6. Comparison of NLMS-OCF with Existing Algorithms

In Chapters 2 – 5 we derived the NLMS-OCF algorithm, analyzed the convergence and tracking behavior of NLMS-OCF, and developed a fast version of the NLMS-OCF algorithm. We also mentioned the existence of a class of algorithms "equivalent" to NLMS-OCF. The distinguishing characteristic of these algorithms is that they update the weights based on multiple input signal vectors. As mentioned in Chapter 2, Ozeki and Umeda's affine projection algorithm (APA) is a well-known example of this class [2]. Algorithms such as the partial rank algorithm (PRA) and generalized optimal block algorithm (GOBA) also belong to this class. An excellent review of this APA class of algorithms was published recently [3]. Several other algorithms, such as New Data-Reusing LMS (NDR-LMS) proposed by Schnaufer and Jenkins [27] and Decorrelating Algorithms (DA) proposed by Rupp [20], also use multiple input vectors for adaptation.

In this chapter, we compare the performance of NLMS-OCF with the APA, NDR-LMS, and DA algorithms. These algorithms use multiple input vectors for adaptation. However, they use the input vectors differently for adaptation and hence yield different performances.

The convergence rate of these algorithms is controlled by the step-size constant $\bar{\mu}$. We analyze the behavior of these algorithms for $\bar{\mu}=1$ due to two reasons. Firstly, $\bar{\mu}=1$ is the optimum step size to maximize the convergence rate. Secondly, the adaptation corresponding to $\bar{\mu}=1$ leads to a nice geometric interpretation of the above algorithms.

Different authors use different notations for variables such as step-size, filter order, and number of vectors used for adaptation. To avoid confusions due to notations, we use the standard notations used earlier in this dissertation while discussing the different algorithms.

6.1 NLMS Algorithm

The weight update equation of the NLMS algorithm is given by [1]

$$\hat{\mathbf{w}}_{n+1} = \hat{\mathbf{w}}_n + \frac{\bar{\mu}}{\left\|\mathbf{x}_n\right\|^2}\mathbf{x}_n e_n \tag{6.1}$$

where $\hat{\mathbf{w}}_n$ is the old estimate (*a priori*) for the plant parameters, $\mathbf{x}_n$ is the input vector given by $\mathbf{x}_n = \begin{pmatrix} x_n & x_{n-1} & \cdots & x_{n-N+1} \end{pmatrix}^T$, and e_n is the *a priori* estimation error given by

$$e_n = d_n - \hat{\mathbf{w}}_n^T \mathbf{x}_n \tag{6.2}$$

In the above equations, $(\bullet)^T$ denotes the transpose operator. For $\bar{\mu} \in (0,2)$, the NLMS algorithm updates the estimate for the weights $\hat{\mathbf{w}}_n$ so that the *a posteriori* estimation error given by

$$\bar{e}_n = d_n - \hat{\mathbf{w}}_{n+1}^T \mathbf{x}_n \tag{6.3}$$

is strictly less than the *a priori* estimation error. In particular, for $\bar{\mu} = 1$, the *a posteriori* error is reduced to zero. That is,

$$\hat{\mathbf{w}}_{n+1}^T \mathbf{x}_n = d_n . \tag{6.4}$$

Since (6.4) is usually an under-determined system of equations (one equation with N unknowns), there is a manifold π_n of solutions $\hat{\mathbf{w}}_{n+1}$ that reduce the *a posteriori* error to zero. The update generated by NLMS is the solution that minimizes the norm of the increment in weights $\left\| \hat{\mathbf{w}}_{n+1} - \hat{\mathbf{w}}_n \right\|$. Equivalently, the new estimate for weights $\hat{\mathbf{w}}_{n+1}$ generated by NLMS is the orthogonal projection of the current estimate $\hat{\mathbf{w}}_n$ onto the manifold π_n of solutions to (6.4).

The NLMS algorithm has a very low computational complexity of $\mathrm{O}(N)$.

6.2 Affine Projection Algorithm

While NLMS updates the weights based only on the current input vector $\mathbf{x}_n$, APA updates the weights based on an additional M such vectors in the most recent past, namely on $\mathbf{x}_n, \mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \ldots, \mathbf{x}_{n-M}$. For $\bar{\mu} = 1$, the AP algorithm updates the weights so that the a *posteriori errors* $\bar{e}_{k/n+1} = d_k - \hat{\mathbf{w}}_{n+1}^T \mathbf{x}_k$ are *simultaneously* reduced to zero for $k = n, n-1, \ldots, n-M$. Equivalently, the new weight estimate $\hat{\mathbf{w}}_{n+1}$ generated by APA is the orthogonal projection of the current estimate $\hat{\mathbf{w}}_n$ onto the intersection of the manifolds $\pi_n, \pi_{n-1}, \ldots, \pi_{n-M}$, i.e. APA orthogonally projects $\hat{\mathbf{w}}_n$ onto $\bigcap_{k=0}^{M} \pi_{n-k}$.

The weight adaptation equation of APA is given by [2]

$$\hat{\mathbf{w}}_{n+1} = \hat{\mathbf{w}}_n + \bar{\mu}\mathbf{X}_n\left[\mathbf{X}_n^T\mathbf{X}_n\right]^{-1}\mathbf{e}_n \tag{6.5}$$

where $\mathbf{X}_n = \left[\mathbf{x}_n \;\; \mathbf{x}_{n-1} \cdots \mathbf{x}_{n-M}\right]$ is the matrix formed by the current and past input vectors used for adaptation and $\mathbf{e}_n = \left[\bar{e}_{n/n} \;\; \bar{e}_{n-1/n} \cdots \bar{e}_{n-M/n}\right]^T$ is the vector formed by the associated *a priori* estimation errors. Notice that when M is chosen to be zero, the APA adaptation, shown in (6.5), reduces to NLMS adaptation, shown in (6.1).

Another interesting interpretation of APA has been put forth by Rupp [20]. The vector used to adapt the adaptive filter (equivalently, the direction along which the filter coefficients are updated) is usually referred to as the information vector. Rupp shows that for $\bar{\mu} = 1$ APA adaptation is equivalent to using the component of $\mathbf{x}_n$ that is orthogonal to $\mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \ldots, \mathbf{x}_{n-M}$ as the information vector, while NLMS uses $\mathbf{x}_n$ itself as the information vector. The weights are adapted so that the *a posteriori* error $\bar{e}_{n/n+1}$ becomes zero. Furthermore, since the adaptation is orthogonal to $\mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \ldots, \mathbf{x}_{n-M}$ the *a posteriori* errors $\bar{e}_{k/n+1}$ for $k = n-1, n-2, \ldots, n-M$ remain the same as their corresponding *a priori* errors, which are zero. Using the component of $\mathbf{x}_n$ that is orthogonal to the past *M* vectors tends to decorrelate the individual elements of the information vector, especially if $\mathbf{x}_n$ is an autoregressive (AR) process of order at most *M*. Hence, Rupp argues, APA can converge faster than NLMS. It would be worth noting here that this does not mean that the convergence rate of APA is the same as the convergence rate of NLMS with white (decorrelated) input, or that no further improvement in convergence rate can be realized by increasing *M* beyond the order of the AR process $\mathbf{x}_n$. This is because APA is not merely decorrelating the information vector; it is also insuring that the current information vector is orthogonal to the *M* input vectors in the most recent past. While the decorrelation interpretation helps to explain the acceleration of convergence corresponding to an AR input, it does not explain, for example, why APA converges faster for a white input signal that already generates decorrelated information vectors (so that APA can not decorrelate $\mathbf{x}_n$ any further!). The accelerated convergence in the latter case can, however, be explained by the orthogonality property. If the information vector is orthogonal to the past *M* vectors, then *M*+1 *a posteriori* errors are *simultaneously* forced to zero. The decorrelation property does not guarantee this; with

a decorrelated information vector, which is not necessarily orthogonal to the past *M* vectors, only the current *a posteriori* error is set to zero. Thus, the orthogonality property provides a heuristic explanation for the faster convergence of APA in the case of a white input signal. Our APA convergence analysis results, derived in Chapter 3, and the simulation results provided in Section 6.6 corroborate the fact that APA converges faster than NLMS for a white input.

The AP algorithm as shown in (6.5) has a computational complexity of $\mathrm{O}(NM^2)$. A fast version of the APA is also known [19]. This fast AP (FAP) algorithm has a reduced computational complexity of $2N + \mathrm{O}(M)$. However, the fast transversal filter used by FAP is known to have numerical problems [1].

6.3 NLMS with Orthogonal Correction Factors

The NLMS-OCF algorithm, as does APA, adapts the weights based on *M* vectors in the past, in addition to the current input vector. However, the past vectors are chosen differently. NLMS-OCF uses a set of vectors that are spaced *D* samples apart, starting from the current input vector. That is, the NLMS-OCF adaptation is based on $\mathbf{x}_n, \mathbf{x}_{n-D}, \mathbf{x}_{n-2D}, \ldots, \mathbf{x}_{n-MD}$, where *D* is the input vector delay. APA is the special case of NLMS-OCF corresponding to $D = 1$. Hence, NLMS-OCF is considered a generalized AP algorithm. For $\bar{\mu} = 1$, the NLMS-OCF algorithm updates the weights so that the a *posteriori errors* $\bar{e}_{k/n+1} = d_k - \hat{\mathbf{w}}_{n+1}^T \mathbf{x}_k$ are simultaneously reduced to zero for $k = n, n-D, \ldots, n-MD$. Equivalently, the new weight estimate $\hat{\mathbf{w}}_{n+1}$ generated by NLMS is the orthogonal projection of the current estimate $\hat{\mathbf{w}}_n$ onto the intersection of the manifolds $\pi_n, \pi_{n-D}, \ldots, \pi_{n-MD}$, i.e. NLMS-OCF orthogonally projects $\hat{\mathbf{w}}_n$ onto $\bigcap_{k=0}^{M} \pi_{n-kD}$.

As we showed in Chapter 2, for $D = 1$, the weight update generated by NLMS-OCF is identical to the weight update generated by APA. NLMS-OCF allows for using an input vector delay larger than unity. For large values of *D*, the input vectors $\mathbf{x}_n, \mathbf{x}_{n-D}, \mathbf{x}_{n-2D}, \ldots, \mathbf{x}_{n-MD}$ are more likely to be mutually uncorrelated and, hence, the input vectors used for weight updates tend to span a more "diverse" subspace. This helps to accelerate the convergence under most

circumstances. Consequently, the NLMS-OCF algorithm can provide faster convergence than the APA for the same number *M* of input vectors used for the weight update.

As we saw in earlier chapters, the NLMS-OCF based on Gram-Schmidt orthogonalization [17] for computing OCF has a computational complexity of $O(NM^2)$ while the fast NLMS-OCF that uses a lattice structure to generate the OCF has a computational complexity of $O(NM)$.

6.4 New Data-Reusing LMS Algorithm

The normalized data-reusing LMS (NDR-LMS) algorithm proposed by Schnaufer and Jenkins also uses the past input vectors for adaptation [27]. However, the past input vectors are used for adaptation differently. The NDR-LMS algorithm repeatedly reuses input vectors from previous iterations, in addition to the input vector from the current iteration, in order to generate gradient estimates and to adapt the coefficient vector along each of the gradients. In mathematical terms, NDR-LMS updates the weights as shown below [27]:

$$\mathbf{c}_0 = \hat{\mathbf{w}}_n \tag{6.6}$$

For $i = 0,\ldots,R\text{-}1$, repeat (6.7)

$$\mathbf{c}_{i+1} = \mathbf{c}_i + \mu_{0,i}\mathbf{x}_n + \mu_{1,i}\mathbf{x}_{n-1} + \ldots + \mu_{M,i}\mathbf{x}_{n-M} \tag{6.7}$$

$$\hat{\mathbf{w}}_{n+1} = \mathbf{c}_R \tag{6.8}$$

where $\mu_{k,i} = \left(d_{n-k} - \mathbf{c}_i^T\mathbf{x}_{n-k}\right) / \left\|\mathbf{x}_{n-k}\right\|^2$. The parameter *R* is the reuse factor. This idea of data reusing was originally motivated by the possibility to exploit the unused cycles available on the processor. Hence, *R* is usually chosen depending on the number of unused cycles available. As *R* approaches infinity, the new weight estimate generated by NDR-LMS approaches the new weight estimate generated by APA [56].

A careful comparison of the NDR-LMS update equation, shown in (6.6)-(6.8), with the NLMS-OCF adaptation equation reveals that NDR-LMS uses the current and past input vectors directly for its weight update, while NLMS-OCF uses an orthogonal basis of the space spanned by the current and past input vectors for its weight update. In geometric terms, NDR-LMS projects $\hat{\mathbf{w}}_n$ during each iteration successively onto $\pi_n, \pi_{n-1}, \ldots, \pi_{n-M}$ and this sequential

projection is repeated *R* times. This adaptation procedure does not ensure that the *a posteriori* errors at *n*, *n*-1,…, *n-M* are minimized simultaneously. Instead, the *a posteriori* errors are minimized sequentially. Because of the sequential minimization, while minimizing the *a posteriori* error at $n-k$, the previously minimized *a posteriori* errors are not maintained at zero. This causes the weight update produced by NDR-LMS to depend on the order in which the input vectors are used. The NDR-LMS algorithm has a complexity of $\mathrm{O}(NMR)$.

6.5 Decorrelating Algorithms

Two algorithms belonging to this family have been proposed by Rupp [20]. These algorithms were motivated by the decorrelating property of APA. Since the decorrelation interpretation of APA is valid only if the input signal is an AR process, APA is modified to preserve the decorrelation property if the input signal is a moving average (MA) or ARMA process, resulting in the corresponding decorrelating algorithms. However, the modified algorithms lose the more important orthogonality property possessed by APA. In other words, for $\bar{\mu}=1$ these algorithms set only the current *a posteriori* error to zero. Hence, these algorithms converge slower than APA.

The update equations of the decorrelating algorithm for MA input are given below [20].

$$\begin{aligned}
\mathbf{Z}_n &= \left[\boldsymbol{\phi}_{n-1}\ \boldsymbol{\phi}_{n-2}\ \cdots\ \boldsymbol{\phi}_{n-M}\right] \\
\mathbf{b}_n &= \left[\mathbf{Z}_n^T\mathbf{Z}_n\right]^{-1}\mathbf{Z}_n\mathbf{x}_n \\
\boldsymbol{\phi}_n &= \mathbf{x}_n - \mathbf{Z}_n\mathbf{b}_n \\
e_n &= d_n - \mathbf{w}_n^T\mathbf{x}_n \\
\hat{\mathbf{w}}_{n+1} &= \hat{\mathbf{w}}_n + \bar{\mu}\frac{\boldsymbol{\phi}_n}{\left\|\boldsymbol{\phi}_n\right\|^2}e_n
\end{aligned} \tag{6.9}$$

The first three steps in (6.9) essentially attempt to decorrelate the MA process x_n. This decorrelation process results in the "decorrelated" vector $\boldsymbol{\phi}_n$, which is orthogonal to $\boldsymbol{\phi}_{n-k}$ for $k=1,2,\ldots,M$. DA adapts the weights along $\boldsymbol{\phi}_n$ such that the *a posteriori* error at *n* is minimized (zero if the step size is unity). Since $\boldsymbol{\phi}_n$ being orthogonal to $\boldsymbol{\phi}_{n-k}$ for $k=1,2,\ldots,M$ does not necessarily mean that $\boldsymbol{\phi}_n$ is orthogonal to $\mathbf{x}_{n-k}$ for $k=1,2,\ldots,M$, the *a posteriori* errors at *n*-1,

n-2,…, *n*-*M* are not necessarily zero. Hence, the decorrelating algorithms do not possess the (desirable) orthogonality property. Consequently, one does not expect DA to outperform APA.

The decorrelating algorithm, shown in (6.9), has a low computational complexity of $2N + \mathrm{O}(M)$. Since the decorrelation process involves (time-varying) feedback, the decorrelation algorithm can possibly become unstable [20]. This is a major drawback of this class of algorithms.

6.6 Simulation Results

The performance of the above algorithms was compared using MATLAB simulations for four different input processes, namely being white, first-order AR with its pole at 0.95, highly colored first-order MA with its zero at 0.95, and reasonably colored first-order MA with its zero at 0.5. The system to be identified in each case is modeled using a 31st order FIR filter. The tap weights of the system are chosen such that the maximum entropy assumption is satisfied. The adaptive filter and the underlying system are assumed to have the same number of taps. We further assume that the measurement noise is absent. The number *M* of additional input vectors (compared to NLMS) is chosen to be 10 for each algorithm. The step size $\bar{\mu}$ is set to unity. We use delay *D* of 32 for NLMS-OCF simulations and reuse factor *R* of unity for NDR-LMS simulations. All simulations use soft initialization. All the adaptive filter coefficients are initialized to zero. All the results shown are obtained by ensemble averaging 25 independent realizations.

Figure 6.1 shows the learning curves of the algorithms corresponding to white input. We observe that APA converges faster than NLMS and that NLMS-OCF has the fastest convergence among all of the algorithms. The fact that APA converges faster than NLMS for white input confirms that APA does not possess merely the decorrelating property (but also the orthogonality property). Interestingly, for white input, NDR-LMS and DA have the same convergence rate as APA. While NLMS converges at a rate of approximately 20 dB for every $5N$ iterations, NLMS-OCF converges at a rate of approximately 20 dB for every $5N/(M+1)$ iterations.

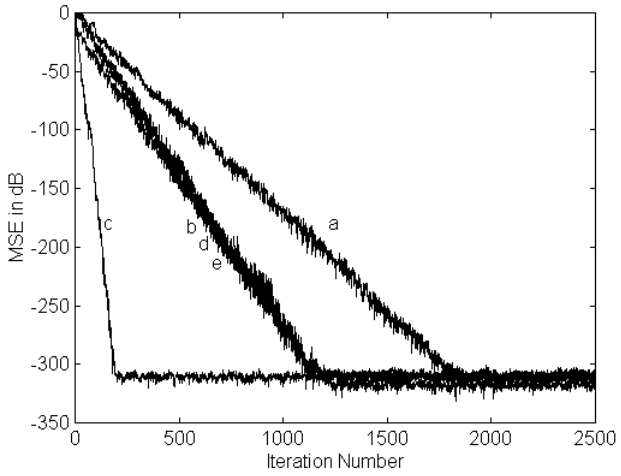

Figure 6.1 Learning Curves for White Input:
(a) NLMS, (b) APA, (c) NLMS-OCF, (d) NDR-LMS, and (e) DA.

Figure 6.2 shows the learning curves for the AR input. It is evident that NLMS-OCF has the fastest convergence and that NDR-LMS has the slowest convergence (but faster than NLMS) among the algorithms being compared.

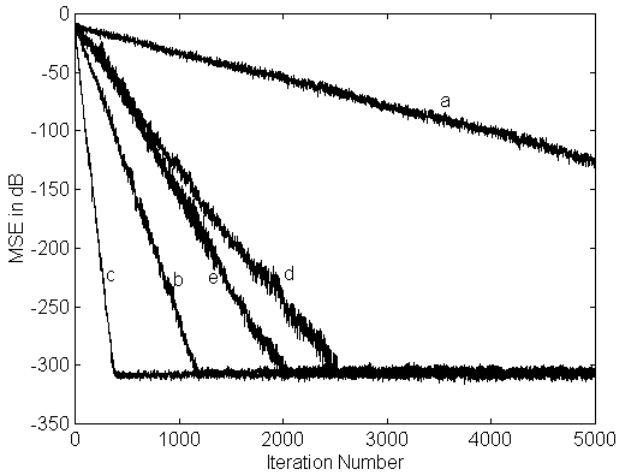

Figure 6.2 Learning Curves for AR Input:
(a) NLMS, (b) APA, (c) NLMS-OCF, (d) NDR-LMS, and (e) DA.

Figure 6.3 shows the learning curves for the highly colored MA input. We observe that APA, which is not an "optimal" decorrelation algorithm for the MA input, has a slightly faster convergence than DA, which "optimally" decorrelates the MA input. NDR-LMS has the slowest convergence (but faster than NLMS) for the MA input as well. We also see from Figure 6.3 that

NLMS-OCF is slower than APA. This suggests that increasing *D* does not help to accelerate the convergence if the input is a highly colored MA process.

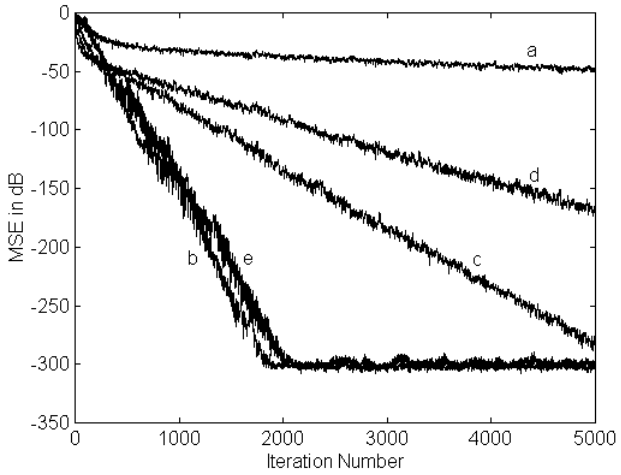

Figure 6.3 Learning Curves for Highly Colored MA Input:
(a) NLMS, (b) APA, (c) NLMS-OCF, (d) NDR-LMS, and (e) DA.

The learning curves corresponding to the reasonably colored MA input are shown in Figure 6.4. We observe that NLMS-OCF has the fastest convergence and also that APA, DA, and NDR-LMS have comparable convergence rates.

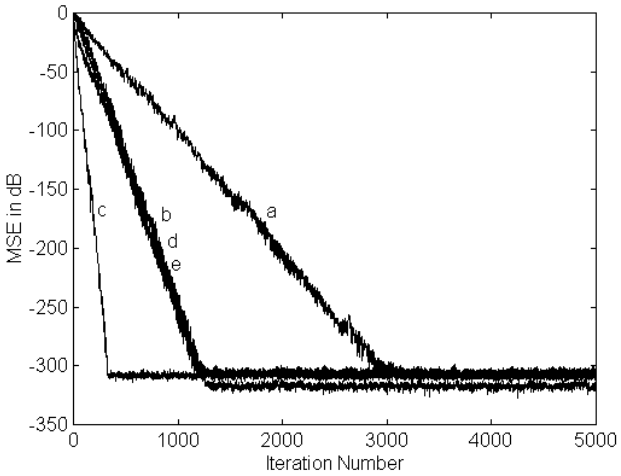

Figure 6.4 Learning Curves for Reasonably Colored MA Input:
(a) NLMS, (b) APA, (c) NLMS-OCF, (d) NDR-LMS, and (e) DA.

6.7 Conclusion

We compared NLMS-OCF with different adaptive filtering algorithms. While all of them perform better than NLMS, NLMS-OCF performs better than the rest for white, AR, and reasonably colored MA inputs. While APA and DA yield comparable performances for the white and MA inputs, APA outperforms DA for the AR input. APA converges faster than NLMS even for the white input, suggesting that APA possesses not only the "decorrelation property" but also the orthogonality property. Even though APA is not the "optimal" decorrelating algorithm for MA inputs, APA performs as well as DA for the MA input. However, DA, which *is* the optimal decorrelating algorithm for MA inputs, does not perform as well as APA for AR inputs.