# SPHERICAL SUBSPACE AND EIGEN BASED AFFINE PROJECTION ALGORITHMS[3]

*Ronald D. DeGroat*[1] *Dinko Begusic*[2] *Eric M. Dowling*[1] *Darel A. Linebarger*[1]

[1] The University of Texas at Dallas
Electrical Engineering, EC 33
Richardson, TX 75083-0688 USA
972-883-2894 degroat@utdallas.edu

[2] University of Split
FESB Split
R. Boskovica b.b.
HR-21000 Split, Croatia
+385(21) 305-777 begusic@fesb.hr

## ABSTRACT

In this paper, we combine spherical subspace (SS) and eigen based updating methods with the affine projection (AP) method to produce a new family of fast SS-AP algorithms that offers additional tradeoffs between computation and adaptive filtering performance. Moreover, the implementation of SS-AP is less complicated than the fast RLS based AP algorithms. For certain applications, e.g., echo cancellation and equalization in digital subscriber loop (DSL) transceivers, SS-AP offers performance that is comparable to AP, but at computational costs that are less than the fast AP algorithms.

## 1. INTRODUCTION

The normalized least mean square (NLMS) algorithm is popular in adaptive filtering applications because of its simplicity in terms of computation and implementation. However, convergence may be slow and tracking poor. Fast recursive least squares (RLS) algorithms have been developed, but computational costs, implementational complexity, stability issues and excessively fast convergence/tracking speeds can sometimes be a problem. Fast [7][3] and medium fast [4] implementations of the affine projection (AP) method provide a range of compromise solutions that fall between NLMS and rectangularly windowed RLS.

If $L$ is the adaptive filter order and $p$ is the projection order ($1 \leq p \leq L$), then choosing $p = 1$ yields NLMS whereas $p = L$ produces rectangularly windowed RLS. The projection order determines how many equations are used at each update to update the filter coefficients. As the projection order is increased, so is the convergence speed as well as the computational complexity. The computational cost of the medium fast AP algorithm is $2L+5p^2+9p$. Fast implementations of the AP method based on using sliding window fast RLS are on the order of $2L + 20p$. The medium fast AP algorithm based on using the matrix inversion lemma is attractive because of its implementational simplicity, and for small values of $p$ ($p \leq \sqrt{L}$), it becomes computationally attractive compared to the fast RLS algorithms. However, compared to the fast AP algorithms, the medium fast AP algorithm must have $p < 4$ to be computationally superior.

[3] This work was supported in part by the Texas Advanced Research Program Grants 009741-022 and 009741-003, and the UTD Telecommunications DSP Consortium.

In this paper, we introduce a modification of the medium fast AP algorithm which offers additional tradeoffs in computation and performance. The modification involves the use of a simplified, eigen based approximation for the $p \times p$ correlation matrix. By tracking only a portion of the correlation matrix eigenstructure [1][5], computational costs can be reduced with a possible cost of some reduction in adaptive performance. However, there are many practical scenarios where the performance reductions (or differences) are minimal. Also, a large number of efficient eigen tracking methods have been developed in the last few years [2][6]. One way to track some of the eigencomponents and update a simplified version of the correlation matrix involves sphericalizing certain subsets of eigenvalues (e.g., replacing them with their average value). This can be described as spherical subspace (SS) updating. By introducing a multiplicity of eigenvalues, deflation can be used to reduce the size of the eigen update and the computation associated with it. If $r$ eigencomponents ($r < p$) are tracked in a $p \times p$ correlation matrix, i.e., $p - r$ eigenvalues are sphericalized , the computational costs of the medium fast AP algorithm can be transformed to $2L+pr^2+2pr+7p+O(r^3)$. In this case, we have used one of the most expensive, but numerically stable eigen tracking methods. However if $r = 1$, we obtain a computational cost of only $2L + 10p$. Using different, simplified updating schemes for the correlation matrix can produce a whole family of different AP-like algorithms. In this paper, we will focus on SS eigen updating methods and we will refer to this class of algorithms as SS-AP algorithms. The algorithms developed in this paper work best with white noise inputs which are encountered in such telecommunications applications as echo cancellation and equalization for digital subscriber loop (DSL) interfaces. In the simulation section, we compare SS-AP with regular AP using a DSL echo canceller.

## 2. AFFINE PROJECTION METHOD

The basic concept of the affine projection (AP) method (as described in [4]) will be summarized in this section . The output of the adaptive filter is given by

$$y(k) = x_L^T(k)h(k) \tag{1}$$

where $h(k)$ is an $L \times 1$ vector containing the filter coefficients at time $k$ ($L$ is the filter order), $x_L^T(k) = [x(k), x(k-1), ..., x(k-L+1)]$ is an $L \times 1$ vector containing the past $L$ input values, and $y(k)$ is the filter output at time $k$. The

objective of the adaptive filter is to generate an output that matches a desired response, $d(k)$, as well as possible. This is accomplished by adjusting the filter coefficient vector at every sample time by

$$h(k+1) = h(k) + \mu \Delta h(k) \tag{2}$$

where $\Delta h(k)$ is the adjustment vector and $\mu$ is the step size. Specifically, this is accomplished by solving $p$ equations ($p$ is called the projection order) at each sample time, i.e.,

$$d_p(k) = X_p^T(k) h(k+1) \tag{3}$$

or equivalently,

$$e_p(k) = X_p^T(k) \Delta h(k) \tag{4}$$

where

$$\begin{aligned} d_p(k) &= [d(k), ..., d(k-p+1)]^T & (5)\\ X_p(k) &= [x_L(k)|...|x_L(k-p+1)]_{L\times p} & (6)\\ e_p(k) &= d_p(k) - X_p^T(k) h(k). & (7) \end{aligned}$$

Note: $X_p(k)$ can also be expressed as $X_p^T(k) = [x_p(k)|...|x_p(k-L+1)]$.

For the underdetermined case ($p < L$), the minimum norm solution is given by

$$\Delta h(k) = X_p(k) \left( X_p^T(k) X_p(k) \right)^{-1} e_p(k). \tag{8}$$

We will find it convenient to rewrite this solution as

$$\Delta h(k) = X_p(k) g_p(k) \tag{9}$$

where

$$g_p(k) = R_p^{-1}(k) e_p(k) \tag{10}$$

and

$$R_p(k) = X_p^T(k) X_p(k). \tag{11}$$

The computational cost of the basic AP method is $(p+1)L + O(p^3)$. However, fast [7][3] and medium fast [4] implementations can be computed in $2L + 20p$ and $2L + 5p^2 + 9p$, respectively. Note: For $p = 1$, the AP method becomes the NLMS method.

## 3. SPHERICAL SUBSPACE UPDATING OF THE CORRELATION MATRIX

The basic idea of spherical subspace updating is to simplify the rank one eigen update of the correlation matrix. The simplification is accomplished by introducing a multiplicity of eigenvalues which allows the size of the eigenproblem to be deflated. For example, if the correlation matrix is $p \times p$ and the smallest $r$ eigenvalues are replaced by a single eigenlevel (usually an average), then the $p \times p$ eigenproblem can be deflated to an $(r+1) \times (r+1)$ eigenproblem. We will refer to a multiplicity of eigenvalues at a given level as an eigenlevel, and the subspace associated with it as a spherical subspace. Sphericalizing certain subspaces can lead to significant reductions in the computation of the eigenproblem. In [5], we showed that sphericalized subspace updating converges in the mean with probability one under certain reasonable assumptions. In fact, [5] shows that eigenlevels and spherical subspaces are multidimensional generalizations of eigenvalues and eigenvectors.

There are many different ways that the correlation matrix could be sphericalized [1][5], but we will concentrate on two major cases: 1) sphericalize all but the largest $r$ eigenvalues, or 2) sphericalize all but the smallest $r$ eigenvalues. We will denote the first case as a signal eigenstructure correlation matrix of order $r$ (SE($r$)) and the second as a noise eigenstructure correlation matrix of order $r$ (NE($r$)). Unless otherwise indicated, we will normally replace the sphericalized eigenvalues with their average level (although we could replace them with some other value, e.g., zero). Another way that we could sphericalize the eigenvalues of the correlation matrix would be to replace both the dominant $r$ eigenvalues with their average and also the subdominant $p-r$ eigenvalues with their average. We will call this two eigenlevel correlation matrix signal averaged of order $r$ (SA($r$)). The three examples of sphericalization just described are defined more precisely in the following equations.

Given the $p \times p$ correlation matrix,

$$R = \sum_{i=1}^{p} \lambda_i u_i u_i^T \tag{12}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$, an SE($r$) version is given by

$$\begin{aligned} R_{SE} &= \sum_{i=1}^{r+1} d_i U_i U_i^T, & (13)\\ d_i &= \lambda_i, \quad U_i = u_i, \quad i = 1 \cdots r, & (14)\\ d_{r+1} &= ave(\lambda_{r+1}, \cdots, \lambda_p),\ U_{r+1} = [u_{r+1}| \cdots |u_p] & (15) \end{aligned}$$

Similarly, an NE($r$) version is given by

$$\begin{aligned} R_{NE} &= \sum_{i=1}^{r+1} d_i U_i U_i^T, & (16)\\ d_1 &= ave(\lambda_1, \cdots, \lambda_{p-r}),\ U_1 = [u_1| \cdots |u_{p-r}], & (17)\\ d_{i+1} &= \lambda_{i+p-r}, \quad U_{i+1} = u_{i+p-r}, \quad i = 1 \cdots r. & (18) \end{aligned}$$

An SA(r) version of the correlation matrix is given by

$$\begin{aligned} R_{SA} &= \sum_{i=1}^{2} d_i U_i U_i^T, & (19)\\ d_1 &= ave(\lambda_1, \cdots, \lambda_r), \quad U_1 = [u_1| \cdots |u_r], & (20)\\ d_2 &= ave(\lambda_{r+1}, \cdots, \lambda_p),\ U_2 = [u_{r+1}| \cdots |u_p]. & (21) \end{aligned}$$

The computational cost of updating SE($r$) and NE($r$) is $O(pr^2)$ whereas SA($r$) is $O(pr_{min})$ where $r_{min} = min(r, p-r)$. Note: Since any SS decomposition can be divided into two complementary orthogonal subspaces, the largest spherical subspace does not need to be explicitly tracked. For example, in an SA update, $U_1 U_1^T = I - U_2 U_2^T$. However, if we can set $r = 1$, SE, NE and SA all have the exact same computational costs, which are quite low: approximately $5p$. In fact, SE(1) is identical to SA(1) and NE(1) is identical to SA($p-1$). They all track two distinct eigenlevels. For more details on SE, SA and and other other variations on SS updating, see [1] [5]. For other $O(pr)$ and $O(pr^2)$ eigen based tracking methods with various tracking abilities and numerical stabilities, see [2]).

## 4. SPHERICAL SUBSPACE BASED AFFINE PROJECTION (SS-AP) METHOD

Combining SS based correlation matrix updating with the AP method is straightforward. We simply replace the correlation matrix update with an SS update and use the simpler structure of the SS decomposition to simplify other computations in the algorithm, namely (10). In addition, we use an exponentially faded window on the SS correlation matrix with a fading factor that is comparable to the length $L$ rectangular window: $\alpha = 1 - 1/L$. The data matrix (6), however, is rectangularly windowed. Thus, SS-AP involves a hybrid kind of windowing: rectangular on the data matrix (6), but exponentially faded on the correlation matrix (11). This accomplishes two goals: 1) we avoid the numerically dangerous downdate associated with a rectangular window, and 2) we reduce the computation by replacing an update and downdate with a single, exponentially faded update. Our simulations indicate that as long as we choose $\alpha = 1 - 1/L$, no significant degradation in performance results. By applying an exponentially faded SS update to the $p \times p$ correlation matrix of the medium fast AP algorithm [4], we can reduce the computation from $2L + 5p^2 + 9p$ to $2L + pr^2 + 2pr + 7p$ if SE($r$) or NE($r$) are used. Alternatively, the computation can be reduced to $2L + 5pr + 5p$ if SA($r$) or certain other $O(pr)$ eigen tracking methods are used. The various eigen/subspace tracking approaches have different effects on the adaptive performance of the AP method, and the choice of $r$ gives us one more degree of control over the tradeoff between computational cost and adaptive performance of the SS-AP family of algorithms.

## 5. SIMULATIONS

In the following simulation, we compare the performance of the AP method with SE(1)-AP. Recall that the SE($r$) update tracks the $r$ dominant (primarily signal) eigencomponents of the correlation matrix and sphericalizes the remaining components. With SE(1), we track only the dominant eigencomponent and sphericalize the rest.

In this simulation, we use echo cancellation with 2500 random input samples line coded as 2B1Q (two binary bits are coded as one quaternary, i.e., four level signal). An echo impulse response is used as shown in fig. 1. In figs. 2 and 3, the tap weight error and the residual MSE are compared for SE-AP (dotted line) and AP (solid line). In both figures, the step size is varied from $\mu = 0.01$, 0.025, 0.05, 0.1, 0.2, 0.3, to 0.4. The plots that converge more slowly correspond to the smaller step sizes. Plots (a), (b) and (c) correspond to projection orders of $p = 1$, 8 and 12. Note: the $p = 1$ case corresponds to NLMS.

In all simulations, white Gaussian noise is added to the echo with an echo to noise $SNR = 30dB$. The filter order is $L = 128$ and the number of eigencomponents tracked is $r = 1$. The SS fading factor is set to $\alpha = 1 - 1/L$. The SS correlation matrix is initialized to $R_p(0) = 10^{-6} diag(p, ..., 2, 1)$. The non-SS correlation and the data matrices for both cases are all initialized to zero. This explains why some of the error measures might increase until the data and correlation matrices are full loaded with $L$ data samples. To prevent the possibility of singularities in $R_p^{-1}(k)$, a small constant diagonal offset ($D = 10^{-6}M$) is added to $R_p(k)$, i.e., $(R_p(k) + 10^{-6}I)^{-1}$ is used in place of $R_p^{-1}(k)$.

Clearly, over a wide range of step sizes and projection orders, SE-AP is comparable to AP. Notice that the only significant differences occur when the step size and/or the projection order becomes too large. Also, we see that setting $r = 1$ does not affect performance much. Of course, this is because we used white input data. If the data is not very white, SE(1)-AP may not work as well. Clearly, forcing most of the eigenvalues to be spherical when they are supposed to be spherical does not cause much degradation in performance. Also, increasing $r$ does not seem to make much difference in performance until $r \approx p$. Note: For $r > 1$, using NE($r$) instead of SE($r$) does not work quite as well because according to [5], the least dominant eigencomponents (tracked by NE($r$)) converge more slowly than the most dominant components (tracked by SE($r$)). The computational costs of SE(1)-AP are just $2L + 10p$ versus $2L + 20p$ for fast implementations of the AP method. Also, it is easier to implement SE(1)-AP than fast AP methods (which use fast transversal RLS algorithms).

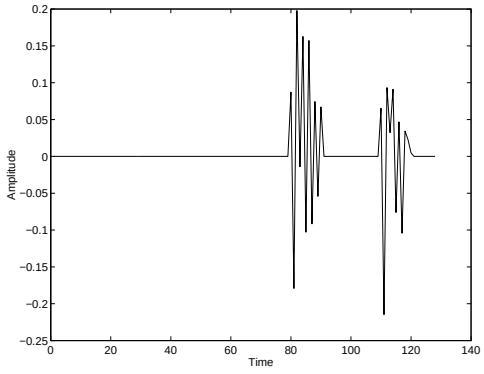



**Figure 1. Echo Impulse Response.**

## REFERENCES


[1] R. DeGroat. Non-iterative subspace tracking. *IEEE Trans. Sig. Proc.*, SP-40(3):571–577, Mar. 1992.

[2] R. DeGroat, E. Dowling, and D. Linebarger. Subspace tracking. In V. Madisetti and D. Williams, editors, *to appear in Signal Processing Handbook*. CRC Press.

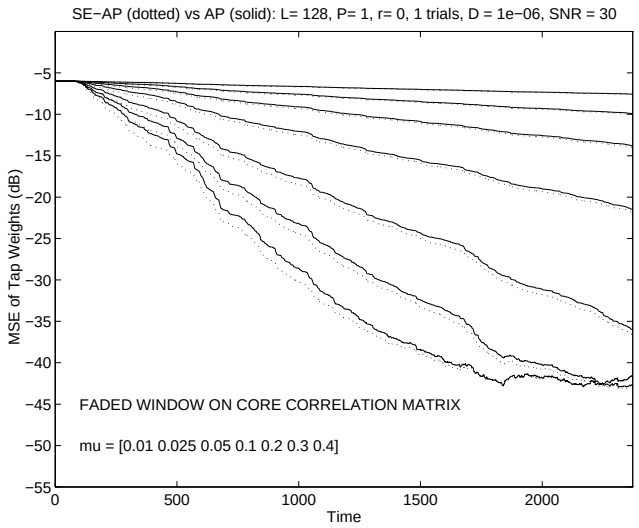
[3] S. Gay and S. Tavathia. The fast affine projection algorithm. In *ICASSP 95*, pages 3023–3026, 1995.

[4] Y. Kaneda, M. Tanaka, and J. Kojima. An adaptive algorithm with fast convergence for multi-input sound control. In *ACTIVE 95*, pages 993–1004, Newport Beach, CA, 1995.

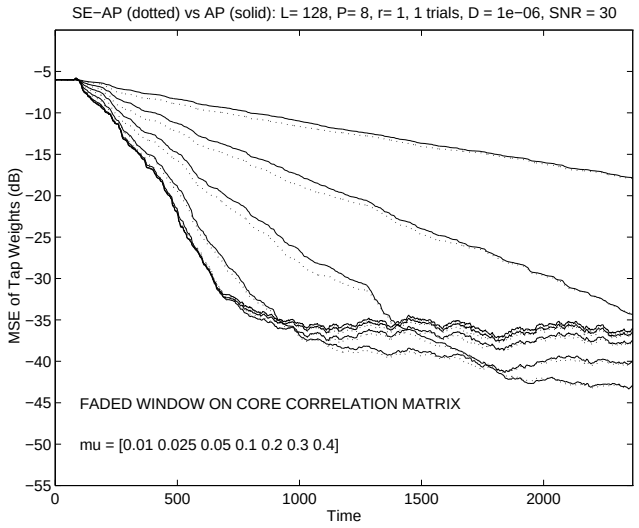
[5] R.DeGroat, E.Dowling, H.Ye, and D.Linebarger. Spherical subspace tracking for efficient, high performance adaptive signal processing applications. *Sig. Proc.*, 50:101–121, April 1996.

[6] V. U. Reddy, G. Mathew, and A. Paulraj. Some algorithms for eigensubspace estimation. *Digital Signal Processing*, 5:97–115, 1995.

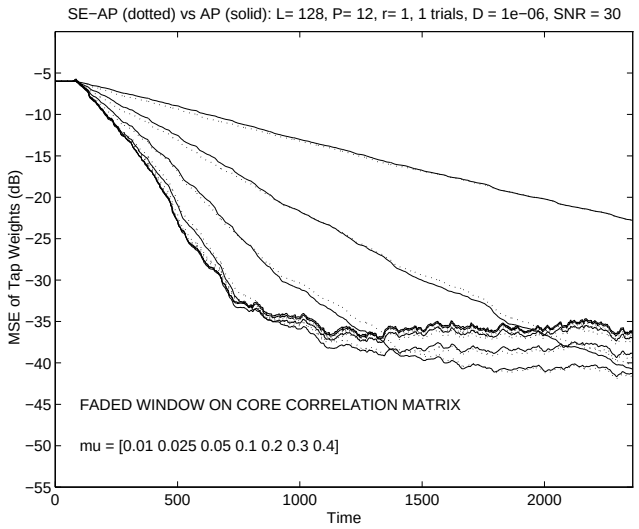
[7] M. Tanaka, Y. Kaneda, S. Makino, and J. Kojima. Fast projection algorithm and its step size control. In *ICASSP 95*, pages 945–948, 1995.




(a) p=1

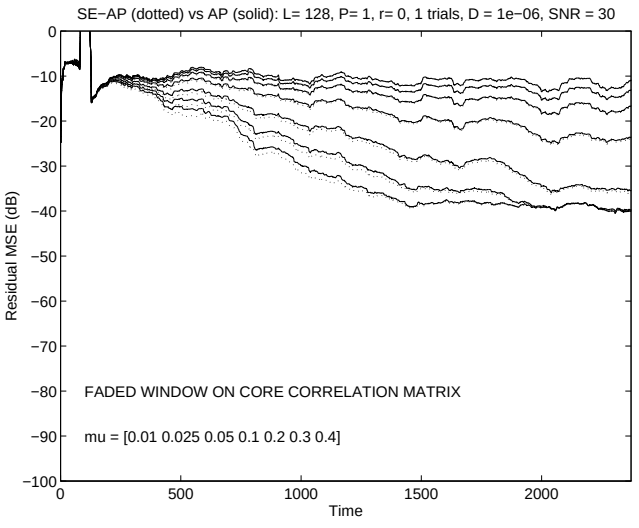



(b) p=8

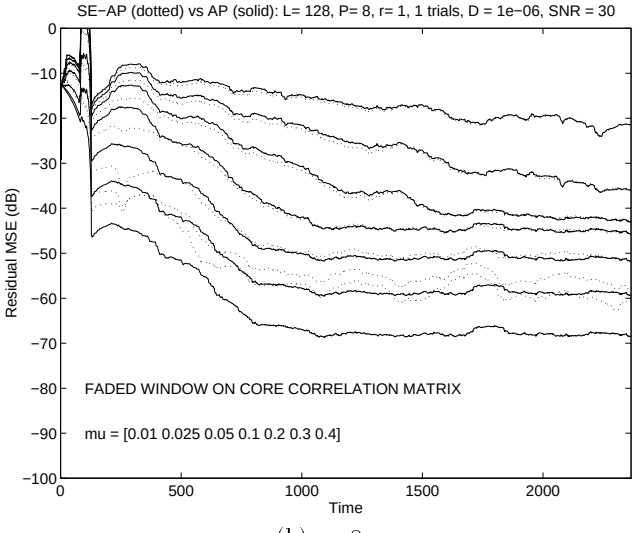



(c) p=12

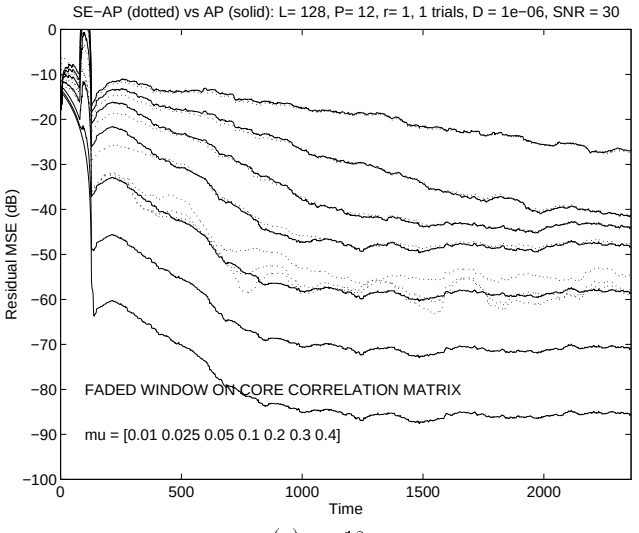
**Figure 2. MSE of Tap Weights: SS-AP (dotted) versus AP (solid) Method.**




(a) p=1




(b) p=8




(c) p=12

**Figure 3. Residual MSE: SS-AP (dotted) versus AP (solid) Method.**