1

Introduction to PtNLMS Algorithms

The objective of this chapter is to introduce proportionate-type normalized least mean square (PtNLMS) algorithms in preparation for performing analysis of these algorithms in subsequent chapters. In section 1.1, we begin by presenting applications for PtNLMS algorithms as the motivation for why analysis and design of PtNLMS algorithms is a worthwhile cause. In section 1.2, a historical review of relevant algorithms and literature is given. This review is by no means exhaustive; however, it should serve the needs of this work by acting as a foundation for the analysis that follows. In section 1.3, standard notation and a unified framework for representing PtNLMS algorithms are presented. This notation and framework will be used throughout the remainder of this book. Finally, with this standardized notation and framework in hand, we present several PtNLMS algorithms in section 1.4. The chosen PtNLMS algorithms will be referenced frequently throughout this work.

1.1. Applications motivating PtNLMS algorithms

Historically, PtNLMS algorithms have found use in network echo cancellation applications as a method for reducing the presence of delayed copies of the original signal, i.e. echoes. For instance, in many telephone communication systems the network consists of two types of wire segments: a four-wire central network and a two-wire local network [MIN 06]. A converting device, called a hybrid, is needed at the junction of the two-wire to four-wire segments. When a far-end user talks a portion of the signal is reflected back to the far-end listener, due to the impedance mismatch in the hybrid as shown in Figure 1.1. This type of echo is called electric echo or circuit echo. An adaptive filter can be used to estimate the impulse response of the hybrid and remove the echo caused by the hybrid.

In modern telephone networks, greater delays increase the need for echo cancellation. Specifically, these communication networks have driven the need for faster converging echo cancellation algorithms when the echo path is sparse. A sparse echo path is that in which a large percentage of the energy is distributed to only a few coefficients. Conversely, a dispersive echo path has distributed most of its energy more or less evenly across all of the coefficients. Examples of a dispersive impulse response and a sparse impulse response are shown in Figures 1.2a and 1.2b, respectively. While most network echo path cancelers have echo path lengths in the order of 64 ms, the active part of the echo path is usually only about 4–6 ms long [GAY 98], hence the echo path is sparse. The active part of an echo path corresponds to the coefficients of the echo path that contain the majority of the energy. When the impulse response is sparse, PtNLMS algorithms can offer improved performance relative to standard algorithms such as the least mean square (LMS) and normalized least mean square (NLMS) [HAY 02].

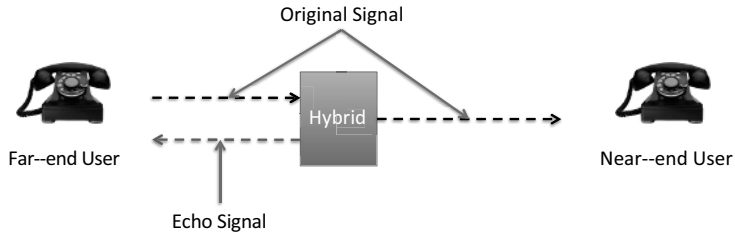
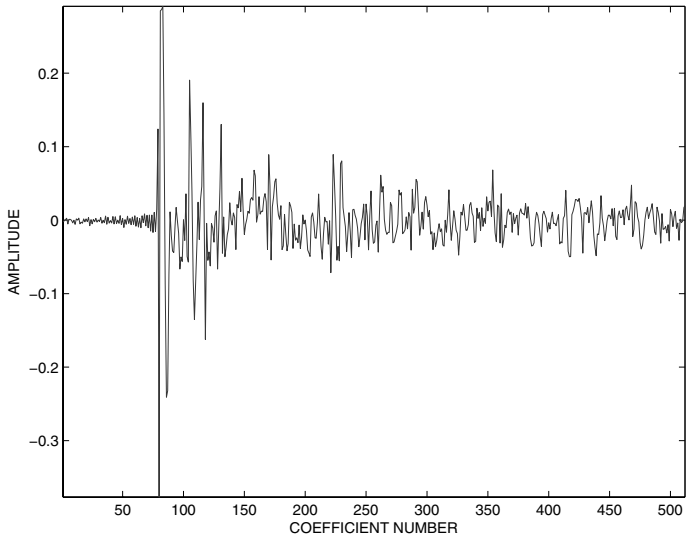


Figure 1.1. *Telephone echo example*

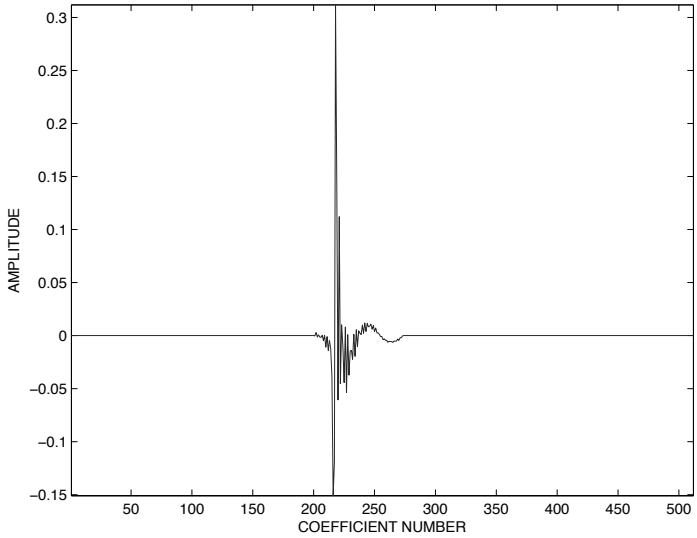
Another application for PtNLMS has been spawned by the emergence of Voice over IP (VOIP) as an important and viable alternative to circuit switched networks. In these systems, longer delays are introduced due to packetization [SON 06]. In addition, echoes can be created during the transition from traditional telephone circuits to IP-based telephone networks [MIN 06].

The advent of telephone communication via satellite has motivated the search for a better way to eliminate echoes [WEI 77]. The distortion caused by echo suppressors is particularly noticeable on satellite-routed connections.

Let us also mention that modern applications of echo cancellation include acoustic underwater communication where the impulse response is often sparse [STO 09], television signals where the delay can be significant due to satellite communication [SCH 95], high-definition television terrestrial transmission that requires equalization due to inter-symbol interference caused by multi-path channels exhibiting sparse behavior [FAN 05], and applications where the impulse response is sparse in the transform domain [DEN 07].



a) Dispersive impulse response



b) Sparse impulse response

Figure 1.2. *Dispersive and sparse impulse responses*

When examining these applications as well as others like them, several questions regarding the desired performance of the PtNLMS algorithms need to be answered in order to design an algorithm for the intended application. For instance, we need to know what the required convergence of the algorithm needs to be. We also need to know the computational complexity that the application can support as well as what level of steady-state bias can be tolerated. To address these questions we need to understand the underlying mechanics of each PtNLMS algorithm as well as what factors control how the algorithms perform. Therefore, we intend to analyze PtNLMS algorithms to find out what factors influence the convergence, steady-state performance, and what the implementation cost of possible improvements in terms of computational complexity is. In doing so, a better understanding of what influences the performance of PtNLMS algorithms is provided. Answering these questions will allow us to design algorithms that perform their desired tasks more efficiently.

1.2. Historical review of existing PtNLMS algorithms

In the past, adaptive filtering algorithms such as the LMS and NLMS have been examined extensively [HAY 02]. These algorithms offer low computational complexity and proven robustness. The LMS and NLMS algorithms share the property of the adaptive weights being updated in the direction of the input vector. These algorithms perform favorably in most adaptive filtering situations.

The LMS and NLMS algorithms fall within the larger class of PtNLMS algorithms. PtNLMS algorithms can update the adaptive filter coefficients such that some coefficients are favored. That is, some coefficients receive more emphasis during the update process. Because of this fact, the PtNLMS algorithms are better suited to deal with sparse impulse responses.

An example of a PtNLMS algorithm is the proportionate normalized least mean square (PNLMS) algorithm [DUT 00]. This algorithm updates the adaptive filter coefficients by assigning a gain proportional to the magnitude of the current coefficient estimates. This approach improves the initial convergence rates. However, this gain adaptation scheme causes the PNLMS algorithm to suffer from slow convergence of small coefficients, and as a result the time needed to reach steady-state error is increased compared to the NLMS algorithm. The PNLMS algorithm has been shown to outperform the LMS and NLMS algorithms when operating on a sparse impulse response. Currently, any analytical model to describe learning curve convergence of the PNLMS algorithm does not exist [SON 06].

Another weakness of the PNLMS algorithm is that when the impulse response is dispersive, the algorithm converges much slower than the NLMS algorithm. To remedy this issue the PNLMS++ was proposed [GAY 98]. The PNLMS++ algorithm solves this issue by alternating between the NLMS and PNLMS algorithms on each sample period of the algorithm. The improved PNLMS (IPNLMS) was introduced to

build upon the PNLMS++ algorithm [BEN 02]. The IPNLMS attempts to exploit the shape of the estimated echo, instead of blindly alternating between the PNLMS and NLMS algorithms as is done in the PNLMS++ algorithm. The IPNLMS algorithm performs better than the NLMS and PNLMS algorithms no matter what the nature of the impulse response.

In the following, the improved IPNLMS (IIPNLMS) algorithm was proposed [CUI 04]. This algorithm attempted to identify active and inactive regions of the echo path impulse response. Active regions received updates more in-line with the NLMS methodology, while inactive regions received gains based upon the PNLMS methodology. In this way, the IIPNLMS was able to outperform the IPNLMS algorithm. The idea of applying gain selectively to active and inactive regions was explored previously in the so-called partial update algorithms. These algorithms, motivated by reducing computational complexity while improving performance, update a subset of all the coefficients during each sampling period. Examples of partial update NLMS algorithms can be found in [TAN 02] and [NAY 03].

Another set of algorithms was designed by seeking a condition to achieve the fastest overall convergence. Initially, the steepest descent algorithm was optimized and then the resulting deterministic algorithm was cast into the stochastic framework. It can be shown that the total number of iterations for overall convergence is minimized when all of the coefficients reach the ϵ-vicinity of their true values simultaneously (where ϵ is some small positive number). This approach results in the μ-law PNLMS (MPNLMS) [DEN 05]. The MPNLMS algorithm addresses the issue of assigning too much update gain to large coefficients, which occurs in the PNLMS algorithms.

The ϵ-law PNLMS (EPNLMS) [WAG 06] algorithm is a second implementation of the same philosophy used to generate the MPNLMS algorithm. This algorithm gives the minimum gain possible to all of the coefficients with magnitude less than ϵ. It assumes that the impulse response is sparse and contains many small magnitude coefficients [DEN 06]. The EPNLMS outperforms the MPNLMS algorithm in many cases, however the MPNLMS algorithm's performance is more robust regarding the choice of algorithm parameters, as well as input signal and unknown system characteristics, than the EPNLMS algorithm.

The individual activation factor PNLMS (IAF-PNLMS) algorithm was introduced in [DAS 10]. The standard PNLMS algorithm performance depends on some predefined parameters controlling proportionality activation through a minimum gain that is common for all of the coefficients. In contrast, the IAF-PNLMS algorithm computes a separate minimum gain for each coefficient. This time varying minimum gain is called the activation factor and has the following characteristics: (1) an individual activation factor is used for each adaptive filter coefficient; (2) each individual activation factor is computed in terms of the past and current values of the corresponding coefficient magnitude, thereby each activation factor presents some

inherent memory associated with its corresponding coefficient magnitude; (3) the individual activation factors do not rely on the proportionality and initialization parameters, since they are no longer in the proposed formulation. As a consequence, the convergence features of the IAF-PNLMS algorithm are improved relative to the NLMS and PNLMS algorithms.

1.3. Unified framework for representing PtNLMS algorithms

We begin by introducing a unified framework for representing PtNLMS algorithms. All signals are real-valued throughout this chapter and the majority of this book. It will be stated explicitly if the signals under examination are complex. Let us assume there is some input signal denoted as $\mathbf{x}(k)$ for time k that excites an unknown system with impulse response $\mathbf{w}$. Let the output of the system be $y(k) = \mathbf{w}^T\mathbf{x}(k)$, where $\mathbf{x}(k) = [x(k), x(k-1), \ldots, x(k-L+1)]^T$ and L is the length of the filter. The measured output of the system, $d(k)$, contains measurement noise $v(k)$ and is equal to the sum of $y(k)$ and $v(k)$. The impulse response of the system is estimated with the adaptive filter coefficient vector (also called weight vector), $\hat{\mathbf{w}}(k)$, which also has length L. The outputs of the adaptive filters is given by $\hat{y}(k) = \hat{\mathbf{w}}^T(k)\mathbf{x}(k)$. The error signal $e(k)$ between the outputs of the adaptive filters $\hat{y}(k)$ and $d(k)$ drives the adaptive algorithm. A diagram of this processing scheme is shown in Figure 1.3. The weight deviation vector is given by $\mathbf{z}(k) = \mathbf{w} - \hat{\mathbf{w}}(k)$.

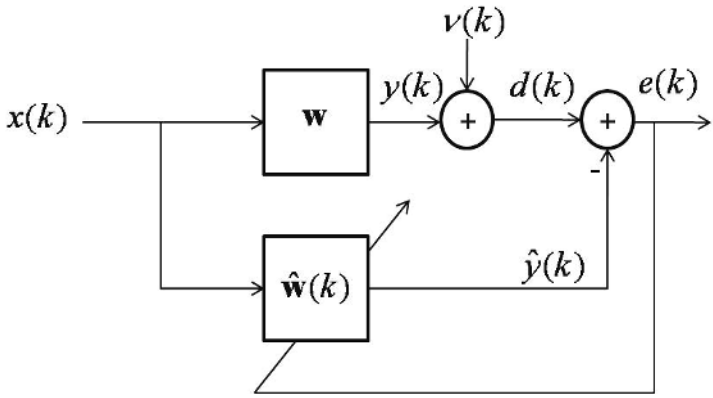


Figure 1.3. *Adaptive filtering "system identification" configuration*

The PtNLMS algorithm is shown in Table 1.1. Here, β is the fixed step-size parameter. The term $F[|\hat{w}_l(k)|, k]$, with $l \in \{1, 2, \ldots, L\}$, governs how each coefficient is updated and we refer to this term as the control law. In the case when $F[|\hat{w}_l(k)|, k]$ is less than $\gamma_{\min}$, the quantity $\gamma_{\min}$ is used to set the minimum gain a coefficient can receive. The constant δ_p, where $\delta_p \geq 0$, is important in the beginning of learning when all of the coefficients are zero and together with ρ, where $\rho \geq 0$,

prevent the very small coefficients from stalling. $\mathbf{G}(k) = \mathbf{Diag}\ \{g_1(k), \ldots, g_L(k)\}$ is the time-varying step-size control diagonal matrix. The constant δ is typically a small positive number used to avoid division by zero if the inputs are zero, that is when $\mathbf{x}(k) = \mathbf{0}$.

$\mathbf{x}(k)$	$= [x(k), x(k-1), \ldots, x(k-L+1)]^T$
$\hat{y}(k)$	$= \mathbf{x}^T(k)\hat{\mathbf{w}}(k)$
$e(k)$	$= d(k) - \hat{y}(k)$
$F[\lvert\hat{w}_l(k)\rvert, k]$	= Specified by the user
$\gamma_{\min}(k)$	$= \rho \max\{\delta_p, F[\lvert\hat{w}_1(k)\rvert, k], \ldots, F[\lvert\hat{w}_L(k)\rvert, k]\}$
$\gamma_l(k)$	$= \max\{\gamma_{\min}(k), F[\lvert\hat{w}_l(k)\rvert, k]\}$
$g_l(k)$	$= \frac{\gamma_l(k)}{\frac{1}{L}\sum_{i=1}^{L}\gamma_i(k)}$
$\mathbf{G}(k)$	$= \mathbf{Diag}\{g_1(k), \ldots, g_L(k)\}$
$\hat{\mathbf{w}}(k+1)$	$= \hat{\mathbf{w}}(k) + \frac{\beta\mathbf{G}(k)\mathbf{x}(k)e(k)}{\mathbf{x}^T(k)\mathbf{G}(k)\mathbf{x}(k)+\delta}$

Table 1.1. *PtNLMS algorithm with time-varying step-size matrix*

Some common examples of the term $F[|\hat{w}_l(k)|, k]$ are $F[|\hat{w}_l(k)|, k] = 1$ and $F[|\hat{w}_l(k)|, k] = |\hat{w}_l(k)|$, which result in the NLMS and PNLMS algorithms, respectively.

Assuming that the impulse response being estimated is sparse, the PtNLMS algorithms begin by setting $\hat{\mathbf{w}}(0) = \mathbf{0}$. These algorithms rely on the fact that the true system impulse response $\mathbf{w}$ is sparse and most coefficients are zero, therefore the initial estimate $\hat{\mathbf{w}}(0) = \mathbf{0}$ is correct for most of the estimated weights. In the following, the weights that differ from zero should be driven toward their true values as quickly as possible to speed up convergence. The question, as proposed in [DEN 05], then becomes how to determine when an estimated coefficient's true value is non-zero and how to assign gain to all of the coefficients in a manner that increases the convergence rate of the overall algorithm. These two issues give rise to the question of switching criteria within the context of the overall control law. The control law determines how to assign gain to each of the estimated coefficients. This process can be broken into separate steps for most algorithms:

1) The first step does switching in order to separate the estimated coefficients into two categories: those that are near to their true values and those that are not.

2) The second step of the overall control law determines how to assign gain once the coefficients have been separated into two categories based on the switching criterion.

In general, we want to assign the minimal possible gain to all of the coefficients that are near their true values. This law is common throughout almost all of the PtNLMS algorithms. The various algorithms addressed mainly differ in how they

assign gain to the estimated coefficients that are not near their optimal values. That is, the algorithms vary in the specification of the function $F[|\hat{w}_l(k)|, k]$.

1.4. Proportionate-type NLMS adaptive filtering algorithms

In this section, mathematical representations of several PtNLMS algorithms are presented in further detail.

1.4.1. *Proportionate-type least mean square algorithm*

The first algorithm we examine is the PtLMS algorithm. Strictly speaking, the PtLMS algorithm is not a PtNLMS algorithm because the update term for the weight deviation is not normalized by the input signal power. However, the PtLMS algorithm serves as a building block toward the PtNLMS algorithms. The PtLMS adaptive filtering algorithm is presented in Table 1.2. When we set $\mathbf{G}(k) = \mathbf{I}$ for all k, where $\mathbf{I}$ is the identity matrix, then the PtLMS algorithm reduces to the widely known LMS [HAY 02] algorithm.

$\mathbf{x}(k)$	=	$[x(k), x(k-1), \ldots, x(k-L+1)]^T$
$\hat{y}(k)$	=	$\mathbf{x}^T(k)\hat{\mathbf{w}}(k)$
$e(k)$	=	$d(k) - \hat{y}(k)$
$\hat{\mathbf{w}}(k+1)$	=	$\hat{\mathbf{w}}(k) + \beta\mathbf{G}(k)\mathbf{x}(k)e(k)$

Table 1.2. *PtLMS algorithm with time-averaging step-size matrix*

1.4.2. *PNLMS algorithm*

The Proportionate NLMS algorithm was first proposed in [DUT 00]. The control law for these algorithms assigns a gain proportionate to the magnitude of the estimated coefficients,

$$F[|\hat{w}_l(k)|, k] = |\hat{w}_l(k)|,\ 1 \leq l \leq L. \qquad [1.1]$$

The motivation for this algorithm is based on knowledge that the impulse response is sparse. Therefore it is desired to adapt coefficients with large magnitudes faster than those that are at or near zero.

1.4.3. *PNLMS++ algorithm*

The PNLMS++ algorithm introduced in [GAY 98] is a combination of the PNLMS and NLMS algorithms. For instance, one implementation of the PNLMS++

algorithm is to alternate between the NLMS and PNLMS gains logic every iteration. This implementation is shown here:

$$F[|\hat{w}_l(k)|,k] = \begin{cases} |\hat{w}_l(k)|,\ 1 \le l \le L, & \text{if } k \text{ is odd,} \\ 1, & \text{if } k \text{ is even.} \end{cases}$$

An alternative implementation of the PNLMS++ algorithm is to alternate between the NLMS and PNLMS algorithms every Mth iteration.

1.4.4. *IPNLMS algorithm*

The improved PNLMS (IPNLMS) was introduced in [BEN 02] and has the following control law:

$$F[|\hat{w}_l(k)|,k] = (1-\alpha_{\text{IPNLMS}})\frac{||\hat{\mathbf{w}}(k)||_1}{L} + (1+\alpha_{\text{IPNLMS}})|\hat{w}_l(k)|, \qquad [1.2]$$

where $||\hat{\mathbf{w}}(k)||_1 = \sum_{j=1}^{L}|\hat{w}_j(k)|$ is the L_1 norm of the vector $\hat{\mathbf{w}}(k)$ and $-1 \le \alpha_{\text{IPNLMS}} \le 1$. The algorithm uses $\rho = 0$, that is the minimum gain logic introduced in Table 1.1, which in this case, is unnecessary. The components of the time-varying gain matrix are given by:

$$\begin{aligned} g_l(k) &= \frac{F[|\hat{w}_l(k)|,k]}{||F[|\hat{w}_l(k)|,k]||_1} \\ &= \frac{(1-\alpha_{\text{IPNLMS}})}{2L} + (1+\alpha_{\text{IPNLMS}})\frac{|\hat{w}_l(k)|}{2||\hat{\mathbf{w}}(k)||_1}. \end{aligned} \qquad [1.3]$$

However in practice, to avoid division by zero, especially at the beginning of adaptation when the estimated coefficients are all close to zero, a slightly modified form of the time-varying gain matrix is used:

$$g_l(k) = \frac{(1-\alpha_{\text{IPNLMS}})}{2L} + (1+\alpha_{\text{IPNLMS}})\frac{|\hat{w}_l(k)|}{2||\hat{\mathbf{w}}(k)||_1 + \epsilon_{\text{IPNLMS}}}, \qquad [1.4]$$

where ϵ_{IPNLMS} is a small positive number. Note for $\alpha_{\text{IPNLMS}} = -1$ this algorithm reduces to the NLMS algorithm and for $\alpha_{\text{IPNLMS}} = 1$ the IPNLMS algorithm behaves like the PNLMS algorithm.

1.4.5. *IIPNLMS algorithm*

In the following, the improved IPNLMS (IIPNLMS) algorithm was given in [CUI 04]. The components of the time-varying gain matrix for the IIPNLMS algorithm are given by:

$$\begin{aligned}
F[|\hat{w}_l(k)|, k] &= |\hat{w}_l(k)| \\
\gamma_{\min}(k) &= \rho \max\{\delta_p, F[|\hat{w}_1(k)|, k], \ldots, F[|\hat{w}_L(k)|, k]\} \\
\gamma_l(k) &= \max\{\gamma_{\min}(k), F[|\hat{w}_l(k)|, k]\} \\
\gamma'_l(k) &= \frac{1 - \alpha_{\text{IIPNLMS }l}(k)}{2} + \frac{1 + \alpha_{\text{IIPNLMS }l}(k)}{2}\gamma_l(k) \\
g_l(k) &= \frac{\gamma'_l(k)}{\frac{1}{L}\sum_{l=1}^{L}\gamma'_l(k)}.
\end{aligned}$$

The term $\alpha_{\text{IIPNLMS }l}(k)$ is unique to the IIPNLMS algorithm and is described as follows. First, the objective of the IIPNLMS algorithm was to derive a rule to locate the "active" portion of the echo path in order to further improve performance. In the IPNLMS, the parameter α_{IPNLMS} was fixed for the whole echo path coefficients. Second, in the IIPNLMS version, α_{IIPNLMS} is allowed to vary as:

$$\alpha_{\text{IIPNLMS }l}(k) = \begin{cases} \alpha_{1\text{ IIPNLMS}}, \text{ when } \gamma_l(k) > \gamma_{\text{IIPNLMS}} \max\limits_{l} \gamma_l(k) \\ \alpha_{2\text{ IIPNLMS}}, \text{ when } \gamma_l(k) < \gamma_{\text{IIPNLMS}} \max\limits_{l} \gamma_l(k) \end{cases}$$

where γ_{IIPNLMS} is a parameter used to control the threshold in order to locate the "active" portion.

1.4.6. *IAF-PNLMS algorithm*

The IAF-PNLMS algorithm was introduced in [DAS 10]. The IAF-PNLMS algorithm proceeds in the following manner:

$$\begin{aligned}
F[|\hat{w}_l(k)|, k] &= |\hat{w}_l(k)| \\
\psi_l(k) &= \begin{cases} \frac{1}{2}F[|\hat{w}_l(k)|, k] + \frac{1}{2}\gamma_l(k-1), & k = mL \\ & m = 1, 2, 3, \ldots \\ \psi_l(k-1), & \text{otherwise} \end{cases} \\
\gamma_l(k) &= \max\{\psi_l(k), F[|\hat{w}_l(k)|, k]\}.
\end{aligned}$$

Typically, $\psi_l(0)$ is initialized to some small positive constant for all of the coefficients.

In contrast to the other PNLMS-type algorithms, such as the PNLMS and IPNLMS, the IAF-PNLMS algorithm transfers part of the inactive coefficient gains via $\psi_l(k)$ to the active coefficient gains [DAS 10], and, as a consequence, has the following properties:

1) It provides better ("truly proportionate") gain distribution compared with the PNLMS and IPNLMS algorithms.

2) It slows down the convergence speed of the small coefficients.

Point (1) leads to an improvement in the convergence speed as well as in tracking ability, enabling the IAF-PNLMS algorithm to outperform both the PNLMS and IPNLMS algorithms for impulse responses with high sparseness. Moreover, the point (2) is undesirable, arising from the fact that the IAF-PNLMS algorithm transfers gains from the inactive coefficients to the active coefficients over the whole adaptation process.

1.4.7. *MPNLMS algorithm*

The control law for the MPNLMS algorithm assigns a gain proportional to the logarithm of the estimated coefficients [DEN 05, DEN 06] as follows:

$$F[|\hat{w}_l(k)|, k] = \ln(1 + \mu|\hat{w}_l(k)|),\ 1 \leq l \leq L, \qquad [1.5]$$

where $\mu = 1/\epsilon$. The parameter ϵ is used to define when a coefficient is considered to be converged. For instance, a coefficient could be considered to have converged if it is within the ϵ-vicinity of its true value.

1.4.8. *EPNLMS algorithm*

The EPNLMS algorithm uses switching [DEN 06]. The switching criterion for this algorithm is based on the magnitude of the estimated coefficients. If the coefficient magnitude is less than ϵ, the minimum possible gain is assigned. Otherwise the gain assigned to the coefficients is proportional to the natural logarithm of the magnitude of the coefficient as shown here:

$$F[|\hat{w}_l(k)|, k] = \begin{cases} 0 & \text{if } |\hat{w}_l(k)| < \epsilon \\ \ln\left(\frac{|\hat{w}_l(k)|}{\epsilon}\right) & \text{if } |\hat{w}_l(k)| \geq \epsilon. \end{cases}$$

This algorithm tries to limit the resources (update gain) applied to the coefficients that have reached the ϵ-vicinity of their assumed true values of zero (sparsity).

1.5. Summary

In this chapter, an introduction to PtNLMS algorithms was given. The chapter started by presenting applications that motivate the design and analysis of PtNLMS algorithms. Classic examples of applications for PtNLMS algorithms such as telephone echo cancellation were discussed as well as more recent applications such as VOIP and acoustic underwater communication. Then a unified framework for representing PtNLMS algorithms was presented, followed by several examples of PtNLMS algorithms. Key strengths and weaknesses of each PtNLMS algorithm were also discussed.