01 Jan 1998

An Efficient, Fast Converging Adaptive Filter for Network Echo Cancellation

Steven L. Grant
Missouri University of Science and Technology, sgrant@mst.edu

Recommended Citation

S. L. Grant, "An Efficient, Fast Converging Adaptive Filter for Network Echo Cancellation," *Conference Record of the 32nd Asilomar Conference on Signals, Systems & Computers, 1998*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1998.
The definitive version is available at https://doi.org/10.1109/ACSSC.1998.750893

An Efficient, Fast Converging Adaptive Filter for Network Echo Cancellation

Steven L. Gay
Acoustics and Speech Research Department
Bell Labs, Lucent
600 Mountain Avenue
Murray Hill, New Jersey, USA

Abstract

This paper discusses a fast efficient adaptive filtering algorithm for network echo cancellers PNLMS++ (Proportionate normalized least mean squares ++). Compared to the conventional normalized least mean squares (NLMS) algorithm, PNLMS++ converges much more quickly when the echo path is sparse. When the echo path is dispersive, the convergence rate is the same as NLMS. In addition, the new algorithm diverges at the same rate and to the same misalignment level as NLMS during periods of undetected double-talk. PNLMS++ is only 50% more computationally complex than NLMS and requires no additional memory.

1. Introduction

Recently, the PNLMS [1] (proportionate normalized least mean squares) adaptive filter was developed for use in network echo cancellers. In comparison to the classical, NLMS [2] (normalized least mean squares) algorithm, PNLMS has extremely fast initial convergence and tracking performance when the echo path is sparse. Fortunately, network echo paths usually are quite sparse.

Though network echo cancellers now have echo path lengths of about 64 ms, the "active" part of the echo path is usually only about 4 to 6 ms long. The additional length of the filter is used to cover the "flat delay" in the long-distance network between the echo canceller and the hybrid/local-loop circuit. The period of this flat delay is unknown from call to call. The true echo path coefficients corresponding to the flat delay are zero. After the flat delay comes a short, exponentially decaying response (the 4 to 6 ms part) which quickly dampens to a relatively insignificant level. PNLMS exploits this characteristic, by effectively windowing the data with weights roughly proportional to the magnitude of the estimated impulse response. Heuristically, this is a good idea, because as the adaptive coefficients converge to a sparse impulse response, the algorithm, in a sense, becomes a progressively shorter adaptive filter; and short adaptive filters converge and track faster than long ones.

An idea by Horna [3] somewhat similar to PNLMS, but apparently less stable, has appeared before. The emphasis there was on computational and memory efficiencies realized by using logarithmic representations of the adaptive coefficients and the excitation sequence, although, the faster-than-NLMS convergence was also noted.

It is important to note that unlike many algorithms, PNLMS's convergence speed is not gained at the price of increased sensitivity to near-end noise and/or double-talk detection errors. Computationally, PNLMS is about 50% more complex than NLMS, but requires no additional memory.

There are, however, some disadvantages to PNLMS. After fast initial convergence on the larger coefficients, the remaining small coefficients adapt at a rate slower than NLMS. Also, if the impulse response is dispersive or consists of several sparse impulse responses, the rate of convergence can actually be much slower than NLMS's. While these are not expected to be serious handicaps in the intended application, they may prove limiting in unforeseen circumstances or alternate applications. PNLMS++ addresses these deficiencies by using both the NLMS and PNLMS algorithms in the coefficient vector update. The result is an algorithm, which converges and tracks near the rate of the faster of the two algorithms, NLMS or PNLMS. PNLMS++ also retains the advantageously slow double-talk divergence rate of PNLMS. Fortunately, PNLMS++'s computational complexity and memory requirements are no greater than PNLMS's.

2. PNLMS

In this section we derive the PNLMS coefficient update from a cost function that favors sparse solutions, a view

0-7803-5148-7/98/$10.00©1998 IEEE

slightly different than taken in [1]. First, we define the signals, vectors and matrices used in the paper,

- x_n is the far-end signal which excites the echo path,
- $\mathbf{x}_n = \left[x_n, \ldots, x_{n-L+1}\right]^T$ is the excitation vector,
- $\mathbf{h}_{ep} = \left[h_0, \ldots, h_{L-1}\right]^T$ is the true echo path impulse response vector,
- v_n is the near-end signal, or near-end noise,
- $y_n = \mathbf{x}_n^T \mathbf{h}_{ep} + v_n$ is the combination of the echo and from the near-end signals,
- $\mathbf{h}_n = \left[h_{0,n}, \ldots, h_{L-1,n}\right]^T$ is the adaptive filter coefficient vector,
- $e_n = y_n - \mathbf{x}_n^T \mathbf{h}_{n-1}$ is the error or residual-echo signal,
- $\mathbf{G}_n = diag\left\{g_{0,n}, \ldots, g_{L-1,n}\right\}$ is the diagonal individual step-size matrix, and
- μ is the "stepsize" parameter and is chosen in the range, $0 < \mu < 1$.
- δ is the regularization parameter (this prevents division by zero when $\mathbf{x}_n^T \mathbf{x}_n$, is very small).

The general form of an adaptive filter's error calculation and coefficient update is

$$e_n = y_n - \mathbf{x}_n^T \mathbf{h}_{n-1} \tag{1}$$

$$\mathbf{h}_n = \mathbf{h}_{n-1} + \mu \mathbf{r}_n \tag{2}$$

where $\mathbf{r}_n$ is the coefficient update vector at sample period n. One way to determine $\mathbf{r}_n$ is to minimize a weighted version of its norm subject to the constraint that the a posteriori error,

$$\dot{e}_n = y_n - \mathbf{x}_n^T \mathbf{h}_n = e_n - \mu \mathbf{x}_n^T \mathbf{r}_n, \tag{3}$$

is zero when μ is one. Hence, we write the cost function,

$$C_n = \delta \mathbf{r}_n^T \mathbf{G}_n^{-1} \mathbf{r}_n + \left(e_n - \mathbf{x}_n^T \mathbf{r}_n\right)^2 \tag{4}$$

This is reminiscent of Douglas's approach in [4] except that there, instead of a quadratic function of $\mathbf{r}_n$ various norms of $\mathbf{r}_n$ are minimized.

For PNLMS, $\mathbf{G}_n$ is diagonal with diagonal elements roughly proportional to the magnitude of the estimated coefficients.

Consider $\mathbf{h}_{n-1}$ as a point in an L dimensional space with basis vectors corresponding to the columns of an L by L identity matrix. When $\mathbf{h}_{n-1}$ has only one large value, then the point that represents it in this space lies near one of the coordinate axes. According to the first term of (4) the cost of moving orthogonal to the axis is expensive (but not impossible) while moving parallel to it is relatively cheap. So, the cost function favors moving the coefficient estimate vector on trajectories roughly parallel to the basis vectors. This is equivalent to saying that it favors sparse coefficient estimate vectors.

Using the matrix inversion lemma it can be seen that (4) is minimized by,

$$\mathbf{r}_n = \mu \mathbf{G}_n \mathbf{x}_n (\mathbf{x}_n^T \mathbf{G}_n \mathbf{x}_n + \delta)^{-1} e_n \tag{5}$$

This is slightly different than the PNLMS update vector described in [1]. There, the denominator in (5) is $\mathbf{x}_n^T \mathbf{x}_n + \delta$ which can become unstable for impulsive excitation signals. Note that when $\mathbf{G}_n = \mathbf{I}$ (5) reduces to the standard regularized NLMS coefficient update.

The individual step-size matrix is calculated from the coefficient vector of the previous sample period according to the following steps,

$$L_{\max} = \max\left\{\delta_p, \left|h_{0,n-1}\right|, \ldots, \left|h_{L-1,n-1}\right|\right\} \tag{6}$$

$$\gamma_i = \max\left\{\rho L_{\max}, \left|h_{i,n-1}\right|\right\} \quad 0 \le i \le L-1 \tag{7}$$

$$L_1 = \sum_{i=0}^{L-1} \gamma_i \tag{8}$$

$$\left(\mathbf{G}_n\right)_{i,i} = \gamma_i / L_1 \qquad 0 \le i \le L-1 \tag{9}$$

where δ_p and ρ are small positive numbers. δ_p prevents the algorithm from misbehaving when the adaptive coefficients are all near zero (e.g. at initialization) and ρ prevents individual coefficients from *freezing* when their magnitude is much smaller than the largest coefficient, $L_{\max}$. Typically, $\delta_p \approx 0.01$ and $\rho \approx 5/L$. If ρ is set to one, then $\mathbf{G}_n = \mathbf{I}$ and PNLMS behaves like NLMS.

Figure 1 compares the trajectories of the adaptive filter coefficients for NLMS, PNLMS and PNLMS++ (described below) for sparse echo paths. For ease of presentation we have chosen L=2. The filter coefficients are initialized at $\mathbf{h}_0 = [.01 \quad 1.4]^T$ and the true coefficients are at $\mathbf{h}_{ep} = [1.4 \quad .01]^T$. Note that the PNLMS trajectory follows the coordinate axes, while the NLMS trajectory proceeds more or less directly from the initial point to the solution ($v_n \approx 0$ in this simulation). Even though it *seems* that NLMS is taking a shorter route, PNLMS is getting there faster. When the simulation is stopped after 40 sample periods, PNLMS is closer than NLMS to $\mathbf{h}_{ep}$.

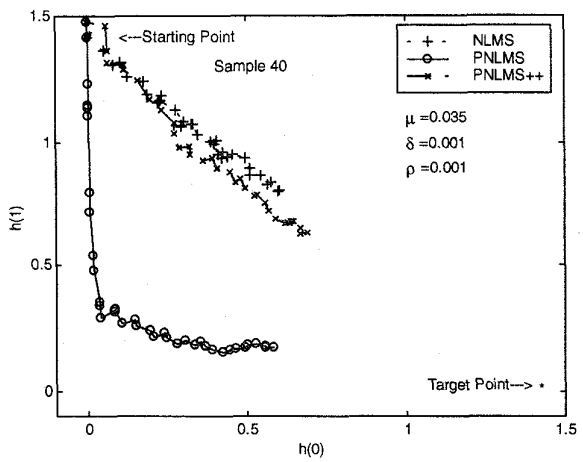


Figure 1: Trajectories of NLMS, PNLMS, and PNLMS++ for a "sparse" echo path.

Figure 2 shows the results of a simulation similar to that of Figure 1 except that here dispersive echo paths are considered. The filter coefficients are initialized at $\mathbf{h}_0 = [-1 \quad 1]^T$ and the true coefficients are at $\mathbf{h}_{ep} = [1 \quad 1]^T$. This time the trajectories of the three algorithms need to cross a coordinate axis. This is no problem for NLMS, but when the PNLMS coefficients get close to the axis, the PNLMS cost function causes the trajectory to favor horizontal rather than vertical movement. Hence, PNLMS bogs down. This time when the simulation is stopped after 40 sample periods it is NLMS that is closer to $\mathbf{h}_{ep}$.

The results of more realistic simulations are shown in Figures 3 and 4. Here the filter lengths are 512 coefficients. Figure 3a shows the coefficient error of PNLMS and NLMS adaptive filters when white Gaussian noise is used as the excitation and near-end signals. The echo path, shown in figure 3b, (a typical network echo path) is relatively sparse. Here, PNLMS converges much faster than NLMS. At 3.2 seconds the background noise power, σ_v^2, was raised from -39 dB below the echo to -10 dB below the echo to show the effect of undetected double-talk on the coefficient error. Note that even though PNLMS converges much faster than NLMS, they both diverge at about the same rate and to about the same value.

Figure 4a shows the coefficient error of PNLMS and NLMS when the echo path is dispersive, as shown in figure 4b. Here, PNLMS converges much more slowly than NLMS.

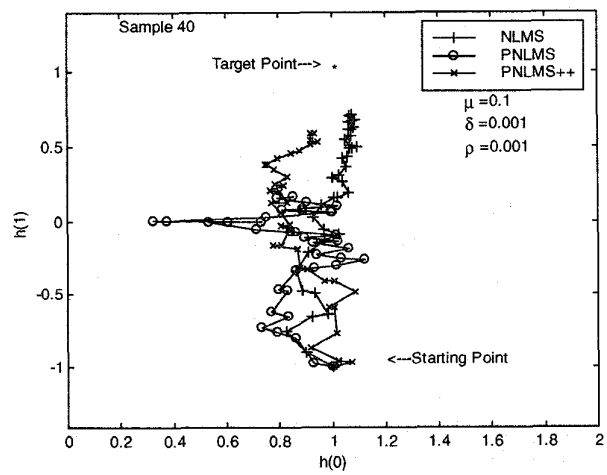


Figure 2: Trajectories of NLMS, PNLMS, and PNLMS++ for a "dispersive" echo path.

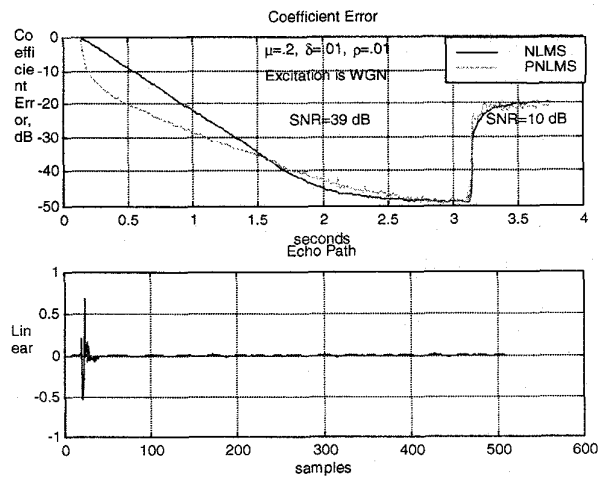


Figure 3: A) Coefficient error convergence for NLMS and PNLMS. B) A sparse echo path impulse response.

3. PNLMS++

PNLMS++ uses both NLMS and PNLMS type coefficient updates. Two implementation methods have been investigated. One, where the update algorithm alternates each sample period and the other where both types of updates are done in the same sample period. The first method is slightly less complex and so is preferred over the second. To distinguish these two methods we will refer to the alternating update algorithm as PNLMS++(AU) and the double update algorithm as PNLMS++(DU).

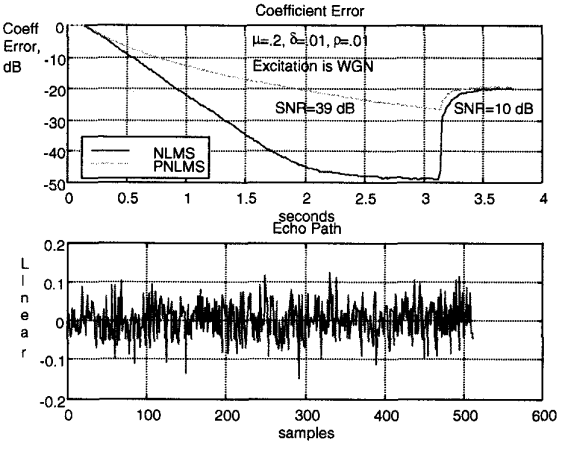
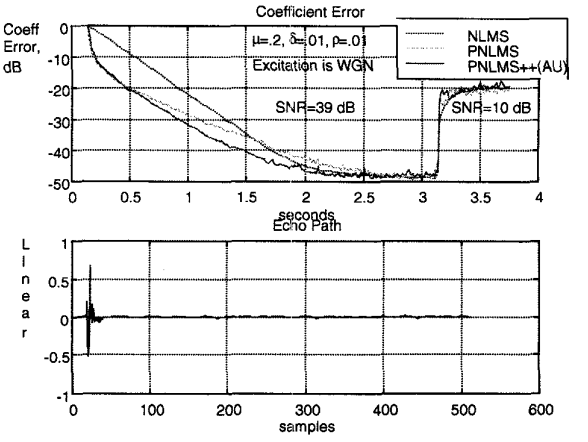


Figure 4: A) Coefficient error convergence for NLMS and PNLMS. B) Dispersive echo path impulse response.

In both implementations, in any given period of samples, whichever algorithm converges faster will contribute most to the updates. If PNLMS bogs down, as it does in figure 2, NLMS moves the coefficients forward. So, as shown in figure 2, PNLMS++ converges about as fast as NLMS. On the other hand, under sparse conditions, when PNLMS converges quickly, it mainly determines the convergence speed of PNLMS++. In figure 1, we see that PNLMS++ is about as close to the true echo path as PNLMS is when the simulation is terminated.

3.1 Alternating Updates:

First, we discuss the "alternating update" procedure, PNLMS++(AU), where each sample period the coefficient vector is updated using either the NLMS or PNLMS algorithm. This could be done any number of ways; for example, one could normally use the NLMS update, and then use PNLMS every k^{th} sample period, or vice-versa. Here we simply use NLMS updates on the odd sample periods and PNLMS on the even. The coefficient error



performance of PNLMS++(AU), PNLMS, and NLMS is shown in figure 5a for a sparse echo path (figure 5b) and for a dispersive echo path in figure 6. The convergence of PNLMS++(AU) is close to the faster of the two algorithms, PNLMS or NLMS.

Figure 5: A) Coefficient error convergence for NLMS, PNLMS, and PNLMS++(AU). B) Sparse echo path impulse response.

The average PNLMS++(AU) complexity is actually lower than PNLMS. However the maximum computational complexity (usually a more relevant benchmark for real-time applications) is the same. Obviously PNLMS++(AU) requires no more memory over PNLMS. The alternating update mechanism is easy to implement. It can be accomplished by always using the PNLMS algorithm and changing ρ back and forth from 1 to its normal PNLMS value.

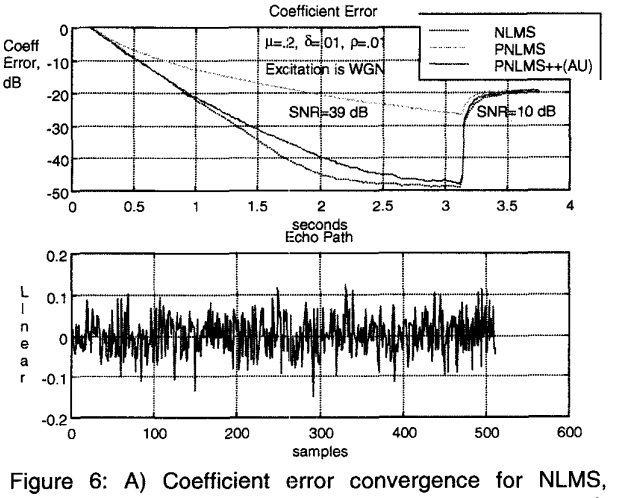


Figure 6: A) Coefficient error convergence for NLMS, PNLMS, and PNLMS++(AU). B) Dispersive echo path impulse response.

3.2 Double Updates

An alternate method of using both NLMS and PNLMS coefficient updates is to use both NLMS and PNLMS coefficient updates each sample period. The error calculation and coefficient updates can be written as,

$$e_n = y_n - \mathbf{x}_n^T \mathbf{h}_{n-1} \tag{10}$$

$$\mathbf{g} = \mathbf{h}_{n-1} + \mu \mathbf{G}_n \mathbf{x}_n (\mathbf{x}_n^T \mathbf{G}_n \mathbf{x}_n + \delta)^{-1} e_n \tag{11}$$

$$\varepsilon = y_n - \mathbf{x}_n^T \mathbf{g} \tag{12}$$

$$\mathbf{h}_n = \mathbf{g} + \mu \mathbf{x}_n (\mathbf{x}_n^T \mathbf{x}_n + \delta)^{-1} \varepsilon \tag{13}$$

Equations (10) and (11) represent the PNLMS part of the update ($\mathbf{G}_n$ is determined from equations (6) through (9)) and equations (12) and (13) represent the NLMS part.

Equations (10) through (13) can be simplified into a single error calculation and coefficient update,

$$e_n = y_n - \mathbf{x}_n^T \mathbf{h}_{n-1} \tag{14}$$

$$\mathbf{h}_n = \mathbf{h}_{n-1} + \mu \hat{\mathbf{G}}_n \mathbf{x}_n e_n \tag{15}$$

where

$$\left(\hat{\mathbf{G}}_n\right)_{i,i} = \frac{\left(\mathbf{G}_n\right)_{i,i}}{\mathbf{x}_n^T \mathbf{G}_n \mathbf{x}_n + \delta} + \frac{1-\mu}{\mathbf{x}_n^T \mathbf{x}_n + \delta} \quad 0 \le i \le L-1 \tag{16}$$

The performance of PNLMS++(DU) versus PNLMS and NLMS under the echo path conditions described above is virtually the same as PNLMS++(AU). The only differences are that the stepsize, μ, and the PNLMS parameter ρ need some minor adjustments in PNLMS++(DU) to yield the same final coefficient error.

Figure 7 shows that the PNLMS++(DU) performance is about the same as PNLMS when the echo path is very

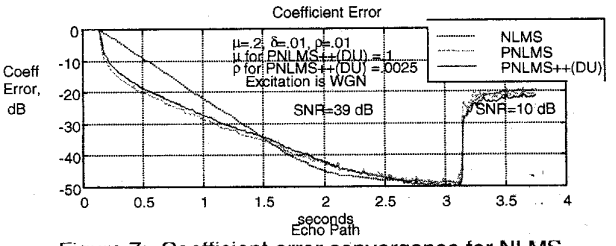


Figure 7: Coefficient error convergence for NLMS, PNLMS, and PNLMS++(DU) for a sparse echo path.

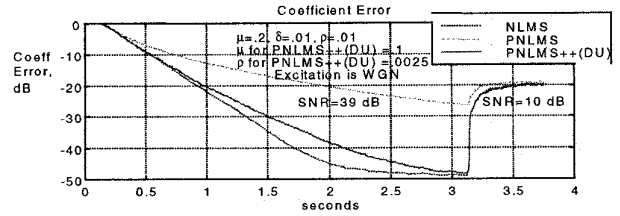


Figure 8: Coefficient error convergence for NLMS, PNLMS, and PNLMS++(DU) for a dispersive echo path.

sparse, while figure 8 shows that PNLMS++(DU) performs about as well as NLMS when the echo path is very dispersive. So, PNLMS++(DU) implicitly "selects" that mode of operation that yields the fastest convergence. This "implicit mode selection" takes place by virtue of equation (13). PNLMS++(DU) costs one more addition per tap per sample period over PNLMS. No extra memory is needed.

4. Conclusions

PNLMS++ provides improved convergence and tracking over PNLMS when the echo path impulse response is dispersive, thus, increasing the range of applications of the algorithm. PNLMS and PNLMS++ have about the same complexity.

References

[1] Duttweiler, D. L., "Proportionate Normalized Least Mean Squares Adaptation in Echo Cancelers", Submitted to Transactions on Speech and Audio Processing.

[2] Widrow, B. and Stearns, S. D., *Adaptive Signal Processing,* Englewood Cliffs, NJ, Prentice-Hall, Inc., 1985.

[3] Horna, O. A., "Echo Canceller Utilizing Pseudo logarithmic Coding," National Telecommunications Conference, 1977, pp. 04:7.1-04:7.

[4] Douglas, S.C., "A family of normalized LMS algorithm," IEEE Signal Processing Letters, Vol. 1, No. 3, March 1994.