A STOCHASTIC ANALYSIS OF THE AFFINE PROJECTION ALGORITHM FOR GAUSSIAN AUTOREGRESSIVE INPUTS

N. J. Bershad[1] , D. Linebarger[2] and S. McLaughlin[3]
November 2000

[1] Department of Electrical and Computer Engineering , University of California, Irvine, CA 92697, (949)-824-6709, fax-(949)-824-3203, bershad@ece.uci.edu.
[2] Department of Electrical Engineering, University of Texas-Dallas, Richardson, Texas, 75083-0688, (972)-883-4950, fax-(972)-883-2710, dl@utdallas.edu.
[3] Department of Electrical Engineering, University of Edinburgh, Edinburgh, Scotland, U.K., (01 31)-650-5578, fax-(0131)-650-6554, S.Mclaughlin@ee.ed.ac.uk.

ABSTRACT

This paper studies the statistical behavior of the Affine Projection (AP) algorithm for $\mu = 1$ for Gaussian Autoregressive inputs. This work extends the theoretical results of Rupp [3] to the numerical evaluation of the MSE learning curves for the adaptive AP weights. The MSE learning behavior of the AP(P+1) algorithm with an AR(Q) input ($Q \leq P$) is shown to be the same as the NLMS algorithm ($\mu = 1$) with a white input with M-P unity eigenvalues and P zero eigenvalues and increased observation noise. Monte Carlo simulations are presented which support the theoretical results.

I. INTRODUCTION

The AP algorithm was originally proposed by Ozeki and Omeda in 1984 [1] and re-discovered by Slock [2] as the USWC RLS algorithm. Subsequently, Rupp [3] presented an interpretation of the AP algorithm which high-lighted two ideas: 1) decorrelation (or pre-whitening properties) and 2) an increase in the background noise floor due to the decorrelation operation. Most recently, Sankaran and Beex [4] analyzed the convergence behavior of the AP algorithm using a signal model due to Slock [5]. The first four papers promote the idea that the increased speed of the algorithm is due to its decorrelation properties. The under-determined Least Squares portion of the algorithm estimates the AR parameters of the input signal and essentially pre-whitens the signal for input to an NLMS type structure. Although the papers agree on the qualitative behavior of the algorithm, only [4] provides a quantitative analysis of the algorithm. The reason is the difficulty of actually calculating many of the expectations in the AP stochastic recursion. This difficulty is primarily due to the numerical complexity of the under-determined LS solution, which is embedded in the algorithm. Reference [4] is able to circumvent these analytic difficulties by careful choice of the signal model.

This paper presents some specific transient numerical results for the AP algorithm for AR Gaussian signal models by extending Rupp [3] and using the NLMS results presented in [6].

II. THEORETICAL BACKGROUND

II.A Input Signal Model

Let z(k) be a white gaussian sequence with unit variance. Let u(k) be an autoregressive process of order P,

$$u(k) = \sum_{i=1}^{P} a_i u(k-i) + z(k) \tag{1}$$

Assuming P is known a priori, samples of u(k) can be written as an (M x 1) column vector $\mathbf{u}(k)$,

$$\mathbf{u}^T(k) = [\, u(k),\ u(k-1),...u(k-M+1)\,] \tag{2}$$

The AR(P) process can then be written as

$$\mathbf{u}(k) = \sum_{i=1}^{P} a_i \mathbf{u}(k-i) + \mathbf{z}(k) = \mathbf{U}(k)\mathbf{a} + \mathbf{z}(k) \tag{3}$$

where $\mathbf{U}(k)$ is a collection of P of the past vectors,

$$\mathbf{U}(k) = [\mathbf{u}(k-1), \mathbf{u}(k-2),\ \ldots\ \mathbf{u}(k-P)] \tag{4}$$

and $\mathbf{z}(k)$ is an M-vector of samples from the white gaussian random sequence.

$$\mathbf{z}^T(k) = [\, z(k),\ z(k-1),\ \ldots z(k-M+1)] \tag{5}$$

The Least Squares estimate of the parameters of $\mathbf{a}$ is

$$\hat{\mathbf{a}}(k) = \left[\mathbf{U}^T(k)\mathbf{U}(k)\right]^{-1}\mathbf{U}^T(k)\mathbf{u}(k) \tag{6}$$

where $\mathbf{U}^T(k)\mathbf{U}(k)$ is assumed of rank P.

II.B Affine Projection Algorithm

The AP recursive algorithm [1] (for $\mu = 1$) is defined by the following set of operations: (4), (6) and

$$\begin{aligned}\phi(k) &= \mathbf{u}(k) - \mathbf{U}(k)\hat{\mathbf{a}}(k) \\ &= \left[I - \mathbf{U}(k)\left[\mathbf{U}^T(k)\mathbf{U}(k)\right]^{-1}\mathbf{U}^T(k) \right]\mathbf{u}(k)\end{aligned} \tag{7}$$

$$e(k) = d(k) - \mathbf{u}^T(k)\hat{\mathbf{w}}(k) \tag{8}$$

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + \frac{\phi(k)}{\phi^T(k)\phi(k)}e(k) \tag{9}$$

where $\hat{\mathbf{w}}(k)$ is the weight vector of the adaptive filter. The desired signal d(k) consists of a plant output $\mathbf{W}^T\mathbf{u}(k)$ plus an additive observation noise v(k), i.e. $d(k) = \mathbf{W}^T\mathbf{u}(k) + v(k)$ (see Figure 1). Rupp [3] shows that:

1) if u(k) is an AR process of order P, $\hat{\mathbf{a}}(k)$ is an estimate of its AR coefficients and $\phi(k) = \hat{\mathbf{z}}(k)$ (i.e. $\phi(k)$ is a vector whose elements are estimates of a white gaussian random sequence), and

2) With $\varepsilon(k) = \mathbf{W} - \hat{\mathbf{w}}(k)$, the AP recursion (9) can be written as

$$\varepsilon(k+1) = \varepsilon(k) - \frac{\phi(k)\phi^T(k)}{\phi^T(k)\phi(k)}\varepsilon(k) - \frac{\phi(k)}{\phi^T(k)\phi(k)}v_a(k) \quad (10)$$

with the filtered noise sequence

$$v_a(k) = v(k) - \sum_{i=1}^{P} \hat{a}_i(k)v(k-i) \quad (11)$$

and $\hat{a}_i(k)$ are the elements of $\hat{\mathbf{a}}(k)$. Rupp concludes that (for μ =1) 1) the AP algorithm essentially works like the NLMS algorithm but with the white input $\phi(k)$, and 2) with a filtered version $v_a(k)$ of the input noise v(k). For a white noise v(k), the background noise power is increased by approximately $(1+\mathbf{a}^T\mathbf{a})$ if the estimates $\hat{\mathbf{a}}(k)$ are close to **a**. Rupp closes this section of the paper with the comment that the recursion for the weight error vector variances (in a transformed coordinate system) for the NLMS algorithm [4, 5] is of the form

$$\mathbf{c}(k+1) = \mathbf{B}\mathbf{c}(k) + \mathbf{h} \quad (12)$$

Eq. (12) can be solved for the AP algorithm to yield a performance prediction for the weight error vector variances. Rupp [3] did not evaluate **B** and **h**. The goal of this paper is to compute **B**, **h**, and generate MSE learning curve predictions.

This paper presents these computations for eq. (12) for Gaussian AR processes. The results predicted by (12) are supported by Monte Carlo simulations ;

III. ANALYSIS OF THE AP ALGORITHM (μ = 1)

III.A Statistical Properties of $\hat{\mathbf{a}}(k)$

Inserting (3) in (6) yields

$$\hat{\mathbf{a}}(k) = \left[\mathbf{U}^T(k)\mathbf{U}(k)\right]^{-1}\mathbf{U}^T(k)\left[\mathbf{U}(k)\mathbf{a} + \mathbf{z}(k)\right] = \mathbf{a} + \left[\mathbf{U}^T(k)\mathbf{U}(k)\right]^{-1}\mathbf{U}^T(k)\mathbf{z}(k) \quad (13)$$

Now, it is clear from (3) and (4) that some components of **U**(k) are algebraicly dependent upon some components of **z**(k). However, it will be assumed here that this dependence can be neglected statistically. This assumption is similar to the independence assumption used to analyze many stochastic algorithms [7]. Averaging (13), using the independence assumption, yields

$$\mathbf{E}\left[\hat{\mathbf{a}}(k)\right] = \mathbf{a} \quad (14)$$

Hence, the estimate is un-biased.

IIIB. Statistical Properties of $v_a(k)$

Squaring and averaging (11) and assuming v(k) is a stationary white sequence yields

$$\begin{aligned} \mathrm{E}\left[v_a^2\right] &= \left\{1 + \sum_{i=1}^{P}\mathrm{E}\left[\hat{a}_i^2(k)\right]\right\}\mathrm{E}\left[v^2(k)\right] \\ &= \left[1 + \mathbf{a}^T\mathbf{a} + \sum_{i=1}^{P}\mathrm{var}\left[\hat{a}_i(k)\right]\right]\mathrm{E}\left[v^2(k)\right] \\ &= \left[1 + \mathbf{a}^T\mathbf{a} + \mathrm{Tr}\left\{\mathrm{E}\left[\left[\mathbf{U}^T(k)\mathbf{U}(k)\right]^{-1}\right]\right\}\right]\mathrm{E}\left[v^2(k)\right] \end{aligned} \quad (15)$$

III.C Statistical Properties of $\phi(k)$

Inserting (3) into (6) and the result into (7) yields

$$\phi(k) = \left\{\mathrm{I} - \mathbf{U}(k)\left[\mathbf{U}^T(k)\mathbf{U}(k)\right]^{-1}\mathbf{U}^T(k)\right\}\mathbf{z}(k) \quad (16)$$

Thus $\phi(k)$ is a white vector **z**(k) pre-multiplied by a matrix whose components are algebraicly dependent on components of **z**(k).

Rupp [3 - eq.(11a)] has shown that $\mathbf{U}^T(k)\phi(k) = \mathbf{0}$. Hence $\phi(k)$ is orthogonal to **u**(k-1), .. ,**u**(k-P). This, in turn, implies that **u**(k-j), $1 \le j \le P$, is an eigenvector of $\mathbf{U}(k)\left[\mathbf{U}^T(k)\mathbf{U}(k)\right]^{-1}\mathbf{U}^T(k)$ with unity eigenvalue for all $1 \le j \le P$. Therefore, expanding **z**(k) in a complete ortho -normal series using the eigenvectors of the matrix inside the brackets in (2) plus additional orthogonal vectors to form a complete set,

$$z(k) = \sum_{i=1}^{P} b_i(k)\mathbf{u}(k-i) + \sum_{i=P+1}^{M} b_i(k)\psi_i(k) \quad (17)$$

where the $\psi_i(k)$ are an orthonormal set with M-P members, also orthonormal to the **u**(k-i). The **u**(k-i) do not have to be mutually orthogonal because they have the same unity eigenvalue. For i > P, the $b_i(k)$ are zero mean, uncorrelated with unit variance. Assuming $\mathbf{U}^T(k)\mathbf{U}(k)$ is of rank P, then (17) is an equality in the mean-square sense. Using (17) in (16),

$$\phi(k) = \sum_{i=P+1}^{M} b_i(k)\varphi_i(k) \quad (18)$$

Thus, $\phi(k)$ has dimensionality M-P with covariance matrix

$$\mathrm{E}\left[\phi(k)\phi(k)^T\right] = \sum_{i=P+1}^{M}\psi_i(k)\psi_i(k)^T = \begin{bmatrix} \mathrm{I}_{M-P} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (19)$$

Thus, $\phi(k)$ is white vector with M-P unity eigenvalues and P zero eigenvalues.

III.D Computation of B and h

The weight error vector covariance learning behavior of the AP(P+1) algorithm with an AR(P) input is the same as the NLMS algorithm (μ = 1) with a white input with M-P unity

eigenvalues and P zero eigenvalues and observation noise $v_a(k)$ as given in (11). For this case, $\mathbf{B} = b\, I_{M-P}$, $\mathbf{h} = h\, \mathbf{1}$ ($\mathbf{1}$ is an M-P column vector of ones), and all components of $\mathbf{c}(k)$ are identical (b and h are scalars). Thus, (12) can be written as two scalar equations [4] for the components of $\mathbf{c}(k)$

$$c_j(k) = b^k c_j(0) \quad \text{for } 1 \le j \le P$$
$$c_j(k+1) = b c_j(k) + h \quad \text{for } P < j \le M \qquad (20)$$

Using [6-eqs. (23) and (25)],

$$b = \left[1 - \frac{\mu(2-\mu)}{M-P}\right], \qquad h = \frac{\mu^2 E[v_a^2]}{(M-P)(M-P-2)} \qquad (21)$$

By comparison, the corresponding expressions for the NLMS algorithm with a white input and arbitrary μ are

$$b_{NLMS} = \left[1 - \frac{\mu(2-\mu)}{M}\right], \quad h_{NLMS} = \frac{\mu^2}{(M-2)} \qquad (22)$$

Hence, the AP algorithm decorrelates the colored input process and has a faster convergence speed (i.e. $b < b_{NLMS}$) but a higher noise floor ($h > h_{NLMS}$) as compared to the NLMS algorithm with white inputs.

III.E Learning Curves for the AP Algorithm for μ =1

Reference [6] presents a general expression for eq. (12) for the NLMS algorithm using the independence assumption for successive data vectors for arbitrary stationary gaussian data vectors. The expression involves certain integrals, which depend upon the eigenvalues of the input data covariance matrix. When the input data vector is white (i.e. $E[\mathbf{u}(n)\mathbf{u}^T(n)]$ is proportional to the identity matrix), the integrals are easily evaluated. Reference [5] presents a general expression for eq. (12) for the NLMS algorithm using the independence assumption for spherically invariant processes. These expressions are easily evaluated for any eigen-value distribution and are shown in exact agreement with [6] for the white case. Assuming that (15) is time-invariant and using (21) in (20) with the initial conditions for the misadjustment, $c_j(0) = E[v_a^2]\, \mathbf{W}_j^2$

$$c_j(k) = \left[1 - \frac{1}{M-P}\right]^k E[v_a^2]\, \mathbf{W}_j^2 + \frac{E[v_a^2]}{(M-P-2)(M-P)} \sum_{q=0}^{k-1} \left[1 - \frac{1}{M-P}\right]^{k-q-1}$$
$$= \left[1 - \frac{1}{M-P}\right]^k E[v_a^2]\, \mathbf{W}_j^2 + \frac{E[v_a^2]}{(M-P-2)} \left[1 - \left(1 - \frac{1}{M-P}\right)^k\right] \qquad (23)$$

for $P < j \le M$.

Finally, using [6-eq. (23)], the MSE can be written as

$$MSE(k) = E[v_a^2] + \sum_{j=1}^{M} c_j^2(k) \qquad (24)$$

In steady-state,

$$\lim_{k\to\infty} MSE(k) = E\left[v_a^2\right]\left[1 + \frac{(M-P)}{M-P-2}\right] \qquad (25)$$

A similar result can be calculated for NLMS.

IV. MONTE CARLO SIMULATION RESULTS

Figures 2-5 compare the misadjustment for the theory and MC simulations (100 trials) for the AP algorithm and a tapped delay line simulation for NLMS for various sets of parameters as shown in the figure labels. The theoretical curves use (23) and (25) for $E\left[v_a^2\right] = 1 + a^2 = 1.81$ (i.e. neglecting the fluctuations in (15)). Figure 2 shows excellent agreement between the theory and MC simulations for AP(2) for an AR(1) input for both the transient and asymptotes. Figure 3 shows very good agreement between the theory and MC simulations for AP(4) for an AR(1) input for both the transient and asymptotes. Figure 4 shows fair agreement between the theory and MC simulations for AP(16) for an AR(1) input. Figures 2-4 show qualitative agreement with the theory in that increases in the order of the AP algorithm increases convergence speed but also increases the steady-state misadjustment. Figure 5 shows theory and MC simulations for the AP(2) algorithm for a 256 tap adaptive filter. The asymptotes are in good agreement but the transients are only in fair agreement. Many other comparisons of the theory and simulations have been made but cannot be shown here for reasons of space. However, in general, the theory has been shown quite accurate over a wide range of possible parameter variations.

V. CONCLUSIONS

This paper has presented an analytical model for predicting the behavior of the Affine Projection algorithm for its fastest convergence (step-size μ = 1). The model is based upon:
a) the AP error vector being a scalar for μ = 1 and
b) showing that the update term in the AP recursion is white.
Then, the variance recursions for the AP algorithm are similar to those of the NLMS algorithm. Thus, previously derived theory for the NLMS algorithm for white inputs can be used. A wide variety of Monte Carlo simulations were shown in good-to-excellent agreement with the behavior predicted by the theory.

REFERENCES

1. K. Ozeki and T. Umeda, " An Adaptive Filtering Algorithm using Orthogonal Projection to an Affine Subspace and its Properties," Electronics and Communications in Japan, Vol. 67-A, no. 5, pp. 19-27, 1984.

2. D. T. Slock, "Under-determined Growing and Sliding Covariance Fast Transversal Filter RLS Algorithms," Proc. Eusipco, Brussels, pp. 1169-1172, 1992
3. M. Rupp, "A Family of Adaptive Filter Algorithms with Decorrelating Properties," IEEE Trans. on Signal Processing, Vol. 46, No. 3, pp. 771-775, March 1998.
4. S. G. Sankaran and A. A. Beex, "Convergence Behavior of the Affine Projection Algorithm," IEEE Trans. on Signal Processing, Vol. 48, No. 4, pp. 1086-1097, April 2000.
5. D.T. Slock, "On the Convergence Behavior of the LMS and the Normalized LMS Algorithms," IEEE Trans. on Signal Processing, Vol 41, No. 9, pp. 2811-2825, Sept. 1993.
6. N. J. Bershad, "Analysis of the Normalized LMS Algorithm with Gaussian Inputs," IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-34, No. 4, pp. 793-806, August 1986.
7. S. Haykin, *Adaptive Filter Theory*, 3rd. ed., Englewood Cliffs, NJ: Prentice-Hall, 1996.

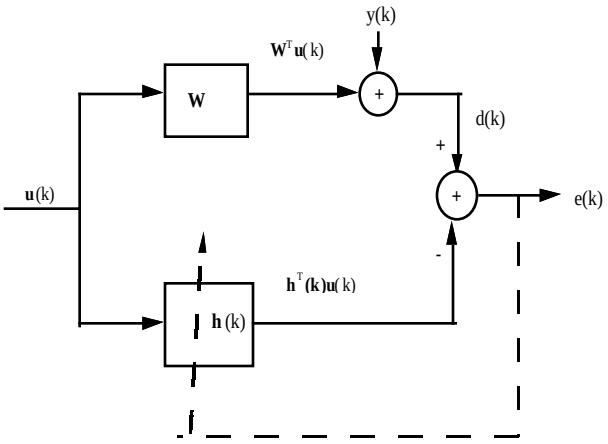

Figure 1 - Adaptive System Identification

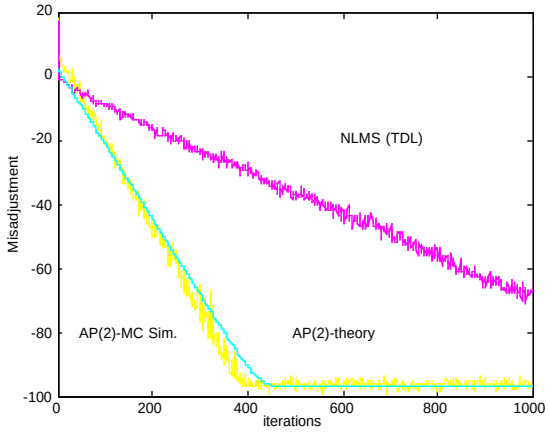

Figure 2 - Misadjustment for AP(2) and NLMS for AR(1) input (a = [.9], M= 20, μ = 1, SNR = 100 dB, AP(2) asymptotes: theory = 2.0229 x 10^(-10), simulations =3.5733 x 10^(-10).

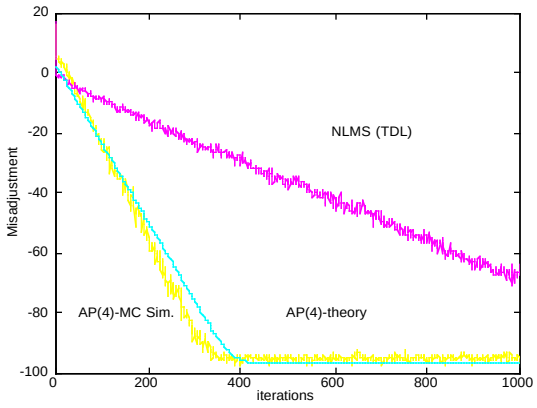

Figure 3 - Misadjustment for AP(4) and NLMS for AR(1) input (a = [.9 0 0 0], M= 20, μ = 1, SNR = 100 dB, AP(4) asymptotes: theory = 2.0513 x 10^(-10), simulations = 4.0998 x 10^(-10).

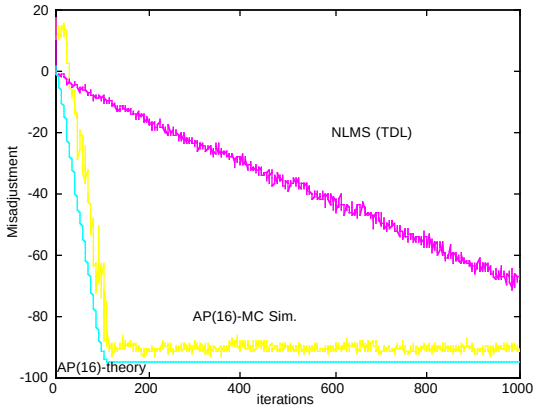

Figure 4- Misadjustment for AP(16) and NLMS for AR(1) input (a = [.9 ones(1,15)]), M= 20, μ = 1, SNR = 100 dB, AP(16) asymptotes: theory = 3.0167 x 10^(-10), simulations = 1.1143 x 10^(-9).

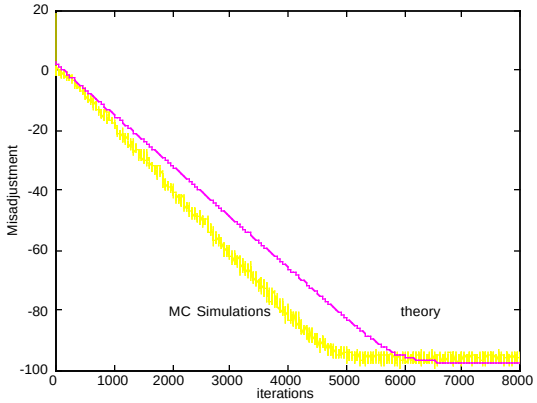

Figure 5 - Misadjustment for AP(2) for AR(1) input (a =.9), M= 256, μ = 1, SNR = 100 dB, asymptotes: theory = 1.8247 x 10^(-10), simulations = 3.6086 x 10^(-10). , M= 256, μ = 1, SNR = 100 dB, asymptotes: theory = 1.8247 x 10^(-10), simulations = 3.6768 x 10^(-10).