THE FAST AFFINE PROJECTION ALGORITHM

Steven L. Gay and Sanjeev Tavathia

Acoustics Research Department
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, New Jersey 07974

ABSTRACT

This paper discusses a new adaptive filtering algorithm called fast affine projections (FAP). FAP's key features include LMS like complexity and memory requirements (low), and RLS like convergence (fast) for the important case where the excitation signal is speech. Another of FAP's important features is that it causes no delay in the input or output signals. In addition, the algorithm is easily regularized resulting in robust performance even for highly colored excitation signals. The combination of these features make FAP an excellent candidate for the adaptive filter in the acoustic echo cancellation problem. A simple, low complexity numerical stablization method for the algorithm is also introduced.

1. INTRODUCTION

The affine projection algorithm (APA)[1]. is a generalization of the well known normalized least mean square (NLMS) adaptive filtering algorithm[2]. Under this interpretation, each tap weight vector update of NLMS is viewed as a one dimensional affine projection. In APA the projections are made in multiple dimensions. As the projection dimension increases, so does the convergence speed of the tap weight vector, and unfortunately, the algorithm's computational complexity. Using techniques similar to those which led to fast (i.e., computationally efficient) recursive least squares (FLRS)[3] from recursive least squares (RLS)[4], a fast version of APA, fast (again, computationally efficient) affine projections (FAP)[5] [6] [7] may be derived*. As with RLS and FRLS, FAP requires the solution to a system of equations involving the implicit inverse of the excitation signal's covariance matrix (although with FAP the dimension of the covariance matrix is the dimension of the projection, N, not the length of the joint process estimation, L). FAP uses a sliding windowed FRLS[8] to assist in a recursive calculate of the solution. Since sliding windowed FRLS algorithms easily incorporate regularization of the covariance matrix inverse, FAP is regularized as well. The complexity of FAP is roughly 2L+20N multiplications per sample period. For applications like acoustic echo cancellation, L is usually much larger than the required N making FAP's complexity comparable to NLMS's (2L multiplications per sample period). Moreover, FAP does not require significantly greater memory than NLMS.

The adaptive filters discussed in this paper will be presented within the context of the echo cancellation problem.

* An independent work covering a substantial portion of the fast technique shown here has been published in Japanese[7] One difference is that the Japanese publication did not consider regularization in the fast algorithm which is important for convergence in the presence of noise.

2. THE AFFINE PROJECTION ALGORITHM

The affine projection algorithm, in a relaxed and regularized form, is defined as follows:

$$\underline{e}_n = \underline{s}_n - \mathbf{X}_n^t \underline{h}_{n-1} \tag{1}$$

$$\underline{\varepsilon}_n = \left[\mathbf{X}_n^t \mathbf{X}_n + \delta \mathbf{I}\right]^{-1} \underline{e}_n \tag{2}$$

$$\underline{h}_n = \underline{h}_{n-1} + \mu \mathbf{X}_n \underline{\varepsilon}_n. \tag{3}$$

The excitation signal matrix, $\mathbf{X}_n$, is L by N and has the structure,

$$\mathbf{X}_n = \left[\underline{x}_n,\ \underline{x}_{n-1},\ \cdots\ \underline{x}_{n-(N-1)}\right] \tag{4}$$

where the $\underline{x}_n = [x_n, \cdots x_{n-L+1}]^t$. The adaptive tap weight vector is $\underline{h}_n = [h_{0,n}, \ldots, h_{L-1,n}]^t$, where $h_{i,n}$ is the i^{th} tap at sample period n. The vector, $\underline{e}_n$, is of length N and consists of background noise and residual echo left uncancelled by the echo canceller's L-length adaptive tap weight vector, $\underline{h}_n$. The N-length vector, $\underline{s}_n$, is the system output consisting of the response of the echo path impulse response, $\underline{h}_{ep}$ to the excitation and the additive system noise, $\underline{y}_n$,

$$\underline{s}_n = \mathbf{X}_n^t \underline{h}_{ep} + \underline{y}_n. \tag{5}$$

The scalar δ is the regularization parameter for the sample autocorrelation matrix inverse used in (2) in the calculation of the N-length normalized residual echo vector, $\underline{\varepsilon}_n$. Where $\mathbf{X}_n^t \mathbf{X}_n$ may have eigenvalues close to zero, creating problems for the inverse, $\mathbf{X}_n^t \mathbf{X}_n + \delta \mathbf{I}$ has δ as its smallest eigenvalue which, if large enough, yields a well behaved inverse. The step-size parameter, μ is the relaxation factor. As in NLMS, the algorithm is stable for $0 \le \mu < 2$.

If N is set to one, relations (1), (2), and (3) reduce to the familiar NLMS algorithm. Thus, APA is a generalization of NLMS.

3. THE FAST AFFINE PROJECTION ALGORITHM

The complexity of APA is $2LN + K_{inv}N^2$ multiplies per sample period, where K_{inv} is a constant associated with the complexity of the inverse required in (2). If a generalized Levinson algorithm is used to solve the systems of equations in (2), K_{inv} is about 7. One way to somewhat mitigate this computational complexity is to only update the coefficients once every N sample periods[9] reducing the average complexity (over N sample periods) to $2L + K_{inv}N$ multiplies per sample period. This is known as PRA, the partial rank algorithm[9]. Simulations indicate that when very highly colored excitation signals are used, the convergence of PRA is somewhat inferior to APA. For speech excitation, however, we have found that PRA achieves close to the same convergence as APA. The main disadvantage of PRA is that its computational complexity is bursty. So, depending on the speed of the implementing technology, there is often a delay in the generation of the error vector, $\underline{e}_n$. As will be shown below, FAP performs a complete N dimensional APA update each sample period with $2L + O(N)$ multiplies per sample; so there is no delay.

0-7803-2431-5/95 $4.00

3.1 Fast Residual Echo Vector Calculation

In this section we develop a fast method of updating the residual echo vector from sample period n-1 to sample period n.

We start by expanding the a priori error calculation of (1),

$$\underline{e}_n = \underline{s}_n - \mathbf{X}_n^t \underline{h}_{n-1} = \begin{bmatrix} s_n - \underline{x}_n^t \underline{h}_{n-1} \\ \underline{\bar{s}}_{n-1} - \bar{\mathbf{X}}_{n-1} \underline{h}_{n-1} \end{bmatrix} \quad (6)$$

where the L by N-1 matrix $\bar{\mathbf{X}}_{n-1}$ consists of the N-1 left-most columns of $\mathbf{X}_{n-1}$, and the N-1 length vector $\underline{\bar{s}}_{n-1}$ consists of the n-1 upper elements of the vector $\underline{s}_{n-1}$.

From (6) we see that the lower N-1 elements of the a priori residual echo, $\underline{e}_n$, are the upper N-1 elements of the a posteriori error vector of sample period n-1, $\underline{e}_{1,n-1}$ (where the 1 in the subscript denotes a posteriori rather than a priori error), defined as,

$$\underline{e}_{1,n-1} = \underline{s}_{n-1} - \mathbf{X}_{n-1}^t \underline{h}_{n-1} \quad (7)$$

$$= \left[\mathbf{I} - \mu \mathbf{X}_{n-1}^t \mathbf{X}_{n-1} \left[\mathbf{X}_{n-1}^t \mathbf{X}_{n-1} + \delta \mathbf{I} \right]^{-1} \right] \underline{e}_{n-1}. \quad (8)$$

The matrix $\mathbf{X}_{n-1}^t \mathbf{X}_{n-1}$ has the similarity decomposition,

$$\mathbf{X}_{n-1}^t \mathbf{X}_{n-1} = \mathbf{V}_{n-1} \Lambda_{n-1} \mathbf{V}_{n-1}^t \quad (9)$$

where $\mathbf{V}_{n-1}$ is an N by N unitary matrix and Λ_{n-1} is a N by N diagonal matrix with its i^{th} diagonal element being the i^{th} eigenvalue of $\mathbf{X}_{n-1}^t \mathbf{X}_{n-1}$, $\lambda_{i,n-1}$. Defining the a priori and a posteriori modal error vectors,

$$\underline{e}'_{n-1} = \mathbf{V}_{n-1}^t \underline{e}_{n-1} \quad \text{and} \quad \underline{e}'_{1,n-1} = \mathbf{V}_{n-1}^t \underline{e}_{1,n-1}, \quad (10)$$

respectively; we can multiply (8) from the left by $\mathbf{V}_{n-1}^t$ and show that the i^{th} a posteriori modal error vector element, $\underline{e}'_{1,i,n-1}$, can be found from the i^{th} a priori modal error vector element, $\underline{e}'_{i,n-1}$, by,

$$e'_{1,i,n-1} = \left[1 - \frac{\mu \lambda_{i,n-1}}{\delta + \lambda_{i,n-1}} \right] e'_{i,n-1}. \quad (11)$$

From (11) it can be shown that

$$e'_{1,i,n-1} \approx \begin{cases} (1-\mu) e'_{i,n-1} & \text{for } \lambda_{i,n-1} >> \delta \\ e'_{i,n-1} & \text{for } \lambda_{i,n-1} << \delta. \end{cases} \quad (12)$$

Assume that δ is chosen to be approximately equal to the power of $\underline{y}_n$. Then, for those modes where, $\lambda_{i,n-1} << \delta$, $e_{i,n-1}$ is mainly dominated by the background noise and little can be learned about $\underline{h}_{ep}$. So, suppressing these modes by multiplying them by $1-\mu$ will attenuate somewhat the background noise's effect on the overall echo path estimate[6]. Applying this to (12) and multiplying from the left by $\mathbf{V}_{n-1}$ we have

$$\underline{e}_{1,n-1} \approx (1-\mu) \underline{e}_{n-1}. \quad (13)$$

Replacing the lower $N-1$ elements of (6) with the upper $N-1$ elements of (13), we have the residual echo update,

$$\underline{e}_n \approx \begin{bmatrix} e_n \\ (1-\mu) \underline{\bar{e}}_{n-1} \end{bmatrix}. \quad (14)$$

From (8) we see that this approximation becomes an equality when $\delta = 0$, but then, the inverse in (2) is not regularized. Simulations show that by making adjustments in δ the convergence performance of APA with and without the approximation of equation (14) can be equated.

The complexity of (14) is L operations to calculate e_n and N-1 operations to update $(1-\mu)\underline{\bar{e}}_{n-1}$. For the case where $\mu = 1$ the N-1 operations are obviously unnecessary.

3.2 Fast Adaptive Coefficient Vector Calculation

This section discusses a method by which the fidelity of $\underline{e}_n$ is maintained at each sample period, but $\underline{h}_n$ is not. Here, we introduce an alternate coefficient vector, $\underline{\hat{h}}_n$, whose update each sample period consists only of adding a weighted version of the last column of $\mathbf{X}_n$[10]; just L multiplications as opposed to NL for the APA update of equation (3).

From (3) the APA tap update is,

$$\underline{h}_n = \underline{h}_{n-1} + \mu \mathbf{X}_n \underline{\varepsilon}_n. \quad (15)$$

One can also express the current echo path estimate, $\underline{h}_n$, in terms of the original echo path estimate, $\underline{h}_0$, and the subsequent $\mathbf{X}_i$'s and $\underline{\varepsilon}_i$'s,

$$\underline{h}_n = \underline{h}_0 + \mu \sum_{i=0}^{n-1} \mathbf{X}_{n-i} \underline{\varepsilon}_{n-i}. \quad (16)$$

Now, expand the vector/matrix multiplication,

$$\underline{h}_n = \underline{h}_0 + \mu \sum_{i=0}^{n-1} \sum_{j=0}^{N-1} \underline{x}_{n-j-i} \varepsilon_{j,n-i}. \quad (17)$$

Assuming that $x_n = 0$ for $n \le 0$, it can be shown that (17) can be rewritten as,

$$\underline{h}_n = \underline{h}_0 + \mu \sum_{k=0}^{N-1} \underline{x}_{n-k} \sum_{j=0}^{k} \varepsilon_{j,n-k+j} + \mu \sum_{k=N}^{n-1} \underline{x}_{n-k} \sum_{j=0}^{N-1} \varepsilon_{j,n-k+j}. \quad (18)$$

If the first term and the second pair of summations on the right side of (18) are defined as

$$\underline{\hat{h}}_{n-1} = \underline{h}_0 + \mu \sum_{k=N}^{n-1} \underline{x}_{n-k} \sum_{j=0}^{N-1} \varepsilon_{j,n-k+j} \quad (19)$$

and the first pair of summations in (18) is recognized as a vector-matrix multiplication,

$$\mathbf{X}_n \underline{E}_n = \mu \sum_{k=0}^{N-1} \underline{x}_{n-k} \sum_{j=0}^{k} \varepsilon_{j,n-k+j} \quad (20)$$

where,

$$\underline{E}_n = \begin{bmatrix} \varepsilon_{0,n} \\ \varepsilon_{1,n} + \varepsilon_{0,n-1} \\ \vdots \\ \varepsilon_{N-1,n} + \varepsilon_{N-2,n-1} + \cdots + \varepsilon_{0,n-(N-1)} \end{bmatrix} \quad (21)$$

then, (18) can be expressed as

$$\underline{h}_n = \underline{\hat{h}}_{n-1} + \mu \mathbf{X}_n \underline{E}_n. \quad (22)$$

It is easily seen from (19) that

$$\underline{\hat{h}}_n = \underline{\hat{h}}_{n-1} + \mu \underline{x}_{n-(N-1)} \sum_{j=0}^{N-1} \varepsilon_{j,n-N+1+j} \quad (23)$$

$$= \underline{\hat{h}}_{n-1} + \mu \underline{x}_{n-(N-1)} E_{N-1,n} \quad (24)$$

Using (24) in (22) the current echo path estimate can alternately be expressed as

$$\underline{h}_n = \underline{\hat{h}}_n + \mu \bar{\mathbf{X}}_n \bar{E}_n \quad (25)$$

Where $\bar{E}_n$ is an N-1 length vector consisting of the upper most N-1 elements of $\underline{E}_n$.

Observing (21) it is seen that $\underline{E}_n$ can also be calculated recursively. By inspection.

$$\underline{E}_n = \underline{\varepsilon}_n + \begin{bmatrix} 0 \\ \underline{\bar{E}}_{n-1} \end{bmatrix}. \quad (26)$$

Now, consider the relationship between $\underline{e}_n$ and $\underline{e}_{n-1}$. Using (14) one could calculate $\underline{e}_n$ from $\underline{e}_{n-1}$ except for the fact that $\underline{h}_{n-1}$ is not readily available. However, using (25) for $\underline{h}_{n-1}$ in the first row of (14), e_n is,

$$e_n = \hat{e}_n - \mu \tilde{\underline{r}}^t_{xx,n} \underline{\bar{E}}_{n-1} \quad (27)$$

where

$$\hat{e}_n = s_n - \underline{x}^t_n \hat{\underline{h}}_{n-1}, \quad (28)$$

$$\tilde{\underline{r}}_{xx,n} = \tilde{\underline{r}}_{xx,n-1} + x_n \tilde{\underline{\alpha}}_n - x_{n-L} \tilde{\underline{\alpha}}_{n-L}, \quad (29)$$

and $\tilde{\underline{\alpha}}_n = \left[x_{n-1}, \ldots, x_{n-N+1} \right]^t$.

Relations (24) and (26) represent the efficient alternate coefficient update, $\hat{\underline{h}}_n$, assuming that there is an efficient update for $\underline{\varepsilon}_n$. Together these require $L+N$ multiplications. Relations (27) through (29) represent an efficient method to calculate e_n using $\hat{\underline{h}}_n$ rather than $\underline{h}_n$. These require $L+3N$ multiplications.

3.3 Fast Normalized Residual Echo Vector Calculation

We now turn to the problem of efficiently updating the normalized residual echo vector, $\underline{\varepsilon}_n$. Define $\mathbf{R}_n = \mathbf{X}^t_n \mathbf{X}_n + \delta \mathbf{I}$, let $\underline{a}_n$ and $\underline{b}_n$ denote the optimum forward and backward linear predictors for $\mathbf{R}_n$, and let $E_{a,n}$ and $E_{b,n}$ denote their respective expected prediction error energies. Also, define $\bar{\mathbf{R}}_n$ and $\tilde{\mathbf{R}}_n$ as N-1 by N-1 matrices consisting of the upper left and lower right corners of $\mathbf{R}_n$, respectively. Then, given the following identities:

$$\mathbf{R}^{-1}_n = \begin{bmatrix} 0 & \underline{0}^t \\ \underline{0} & \tilde{\mathbf{R}}^{-1}_n \end{bmatrix} + \frac{1}{E_{a,n}} \underline{a}_n \underline{a}^t_n = \begin{bmatrix} \bar{\mathbf{R}}^{-1}_n & \underline{0} \\ \underline{0}^t & 0 \end{bmatrix} + \frac{1}{E_{b,n}} \underline{b}_n \underline{b}^t_n \quad (30)$$

and the definitions,

$$\tilde{\underline{\varepsilon}}_n = \tilde{\mathbf{R}}^{-1}_n \tilde{\underline{e}}_n \quad \text{and} \quad \bar{\underline{\varepsilon}}_n = \bar{\mathbf{R}}^{-1}_n \bar{\underline{e}}_n, \quad (31)$$

(where $\tilde{\underline{e}}_n$ is an N-1 length vector containing the N-1 lower elements of $\underline{e}_n$), one can multiply (30) from the right by $\underline{e}_n$ and use (2) and (31) to obtain,

$$\underline{\varepsilon}_n = \begin{bmatrix} 0 \\ \tilde{\underline{\varepsilon}}_n \end{bmatrix} + \frac{1}{E_{a,n}} \underline{a}_n \underline{a}^t_n \underline{e}_n \quad (32)$$

and

$$\begin{bmatrix} \bar{\underline{\varepsilon}}_n \\ 0 \end{bmatrix} = \underline{\varepsilon}_n - \frac{1}{E_{b,n}} \underline{b}_n \underline{b}^t_n \underline{e}_n. \quad (33)$$

The quantities, $E_{a,n}$, $E_{b,n}$, $\underline{a}_n$, and $\underline{b}_n$ can be calculated efficiently (complexity 10N) using a sliding windowed FRLS algorithm[8].

The relationship between $\tilde{\underline{\varepsilon}}_n$ and $\bar{\underline{\varepsilon}}_{n-1}$ is now investigated. It can easily be shown that

$$\tilde{\mathbf{R}}_n = \bar{\mathbf{R}}_{n-1}. \quad (34)$$

Using (34), the definition of $\tilde{\underline{e}}_n$, $\bar{\underline{e}}_n$, (31), and (14) we have,

$$\tilde{\underline{\varepsilon}}_n = \tilde{\mathbf{R}}_n \tilde{\underline{e}}_n = \bar{\mathbf{R}}_{n-1} (1-\mu) \bar{\underline{e}}_{n-1} = (1-\mu) \bar{\underline{\varepsilon}}_{n-1}. \quad (35)$$

3.4 FAP

The relations derived above are brought together in the relaxed FAP algorithm shown in ALGORITHM 1.

Step 1 of the algorithm is of complexity 10N multiplications. Steps 3 and 9 are both of complexity L, steps 2, 6, and 7 are each of complexity 2N, and steps 4, 5, 8 and 10 are of complexity N. This gives an overall complexity of 2L+20N multiplications per sample period. If relaxation is eliminated, that is, μ is set to one, the computational complexity can be reduced to 2L+14N multiplications per sample period[5][6].

ALGORITHM 1: FAP

0) *Initialization*: $E_{a,n} = E_{b,n} = \delta$ and $\underline{a}_0 = [1, \underline{0}^t]^t$, $\underline{b}_0 = [\underline{0}^t, 1]^t$.

1) *Use sliding windowed FRLS to update* $E_{a,n}$, $E_{b,n}$, $\underline{a}_n$, and $\underline{b}_n$.

2) $\tilde{\underline{r}}_{xx,n} = \tilde{\underline{r}}_{xx,n-1} + x_n \tilde{\underline{\alpha}}_n - x_{n-L} \tilde{\underline{\alpha}}_{n-L}$ (29)

3) $\hat{e}_n = s_n - \underline{x}^t_n \hat{\underline{h}}_{n-1}$ (28)

4) $e_n = \hat{e}_n - \mu \tilde{\underline{r}}^t_{xx,n} \bar{\underline{E}}_{n-1}$ (27)

5) $\underline{e}_n = \begin{bmatrix} e_n \\ (1-\mu) \bar{\underline{e}}_{n-1} \end{bmatrix}$ (14)

6) $\underline{\varepsilon}_n = \begin{bmatrix} 0 \\ \tilde{\underline{\varepsilon}}_n \end{bmatrix} + \frac{1}{E_{a,n}} \underline{a}_n \underline{a}^t_n \underline{e}_n$ (32)

7) $\begin{bmatrix} \bar{\underline{\varepsilon}}_n \\ 0 \end{bmatrix} = \underline{\varepsilon}_n - \frac{1}{E_{b,n}} \underline{b}_n \underline{b}^t_n \underline{e}_n$ (33)

8) $\underline{E}_n = \begin{bmatrix} 0 \\ \bar{\underline{E}}_{n-1} \end{bmatrix} + \underline{\varepsilon}_n$ (26)

9) $\hat{\underline{h}}_n = \hat{\underline{h}}_{n-1} + \mu \underline{x}_{n-(N-1)} E_{N-1,n}$ (24)

10) $\tilde{\underline{\varepsilon}}_{n+1} = (1-\mu) \bar{\underline{\varepsilon}}_n$ (35)

4. SIMULATIONS

Figure 1 shows a comparison of the convergence of NLMS, FTF (Fast Transversal Filter, an FRLS technique), and FAP coefficient error magnitudes. The excitation signal was speech sampled at 8 KHz, the echo path of length, $L = 1000$, was fixed and the white gaussian additive noise, y_n, was 30 dB down from the the echo. Soft initialization was used for both algorithms. For FTF, $E_{a,0}$ and $E_{b,0}$ were both set to $2\sigma^2_x$ (where σ^2_x is the average power of x_n) and λ, the forgetting factor was set to $(3L-1)/3L$. For FAP, $E_{a,0}$ and $E_{b,0}$ were set to $\delta = 20\sigma^2_x$ and N was 50. FAP converges at roughly the same rate as FTF with about 2L complexity versus 7L complexity, respectively. Both FAP and FTF converge faster than NLMS.

In figure 2 we show the convergence of NLMS and FAP with various orders of projections. Once again, speech was the excitation, the length of the filter was 1000 samples, and the signal to noise ratio was 30 dB. We see that quite a bit of improvement is gained with just N=2 and that increasing N to 10 does not improve the speed of convergence significantly. However, if N is further increased to 50, there is again a significant gain in the speed of convergence. Note that for FAP, the increase from N=2 to N=50 does not significantly increase the computational complexity. Thus, very perceptible increases in convergence are realized with only moderate increases in computational complexity.

5. NUMERICAL CONSIDERATIONS

FAP uses the sliding window technique to update and downdate data in its implicit regularized sample correlation

matrix and cross correlation vector. Errors introduced by finite arithmetic in practical implementations of the algorithm therefore cause the correlation matrix and cross correlation vector to take random walks with respect to their infinite precision counterparts. A stablized sliding windowed FRLS algorithm[11] has been introduced, with complexity 14N multiplications per sample period (rather than 10N for non-stablized versions). However, even this algorithm is stable only for stationary signals, a class of signals which certainly does not include speech. Another approach, which is very straightforward and rather elegant for FAP, is to periodically start a new sliding window in parallel with the old sliding window, and when the data is the same in both processes, replace the old sliding window based parameters with the new ones. Although this increases the sliding window based parameter calculations by about 50% on average (assuming the restarting is done every L+N sample periods), the overall cost is small since only those parameters with computational complexity proportional to N are affected. The overall complexity is only 2L+21N for FAP without relaxation and 2L+30N for FAP with relaxation. Since this approach is basically a periodic restart, it is numerically stable for all signals.

6. CONCLUSIONS

This paper has discussed a fast version of the affine projection algorithm, FAP. When the length of the adaptive filter is L and the dimension of the affine projection (performed each sample period) is N, FAP's complexity is either 2L+14N or 2L+20N depending on whether the relaxation parameter is one or smaller, respectively. Simulations demonstrate that FAP converges as fast as FRLS methods when the excitation signal is speech. The implicit correlation matrix inverse of FAP is regularized, so the algorithm is easily stabilized for even highly colored excitation. Finally, a simple, low complexity numerical stabilazation method for the algorithm was also introduced.

7. ACKNOWLEDGEMENTS

The authors would like to thank Drs. M. M. Sondhi, J. Cezanne, D. R. Morgan, and G. Kubin for many fruitful discussions.

REFERENCES

1. K. Ozeki, T. Umeda, "An Adaptive Filtering Algorithm Using an Orthogonal Projection to an Affine Subspace and Its Properties," Electronics and Communications in Japan, Vol. 67-A, No. 5, 1984.
2. B. Widrow, S. D. Stearns, "Adaptive Signal Processing," Prentice-Hall, Inc. Englewood Cliffs, N.J. 1985.
3. J. M. Cioffi, T. Kailath, "Fast, Recursive-Least-Squares Transversal Filters for Adaptive Filtering," IEEE Trans on Acoustics, Speech, and Signal Proc., Vol. Assp-32, No. 2, April 1984.
4. Orfanidis, Sophocles J., "Optimum Signal Processing, An Introduction," MacMillan, New York, 1985.
5. Gay, Steven L., "A Fast Converging, Low Complexity Adaptive Filtering Algorithm," Lannion, Sept. 1993.
6. Gay, Steven L., "Fast Projection Algorithms with Application to Voice Excited Echo Cancellers," Ph.D. Dissertation, Rutgers University, Piscataway, New Jersey, October, 1994.
7. Tanaka, M., Kaneda, Y., Makino, S., "Reduction of Computation for High-Order Projection Algorithm," 1993 Electronics Information Communication Society Autumn Seminar, Tokyo, Japan (in Japanese)
8. J. M. Cioffi, T. Kailath, "Windowed Fast Transversal Filters Adaptive Algorithms with Normalization," IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-33, No. 3, June 1985.
9. S. G. Kratzer, D. R. R. Morgan, "The Partial-Rank Algorithm for Adaptive Beamforming," SPIE Vol. 564, Real Time Signal Processing VIII (1985).
10. Maruyama Y, "A fast method of projection algorithm," Proc. 1990 IEICE Spring Conf., B-744.
11. Slock, D. T. M. and Kailath, T., "Numerically Stable Transversal Filters for Recursive Least Squares Adaptive Filtering," IEEE Trans. on Signal Processing, Vol. 39, No. 1, Jan. 1991.

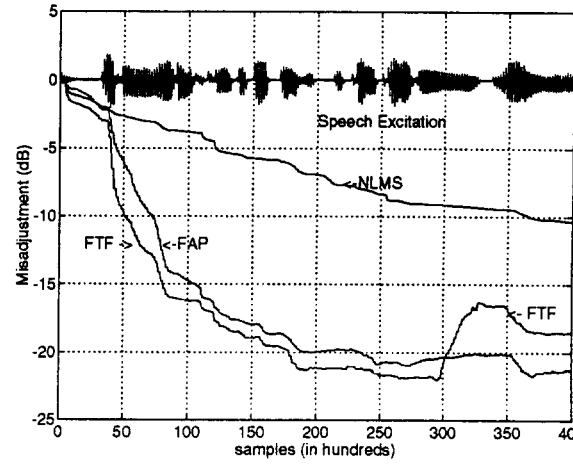

Figure 1. Comparison of coefficient error for FAP, FTF, and NLMS with speech as excitation.

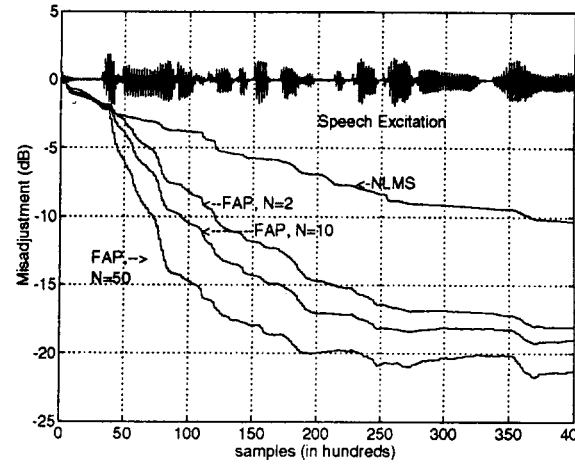

Figure 2. Comparison of FAP for different orders of projection, N, with speech excitation