathbf{P}_n$ is estimated on-line without explicit inversion with the *matrix inversion lemma* [4] (MIL) as

$$\mathbf{k}_n = \frac{\lambda^{-1}\mathbf{P}_{n-1}\mathbf{x}_n}{1 + \lambda^{-1}\mathbf{x}_n^H \mathbf{P}_{n-1}\mathbf{x}_n}$$
$$\mathbf{P}_n = \lambda^{-1}(\mathbf{P}_{n-1} - \mathbf{k}_n \mathbf{x}_n^H \mathbf{P}_{n-1}).$$

A summary of the conventional RLS is reported in Table 7.3.

For more details see [9]-[11].

[4] A variant of the matrix inversion lemma (or Sherman-Morrison-Woodbury formula) widely used in adaptive filtering proposed by Kailath [3], is defined as: $(\mathbf{A} + \mathbf{x}\mathbf{x}^H)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{x}\mathbf{x}^H\mathbf{A}^{-1}}{1+\mathbf{x}^H\mathbf{A}^{-1}\mathbf{x}}$.

7.5 Frequency Domain Adaptive Filtering

The subject of *frequency domain adaptive filtering* (FDAF) is a very broad topic, which presents many variations and specializations, evidenced by the numerous contributions, including recent ones, in the scientific literature (see for example [15]-[23]).

Closely related to the fast frequency domain filtering introduced in §4.6, these algorithms have a high usability in audio applications in which the filter length is very high and is also required high computational efficiency.

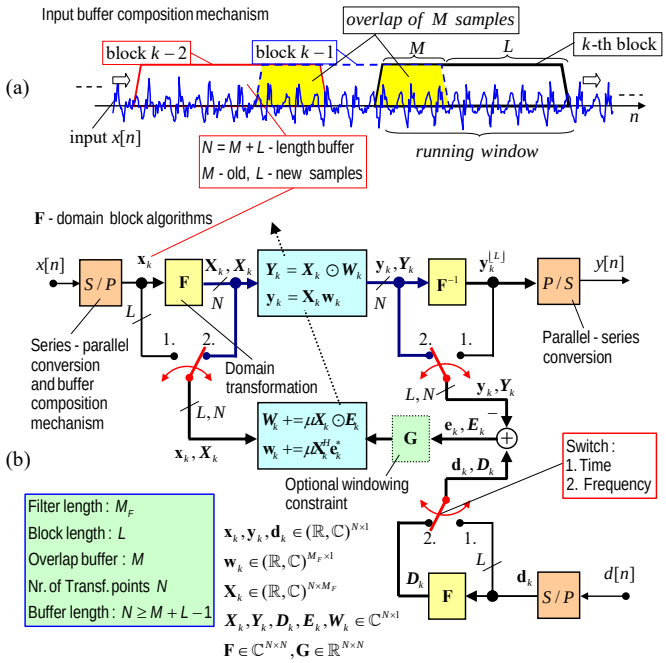


Fig. 7.9 Block adaptive filtering algorithms. a) Input signal buffer composition mechanism of overlap-save (OLS) convolution (see §4.6). b) General framework for block algorithms in time and transformed domain by the operator $\mathbf{F}$. For $\mathbf{F} = \mathbf{I}$, the algorithm is entirely in the time domain and the switches 1. or 2. position, is indifferent. For $\mathbf{F} \neq \mathbf{I}$, the weights adaptation can be done in the time-domain (switch position 1.) or in the transformed domain (switch position 2.). (Modified from [11]).

In particular, in this section are presented some known algorithms such as FDAF, which has a recursive formulation similar to BLMS. Also known in the literature as *fast LMS* (FLMS), were presented for the first time by Ferrara in [17] and, independently, by Clark, Mitra and Parker in [18].

A general representation framework for block algorithms described in this chapter, is reported in Fig. 7.9 that shows a possible mechanism example, for buffer composition.

- $\mathbf{F} \in \mathbb{C}^{N \times N}$ - Linear transform operator (e.g. discrete-Fourier transform (DFT) matrix) such that $\mathbf{F} \cdot \mathbf{F}^H = \mathbf{I}$;
- $\mathbf{G}$ - Symbol that represents a *windowing constraint* of the output signal, error or weights;

- $\mathbf{x}_k$, $\mathbf{y}_k, \mathbf{w}_k \in \mathbb{R}^{L\times 1}$ - Time-domain vectors sequence, respectively, of the input, output blocks and filter weights;
- $\mathbf{X}_k \in \mathbb{R}^{N\times M}$ - Time-domain input data matrix;
- $\boldsymbol{X}_k$, $\boldsymbol{Y}_k$, $\boldsymbol{W}_k \in \mathbb{C}^{N\times 1}$, ... - Frequency domain quantity: input, output, filter weights vectors etc.

Again with reference to Fig. 7.9, the output and error signal windowing constraint $\mathbf{G}$, and that of the weights (the latter not shown in the figure) is necessary for the proper implementation of the inverse FFT. Note, also, the presence of the switches with position 1. and 2. This presence indicates that the adaptive filtering algorithm can be implemented in mixed mode: the output calculation in the transformed domain and weights update in time domain. For $\mathbf{F} = \mathbf{I}$, the algorithm operates entirely in the time domain and, as the reader can observe from the figure, in this case the switches positions are indifferent.

7.5.1 Introduction to FDAF

The FDAF has a recursive formulation similar to block LMS in §7.3.4. In the time domain BLMS learning rule is $\nabla J_k = \mathbf{X}_k^T \mathbf{e}_k$, the gradient estimate is determined by the cross-correlation between the data vector $\mathbf{x}_k$ and the error $e[k]$. Thus the weight update equation can be formulated as $\mathbf{w}_{k+1} = \mathbf{w}_k + \frac{\mu_B}{L}\mathbf{X}_k^T \mathbf{e}_k$. Transforming this rule in the frequency domain, as suggested in [23], and using a compact and general notation [11], we obtain

$$\boldsymbol{W}_{k+1} = \boldsymbol{W}_k + \mathbf{G}\left[\mu_k \odot \boldsymbol{X}_k^H \odot \boldsymbol{E}_k\right] \tag{7.69}$$

where $[\cdot]^H$ is the Hermitian operator, the symbol $\odot$ represents the pointwise product (i.e. the time-domain matrix-vector product $\mathbf{X}_k^T \mathbf{e}_k$ that implements the cross-correlation is transformed by a more simple pointwise vector product $\boldsymbol{X}_k^H \odot \boldsymbol{E}_k$ in the frequency domain), the vector $\mu_k = [\mu_k(0), \mu_k(1), ..., \mu_k(N-1)]$, contains the learning rates or step size that can be of different value for each frequency bin. The matrix $\mathbf{G}$ represents the windowing or gradient constraint, necessary to impose the linearity of the correlation in the gradient calculation that can be interpreted as a particular signal pre-windowing in the time domain. The matrix $\mathbf{G}$ is inserted in learning rule only in order to generalize the FDAF formalism.

In the class of the FDAF algorithms the error calculation can be performed directly in the time or frequency domain. In the case where the error is calculated in frequency domain, the gradient constraint can be chosen unitary $\mathbf{G} = \mathbf{I}$ and the FDAF is said unconstrained frequency domain adaptive filter (UFDAF).

7.5.2 Frequency-bin step size normalization

One of the main advantages of the frequency approach is that the adaptation equations (7.69) are decoupled, i.e. in the frequency domain, the convergence of each filter coefficient is not dependent from the others. It follows that to speed-up the convergence rate such that we can obtain a uniform convergence, it is possible to define a

simple *power normalization rule*. Indicating with $P_k(m)$ the estimation power of the m-th frequency bin and, let μ a suitable predetermined scalar parameter, the step-size can be chosen independently for each frequency bin m, proportional to the inverse of its power, i.e.

$$\mu_k(m) = \mu/[\alpha + P_k(m)], \qquad m = 0, \ \ldots, \ N-1 \tag{7.70}$$

where the parameter $0 < \alpha \ll 1$, avoid the zero-division. So, the power normalization rule allows to accelerate the slower convergence modes. Obviously, in the case of white and stationary input processes, the powers are identical for all frequencies bin and we have $\boldsymbol{\mu}_k = \mu \mathbf{I}$. Moreover, to avoid significant step-size discontinuity that could destabilize the adaptation, as suggested by some authors (see, for example, [20]), it is appropriate to estimate m-th power frequency bin with a one-pole lowpass smoothing filter usually implemented by the following finite difference equation

$$P_k(m) = \lambda P_{k-1}(m) + (1-\lambda)|X_k(m)|^2 \tag{7.71}$$

where λ represents a forgetting parameter and $|X_k(m)|^2$ the m-th measured energy bin.

7.5.3 Overlap save FDAF

The overlap-save FDAF (OS-FDAF) algorithm, is the frequency domain equivalent of the BLMS, it has the same convergence characteristics in terms of speed, stability, misalignment, and the algorithm converges, in average, to the optimum Wiener solution (see, for example, [23]). The possibility of choosing learning rates different for each frequency bin, as with the power normalization in Eqn. (7.71), allows a convergence speed improvement without, however, improving the reachable minimum MSE. Compared to BLMS, the OS-FDAF allows the dual advantage of having reduced complexity and, exploiting the step-size normalization an higher convergence speed.

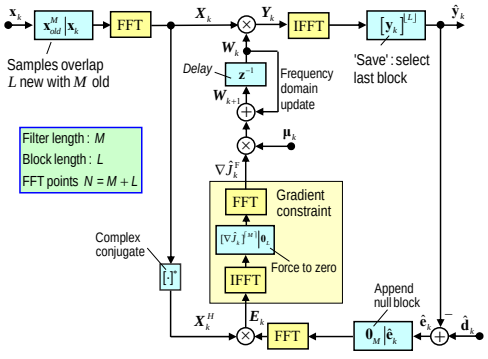


Fig. 7.10 Overlap-save FDAF (OS-FDAF) algorithm structure, also known as fast block LMS (FBLMS). The FFT is calculated for each signal block for which the algorithm introduces a systematic delay of (at least) L samples. In total, the OS-FDAF requires five N-points FFT calculation. (Modified from [11]).

In implementation, the constraint matrix $\mathbf{G}$ does not appear explicitly (i.e., the matrix can not be pre-calculated) is used instead, the FFT. In fact, with its explicit

determination, we would lose the computational cost reduction inherent in the FFT calculation.

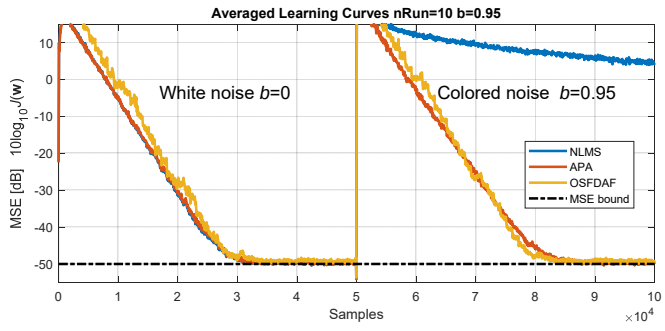
Let $\mathbf{e}_k, \mathbf{d}_k \in \mathbb{R}^{L\times 1}$ the time domain error and desired output of the k-th samples block, from the implementation point of view, the algorithm can be realized as follows.

1. Initialization $\boldsymbol{W}_0 = \mathbf{0}$, $P_0(m) = \delta_m \ \forall m$
2. for $k = 0,1, \ldots \{$ *for each L-samples block*

$$\mathbf{x}_k \leftarrow [\mathbf{x}_{old}^{(M)} \ \mathbf{x}_{new}^{(L)}] \quad \textit{buffer composition rule}$$
$$\boldsymbol{X}_k = \mathrm{FFT}\,[\mathbf{x}_k] \quad \textit{fast Fourier transform}$$
$$\mathbf{y}_k = [\mathrm{IFFT}(\boldsymbol{X}_k \odot \boldsymbol{W}_k)]^{\lfloor L \rfloor} \quad \textit{convolution}$$
$$\boldsymbol{E}_k = \mathrm{FFT}\,([\mathbf{0} \ \ \mathbf{d}_k - \mathbf{y}_k]) \quad \textit{frq. domain error}$$
$$P_k(m) = \lambda P_{k-1}(m) + (1-\lambda)\,|X_k(m)|^2, \ \forall m$$
$$\boldsymbol{\mu}_k = \left[P_k^{-1}(0) \ P_k^{-1}(1) \ \cdots \ P_k^{-1}(N-1)\right]^T$$
$$\nabla J = \boldsymbol{\mu}_k \odot \boldsymbol{X}_k^{\star} \odot \boldsymbol{E}_k \quad \textit{stochastic gradient}$$
$$\nabla J = \mathrm{FFT}\left(\begin{bmatrix} \mathrm{IFFT}\,[\nabla J]^{\lceil M \rceil} \\ \mathbf{0}_L \end{bmatrix}\right) \quad \textit{grad. constraint}$$
$$\boldsymbol{W}_{k+1} = \boldsymbol{W}_k + \nabla J \quad \textit{frq. domain up-date rule}$$

}.

In Fig. 7.11 there is a comparison between three of the most used algorithms in the audio scenario, the NLMS, the APA and OS-FDAF. To evaluate the averaged learning curve The experiment is repeated 10 times so that the total number of processed samples is equal to 10^6.




```
AF performance,  M=1024, nOfSamples=10^6
    NLMS    Elapsed time:  12.504 [s]
     APA    Elapsed time: 103.231 [s]
  OSFDAF    Elapsed time:   0.233 [s]
---------------------------------------
Bias vs variance
Alg:   NLMS  <Var> = 1.05e-03  <Bias> = 9.39e-06
Alg:    APA  <Var> = 2.68e-08  <Bias> = 3.26e-07
Alg: OSFDAF  <Var> = 7.35e-08  <Bias> = 4.60e-08
```

Fig. 7.11 Adaptive filter comparison performance in term of convergence performance, bias and variance of the estimation and computation time (averaged over 10 runs). The target filter length is $M = 1024$ randomly generated, and the number of signal samples is equal to $N = 10^5$. The adaptation steps-size are adjusted to achieve the best and identical performance in the case of the white input. The first half of the signal is WGN with unit variance, while the second half is a narrow-band colored noise generated with Eqn. (7.36) for $b = 0.95$.

From the learning curve we can observe that the convergence characteristics of the OS-FDFA are those typical of a second order algorithm, in fact the performances are similar in the case of white and colored input. However, the processing time (using a MATLAB program run on a laptop) of the OS-FDAF is 50 times faster than the LMS and more than 400 times faster than the APA.

7.5.4 Partitioned OS-FDAF

As for fast frequency domain FIR filtering (see §4.6.2.1 and §4.6.2.2), an alternative implementation of FDAF suitable to decrease of the block length is to partition the filter impulse response in P subfilters [24], [25], [60]-[62].

Let us consider the implementation of a filter of length equal to $M_F = P \cdot M$, the convolution is implemented in P smaller convolutions, each of these implemented in the frequency domain. With this type of implementation, the frequency domain approach advantages are associated with a significant latency reduction. The PFDAF algorithm structure is shown in Fig. 7.12.

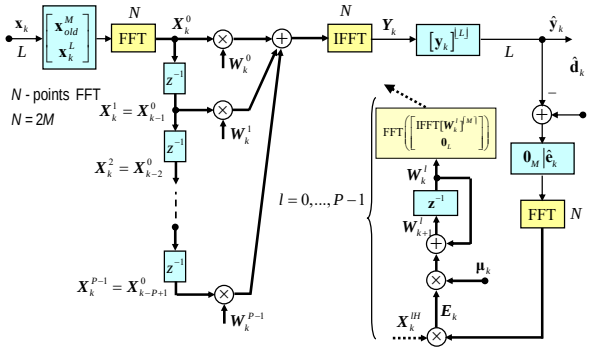


Fig. 7.12 Structure of the PFDAF algorithm, also known as PFBLMS, for $L = M$. (Modified from [11]).

7.6 Direct Acoustic Modeling

The applicability of adaptive filtering methods to audio problems is very wide. However, the various AF applications for DASP can be summarized in the following tasks:

1. Direct modeling;
2. Inverse modeling.

7.6.1 Room Impulse Response Estimation

The quality of a listening room depends on the characteristics of its reverb. Therefore, *Room Impulse Response* (RIR) estimation is of great importance in acoustic modeling.

In general, the estimation of the impulse response can be performed with the scheme indicated in Fig. 3.10. The environment is excited with a broadband signal, as the PRBS (see Appendix A), usually radiated by an array of loudspeakers placed in a regular dodecahedron-shaped encloser, in order to approximate an isotropic source. An omnidirectional microphone acquires the signal and its reflections.

On the contrary, in online estimation using the signal currently on the speaker, e.g. voice music, the probably non-Gaussian statistical characteristics of the excitation signal should be taken into account. In these cases, a widely used algorithm is the APA or a variant of it, which allows scaling of the Hessian information.

7.6.2 Acoustic Echo Cancellation

In the modern hands-free voice over IP (VoiIP) systems, the echo is generated by the coupling between the loudspeaker and the microphone. The problem of acoustic echo can be easily explained by considering the typical teleconference scenario shown in Fig. 7.13. The microphone, as well as capture the voice of the subject, also acquires the signal from the loudspeaker which, together with walls reflections, it is be returned to the sender (far-end side). So, at the sender side there will have a return echo that can seriously affect the intelligibility of communication.

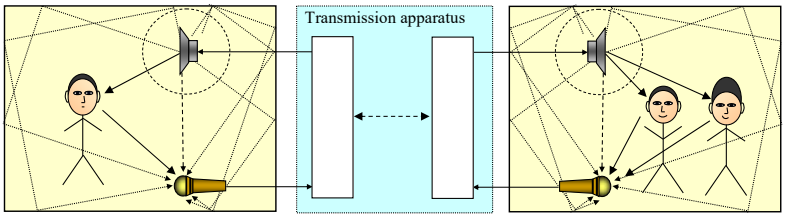


Fig. 7.13 Acoustic echo generation in teleconference scenario. The microphone, in addition to capturing the voice of the speaker, also acquires the signal coming from the loudspeaker and the various reflections due to the walls of the room (reverberation). (Courtesy of [11]).

The prevention of the echo return is therefore of central importance for the quality of the transmission itself and can be performed in various modes. In specific video conference rooms, as in the television studios where it is possible (or desirable) to intervene acoustically, we can use unidirectional microphones or microphone array beam oriented towards the talker (or wireless body or headworn mics, etc.), appropriately position the loudspeaker and treat the room with sound absorbing material in order to make it the most anechoic as is possible.

It is evident that in most real world situations, in living-room, offices, cars, etc., those remedies, which however do not guarantee the complete absence of echo, are in practice impossible. In most practical cases a sophisticated acoustic treatment is unthinkable, in addition, the use of directional microphones together with the correct positioning of loudspeakers strongly binds the speaker to assume predetermined fixed positions.

The echo cancellation can be performed with an AF, placed in parallel to the respective transmission sides, said *adaptive echo canceller* (AEC). With reference to Fig. 7.14 the far-end signal $x[n]$ transmitted by the other side is filtered and subtracted from the microphone signal indicated as $d[n]$. The purpose of the adaptive filter and to model the acoustic path between the loudspeaker and the microphone in order to subtract from the signal to be transmitted $d[n]$, the far-end signal $x[n]$ together with all the reflections due to the walls of the room.

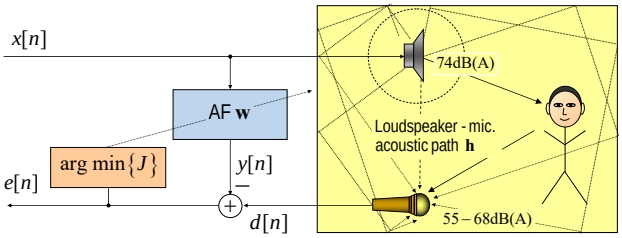


Fig. 7.14 The acoustic path between the loudspeaker and microphone is modelad by room impulse response (RIR) **h**. (Courtesy of [11]).

Remark 7.16. Note that, although conceptually simple, acoustic echo cancellation reveals a problem of a certain complexity. A typical office room reverberation is of the order of hundreds of ms. For example, considering a sampling frequency of 16kHz and a reverberation time of 100-200ms, for the cancellation of the echo effect, the filter should have a length of not less than 1600-3200 coefficients (taps). Moreover, if the speaker is not in a fixed position but moves relative to the microphone and the walls, the acoustic configuration changes continuously. So, the adaptive filter must perform a real time tracking of acoustics variant of the system and, in these cases, the efficiency of the adaptation algorithm, in terms of convergence speed, plays a fundamental role.

Other aspects of current research on acoustic echo cancellation, concern the extension to multi-channel case, i.e, when there are multiple loudspeakers and/or more microphones. In this context, we think that the inclusion of the positional audio paradigm, can be used in order to make video-conference a more natural communication systems (augmented reality). In this class of systems, at the position of the talker on the video, also corresponds a positional acoustic model. To have an adequate acoustic spatiality are necessary, as in the simple stereo case, at least two microphones and two loudspeakers appropriately driven.

The acoustic transduction devices, such as microphones and especially loudspeakers, are by their nature (sometimes strongly) nonlinear and, for this reason, the adaptive acoustic systems that use such devices should take into account of such nonlinearity. In the case of echo cancellation and even, as we shall see in the next section, in the active noise cancellation is almost always considered the hypothesis of linear acoustic transducer. The treatment of nonlinearity, in particular those of the loudspeaker, is a very promising active area of current research. Such nonlinearity, are dynamic, of difficult modeling and, in addition, negligible only in high cost devices.

7.6.3 MIMO Extentions

7.6.4 Limite delle tecniche adattative in campo audio: la funzione di coerenza

7.7 Inverse Acoustic Modeling

Application : room equalization, cross-talk cancellation

7.7.1 Delayed Learning LMS Algorithms

7.7.2 Filtered-X LMS Algorithm

Let $S(z)$ an unknown linear system, also called *secondary path*, the problem of *predistortion* consists in the estimation of its inverse model $W(z)$, also called *controller*, when it is located upstream with respect to the secondary path. In the case of linear transfer function the estimation of predistorter can be solved is similar way of the equalization problem [6].

Considering the diagram of Fig. 7.15-a), we want to estimate the controller transfer function such that $W(z) \to 1/S(z)$, for the commutative property we have that $W(z)S(z) = S(z)W(z) = 1$. In the case where the transfer function $S(z)$ is *a priori* known, for the estimate of $W(z)$ is sufficient to use the scheme of Fig. 7.15-b), and in the normal operation re commute the two transfer functions.

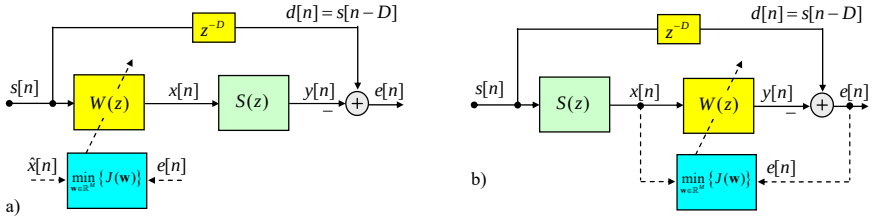


Fig. 7.15 Inverse modeling of secondary path $S(z)$. In the case of linear transfer function the upstream estimation schemes can be solved by switching the two transfer functions. a) inverse modeling by upstream predistortion scheme; b) inverse modeling by downstream equalization scheme.

Usually the transfer function $S(z)$ is unknown, and it is necessary to proceed to an estimate $\hat{S}(z)$. When the secondary path is easy accessible, and presumably stationary, its estimate can be made before the adaptation process (or calibration process). In this case, the estimates of $W(z)$ can be made, by LMS like algorithms, with the diagram in Fig. 7.15-b), using the secondary path estimation $\hat{S}(z)$.

In many practical cases, the secondary path is not simply accessible and also it can be a nonstationary system. So, for its estimate we may proceed with an on-line

approach as, for example, that in the scheme shown in Fig. 7.16, denoted as *filtered-x least mean squares* (FX-LMS). In FX-LMS the input signal is filtered with the estimated $\hat{S}(z)$ (from which the name "filtered-x"), in this way it is produced the signal $\hat{x}[n]$ such that the adaptation of $W(z)$ can be traced to the switched scheme illustrated in Fig. 7.15-b). Thus in FX-LMS the on-line learning rule is

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \mu\hat{\mathbf{x}}e[n].$$

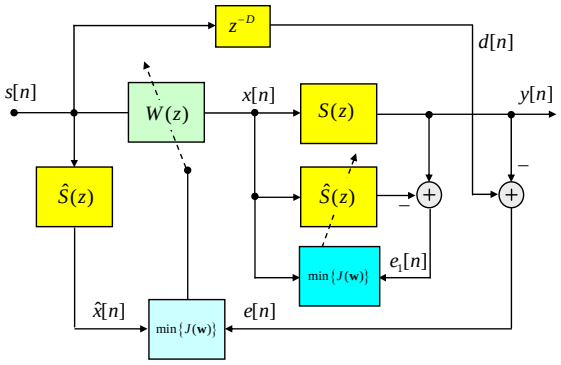


Fig. 7.16 Block diagrams of FX-LMS algorithm for $S(z)$ system compensation by predistortion. The filter $W(z)$ is adapted considering the input signal $\hat{x}[n]$ and the output error $e[n]$.

Example 7.1. Below is a MATLAB code that implements the FX-LMS algorithm, in the case that we proceed to the on-line estimation both of the secondary path and the controller.

The adaptive controller $W(z)$ is implemented with the structure `FXLMS_ W` (see the code). Moreover, its filter's weights are initialized as $\mathbf{w} = \delta[0]$. After the creation of the filter's structure as `FXLMS_ W = create_ STRC_ LMS_ AF(Mw, 0.001);` the first element of the vector $\mathbf{w}$ is forced as $w[1] = 1$, i.e. `FXLMS_ W.w(1) = 1`.

With this initial condition (I.C.), the filter output $W(z)$ is identical to its input. For which if the block (see the parameter `BlkSize` in the MATLAB code) is long enough, in the first iteration the secondary path $S(z)$ is fed with white noise and this facilitates the estimation procedure of the $S(z)$ (see the line code `[LMS_ s0] = AF_ STRC_ LMS_ F_ B(LMS_ s0, BlkSize, x, y(nn));`).

The Fig. 7.17, shows the results of the FX-LMS algorithm in the case in which the secondary path is random modeled as

$$s_0[n] = e^{-0.1n} \cdot \eta[n], \quad n = 0,\ 1,\ \ldots\ M_{s0} - 1$$

where $\eta[n]$ is WGN.

In the figure we can observe that the error signal $e[n]$ tend to zero. Moreover, note that $h[n]$ is the convolution between $s[n]$ and $w[n]$ and, as we can see from the figure, it is very close to a translate unitary impulse such that $H(z) \sim z^{-D}$.

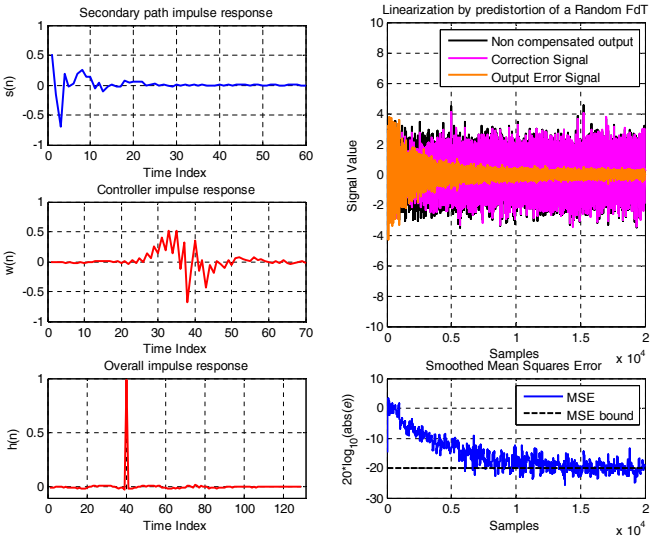


Fig. 7.17 Results of the FX-LMS algorithm. $s[n]$ is the impulse response of secondary path, $w[n]$ is the impulse response of estimated controller and $h[n] = s[n] * w[n] \sim \delta[n - D]$ (in the simulation $D = 40$).

7.7.3 Active Noise Control by Filtered-X LMS Algorithm

The *active noise cancellation* or *active noise control* (ANC) consists in producing of an acoustic wave, said *antinoise*, in phase opposition with respect to the wave generated by the noise source. This wave has the objective of creating of a silence zone in a given region of space [26].

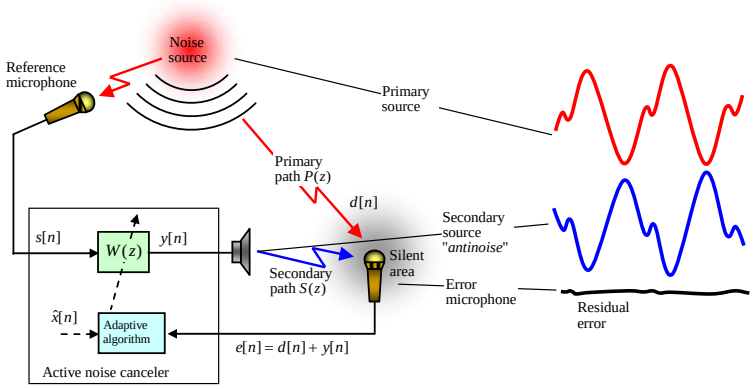


Fig. 7.18 Diagram of the operating principle of an *active noise canceler* (ANC). The loudspeaker makes a wave in phase opposition with respect to the noise at the point where is located the *error microphone*. The reference microphone should be placed as close as possible to the acoustic source of noise.

Referring to the operating principle that is shown in the Fig. 7.18, the first task in ANC is to estimate the impulse response of the *secondary propagation path* denoted as $S(z)$. This step is usually performed prior to noise control using a synthetic random signal, as for example WGN or *pseudo random binary sequences*[5] (PRBS), played through the loudspeaker while the unwanted noise is not present [26]. The transfer function $S(z)$ is the model between the loudspeaker, that produce the anti-noise, and error microphone. Note that the estimate of the secondary path must necessarily be made when the primary noise source is off. Unlike the generic predistorter, in the case of the ANC this stage can not be done online. In addition, always referring to Fig. 7.18, the transfer function $P(z)$ is the model between the noise source and the error microphone and represents the *primary acoustic path*.

It is easy to verify that the schematization the ANC operating principle shown in the Fig. 7.18 is that reported in Fig. 7.19. The ANC scheme is very similar to the scheme of the predistorter of Fig. 7.16. The only difference is that, in ANC case, $P(z)$ is an *acoustic transfer function* (ATF) that model the distance and the riverberation between the primary acoustic noise and the error microphone. Thus, in order to create a silent area, we need to determine a controller $W(z)$ that has a transfer function such that at the signal at the error microphone is minimized, in other word such that $W(z)S(z) \sim P(z)$.

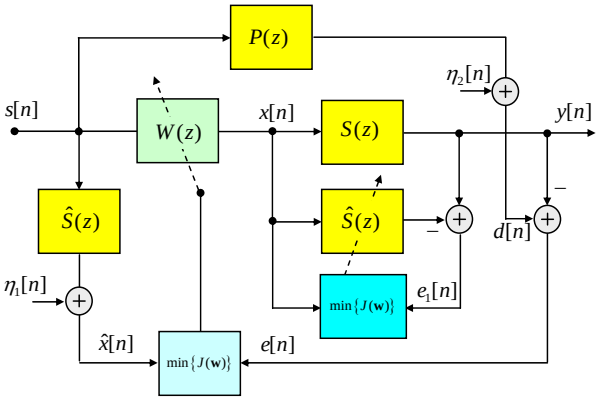


Fig. 7.19 Block diagrams of FxLMS algorithm for for active noise control. The transfer function $P(z)$ is the primary acoustic propagation path. In the ANC simulation, the noise $\eta_1(n)$ taking account of any error of the secondary path estimates, while $\eta_2(n)$ is the not cancellable noise.

Example 7.2. In ANC simulation systems, in order the generate the primary acoustic propagation, we may consider the frequency range 50-3000 [Hz], and that for the filter length we may consider an impulse response of duration in the range 0.1 - 0.3 [s]. For example, considering a duration of 0.1 [s], for a sampling frequency of 8 [kHz], the impulse response has length equal to 800 samples. In addition, note in the secondary path we must also consider the loudspeaker's frequency response that, in our tests, was

[5] A *pseudo random binary sequence* is a 2-level signal (i.e. binary) that has statistical behavior similar to a truly-random sequence. It is used in identification techniques in that it has a constant amplitude and, therefore, can work in a zone in which the system the to be identified and the device for generating the excitation source (e.g. a loudspeaker), are linear.

chosen in the range 30 - 1000 [Hz]. In the simulation, the estimation of the secondary path has been made prior to the noise control phase, using a WGN signla with a duration of 2 sec.

In the tests, to simulate the noise source, it has been considered as a sum of sinusoids with random frequency, generated in the range between 50 and 720 [Hz].

The results of the simulation is reported in Fig. 7.20.

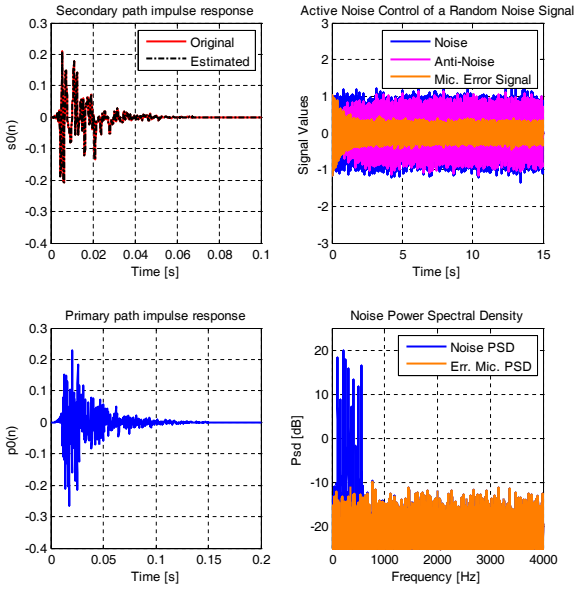


Fig. 7.20 Active noise control by FxLMS algorithm. Simulation results.

7.7.4 ANC by Adjoint LMS Algorithm

The adjoint LMS (AD-LMS) algorithm, represents an alternative way of FX-LMS algorithm for the controller adaptation in predistortion and ANC systems. Referring to Fig. 7.21 the AD-LMS can be derived by considering three steps.

1. As shown in Fig. 7.21-b) we must switch the adaptation block LMS and the block of the secondary path estimation $S(z)$. In order for this operation to be consistent, as shown in Fig.s 7.22-a) and b), the transfer function path $\hat{S}(z)$ become non causal as $\hat{S}(z) \rightarrow \hat{S}(z^{-1})$, i.e. in the transfer function definition the delays are transformed in anticipation elements $z^{-1} \rightarrow z^{+1}$. This means that the signal $\hat{e}[n]$ is dependent on the future samples of the error output $e[n+k]$.
2. In order to causalize the transfer function $\hat{S}(z^{-1})$, it is sufficient to appropriately delay, the output error signal. This means that $\hat{S}(z^{-1}) \rightarrow z^{-D}\hat{S}(z^{-1})$, where D is a suitable delay

3. Finally, for the correct LMS adaptation the input and the error signals must be perfectly aligned. Thus, as shown in Fig. 7.21-c), is also necessary to delay, of a factor D, the input signal of the LMS block.

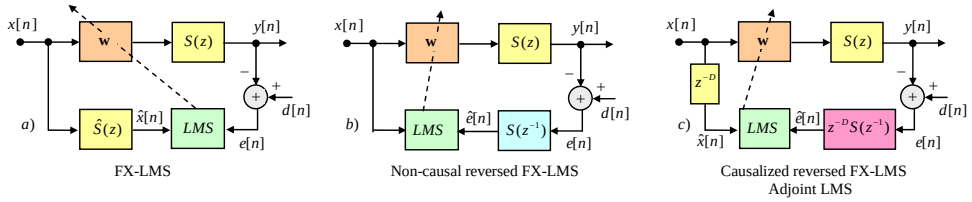


Fig. 7.21 The adjoint LMS AD-LMS algorithm derivation.

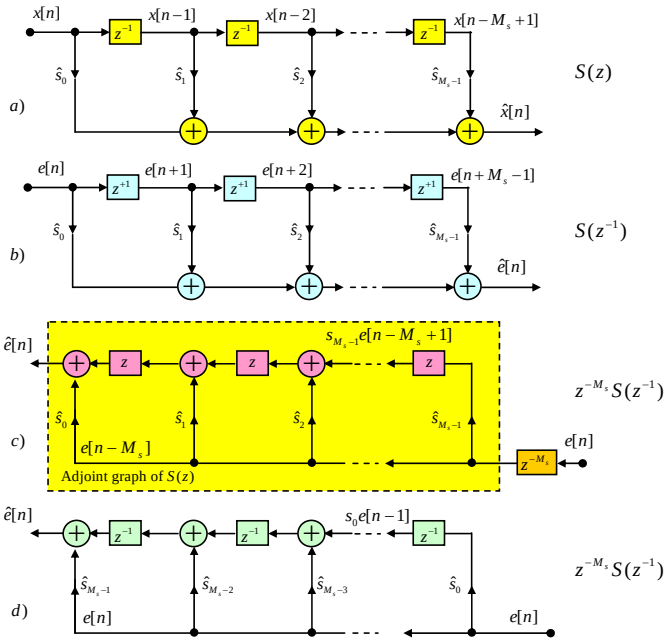


Fig. 7.22 The signal flow graph (SFG) of the error signal filtered by secondary path transfer function. a) SFG of S(z); b) SFG of non causal $S(z^{-1})$. c) Causalized SFG $z^{-M_s}S(z^{-1})$. d) Equivalent causalized SFG, implemented with only the delay elements, but with the filter coefficients arranged in reversed order.

Remark 7.17. By a simple visual inspection of Fig. 7.22-c), we can observe that the SFG of the transfer function in the dotted box corresponds exactly to the so-called

adjoint graph or *adjoint network*[6] of the SFG of $S(z)$ (not reported for brevity). The name of the algorithm adjoint LMS, originally developed in [27], is derived from this property.

In AD-LMS, the LMS update is performed considering a filtered version of the delayed error signal. Let $\hat{\mathbf{s}} = [\hat{s}_0 \;\; \hat{s}_1 \cdots \hat{s}_{M_s-1}]^T$, be the estimated impulse response of the secondary path, in order to have a correct LMS update, the transfer function $\hat{S}(z^{-1})$, in order to compensate the anticipation elements, is fed by a delayed error signal $e[n-D]$. In other word, we have that

$$\frac{\hat{E}(z)}{E(z)} = z^{-D}\hat{S}(z^{-1}).$$

More formally, for $D = M_s - 1$, the filter error transfer function can be written as

$$\begin{aligned} z^{-M_s+1}\hat{S}(z^{-1}) &= z^{-M_s+1}\left(\hat{s}_0 + \hat{s}_1 z^1 + \cdots + \hat{s}_{M_s-1}z^{M_s-1}\right) \\ &= \hat{s}_0 z^{-M_s+1} + \hat{s}_1 z^{M_s} + \cdots + \hat{s}_{M_s-1}z^0 \\ &= \hat{s}_{M_s-1} + \hat{s}_{M_s-2}z^{-1} + \cdots + \hat{s}_0 z^{-M_s+1}. \end{aligned}$$

Note that the last expression corresponds to the equivalent causalized SFG version of $z^{-M_s+1}S(z^{-1})$, implemented with only the delay elements, and with the coefficients arranged in reverse order (see Fig. 7.22-d)). Thus, in the time domain we have that the filtered error $\hat{e}[n]$ should be evaluated as

$$\hat{e}[n] = \sum_{k=0}^{M_s-1} \hat{s}[M_s-k]e[n-k].$$

Moreover, in order to keep the filtered version of the error aligned with the signal that is used for the LMS update, the input signal must by delayed by a delay element z^{-M_s+1}. The resulting algorithm is illustrated in Fig. 7.23. The corresponding AD-LMS adaptation rule can be written as

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu\mathbf{x}_{n-M_s}\hat{e}[n]. \tag{7.72}$$

Example 7.3. Below is reported the MATLAB code of the AD-LMS algorithm for the same problem previously presented in Example 7.2. The simulation results are reported in Fig. 7.24.

[6] Given a discrete-time circuit defined by a graph $\mathcal{G}$, we define the adjoint network a circuit whose graph is determined by $\mathcal{G}$ with the following modifications: 1) the paths verses are reversed; 2) junction nodes are switched with sum nodes; 3) delay elements are replaced with anticipation elements.

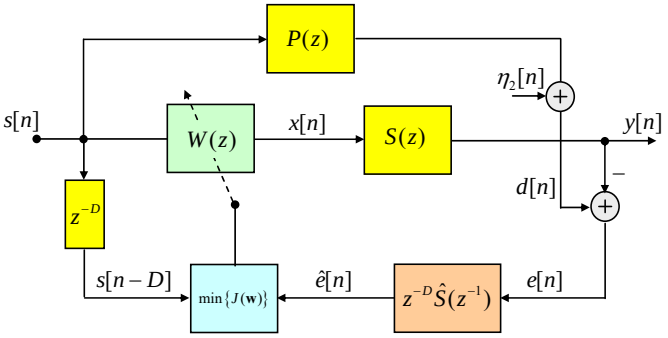


Fig. 7.23 Block diagrams of the adjoint LMS algorithm for active noise control.

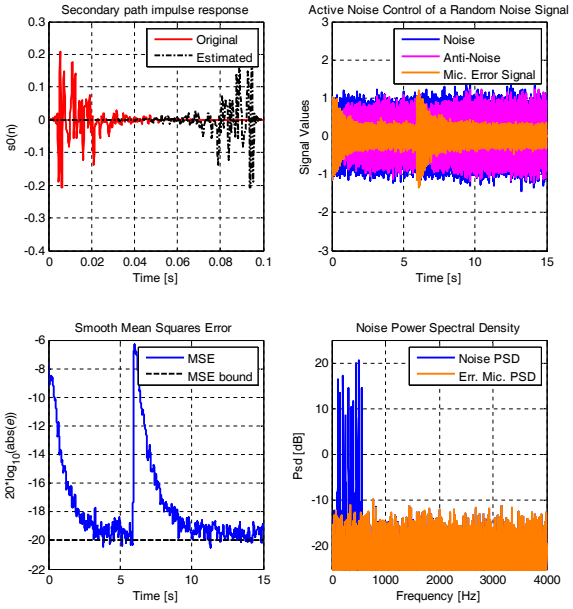


Fig. 7.24 Active noise control by AD-LMS algorithm. Simulation results.

7.7.5 Crosstalk Cancellation

References

1. N. Wiener, “Extrapolation, Interpolation and Smoothing of Stationary Time Series, with Engineering Applications”, New York, Wiley, 1949.
2. N. Wiener and E. Hopf, “On a class of singular integral equations,” Proc. Prussian Acad., Math.-Phys. Ser., p. 696, 1931.
3. T. Kailath, “A View of Three Decades of Linear Filtering Theory,” IEEE Trans. On Information Theory. Vol. IT-20, No 2, pp 146-181, March 1974.
4. G. U. Yule, “On a method of investigating periodicities in disturbed series, with special reference to wolfer’s sunspot numbers,” Phil. Trans. Roy. Soc., 226-A:267–298, 1927.
5. B. Widrow, M.E. Hoff, “Adaptive switching circuits,” IRE WESCON, Conv. Rec., pt. 4, pp. 96-104, 1960.
6. B. Widrow, S.D. Stearns, “Adaptive Signal Processing”, Prentice Hall ed., 1985.
7. N. Ahmed, D.L. Soldan, D.R. Hummels, D.D. Parikh, “Sequential regression Considerations of Adaptive Filter”, IEE Electronics Letters, Vol. 13, No.15 pp. 446-447, July 1977.
8. S.C. Douglas, “Introduction to Adaptive Filters”, Digital Signal Processing Handbook Ed. Vijay K. Madisetti and Douglas B. Williams Boca Raton: CRC Press LLC, 1999
9. S. Haykin, “Adaptive Filter Theory”, Third Edition, Prentice Hall ed., 1996
10. Ali H. Sayed, “Fundamentals of Adaptive Filtering,” IEEE Wiley - Interscience, 2003.
11. A. Uncini, “Fundamentals of Adaptive Signal Processing,” Springer, ISBN: 978-3-319-02806-4, 2015.
12. L. Landweber, “An iteration formula for Fredholm integral equations of the first kind,” Amer. J. Math. 73, 615-624, 1951.
13. D. Marquardt, “An Algorithm for Least Squares Estimation on Nonlinear Parameters”, SIAM J. APPL. MATH.11, pp 431–441, 1963.
14. H.J. Kushner, “Approximation and Weak Convergence Methods for Random Processes, with Applications to Stochastic Systems Theory,” MIT Press, ISBN 0262110903, 9780262110907, 1984.
15. M. Dentino, J. McCool, and B. Widrow, “Adaptive filtering in the frequency domain,” Proceedings of IEEE, Vol. 66, pp. 1658-1660, Dec. 1978.
16. N. J. Bershad and P. D. Feintuch, “Analysis of the frequency domain adaptive fiter,” Proc. IEEE, vol. 67, pp. 1658 - 1659, Dec. 1979.
17. E. R. Ferrara, “Fast implementation of LMS adaptive filters,” IEEE Transactions Acoustic Speech, and Signal Processing, Vol. ASSP-28, pp. 474-475, Aug. 1980.
18. G.A. Clark, S.K. Mitra, S.R. Parker, “Block Implementation of Adaptive Digital Filters,” IEEE Transactions on Circuits and Systems, Vol. CAS-28, No. 6, pp. 584-592, June 1981.
19. G.A. Clark, S.R. Parker, S.K. Mitra, “A Unified Approach to Time- and Frequency-Domain Realization of FIR Adaptive Digital Filters,” IEEE Transactions Acoustic Speech, and Signal Processing, Vol. ASSP-31, No. 5, pp. 1073-1083, October 1983.
20. S.S. Narayan and A. M. Peterson, “Frequency Domain Least-Mean Square Algorithm,” Proceedings of IEEE, Vol. 69, No. 1, pp. 124-126, January 1981.
21. J.C. Lee and C.K.Un, “Performance analysis of frequency-domain block LMS adaptive digital filters,” IEEE Trans. Circuits Syst., vol. 36, pp. 173-189, Feb. 1989.
22. D. Mansour, A.H. Gray, “Unconstrained Frequency-Domain Adaptive Filter,” IEEE Transactions Acoustic Speech, and Signal Processing, Vol. ASSP-30, No. 5, pp. 726-734, October 1982.
23. J.J. Shynk, “Frequency Domain and Multirate Adaptive Filtering,” IEEE Signal Processing Magazine, Vol. pp. 14-37, January 1992.
24. M. R. Asharif, T. Takebayashi, T. Chugo, and K. Murano, “Frequency domain noise canceler: Frequency-bin adaptive filtering (FBAF)”, in Proc. ICASSP, pp. 41.22.1–41.22.4, 1986.
25. B. Farhang-Boroujeny, K.S. Chan, “Analysis of the Frequency-Domain Block LMS Algorithm”, IEEE Transactions on Signal Processing, Vol. SP-48, No. 8, pp. 2332 - 2342, August 2000.
26. S. M. Kuo and D. R. Morgan, “Active Noise Control Systems’ - Algorithms and DSP Implementations” New York: Wiley, 1996.

27. E. A. Wan, “Adjoint LMS: an efficient alternative to the filtered-X LMS and multiple error LMS algorithms,” in Proc. IEEE ICASSP-1996, pp. 1842–1845, 1996.
28. S. Shaffer and C. S. Williams, “The filtered error LMS algorithm,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., Boston, USA, Apr. 1983, pp. 41-44.
29. S. C. Douglas, “An efficient implementation of the modified filtered-x LMS algorithm,” IEEE Signal Process. Lett., vol. 4, no. 10, pp. 286-288, Oct. 1997.
30. S. J. Eliott and P. A. Nelson, “Active noise control,” IEEE Signal Processing Mag., vol. 10, no. 4, pp. 12-35, Oct. 1993.