1-1-1993

A Fast Converging, Low Complexity Adaptive Filtering Algorithm

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

Recommended Citation

S. L. Grant, "A Fast Converging, Low Complexity Adaptive Filtering Algorithm," *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 1993*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1993.
The definitive version is available at https://doi.org/10.1109/ASPAA.1993.380010

A FAST CONVERGING, LOW COMPLEXITY ADAPTIVE FILTERING ALGORITHM

Steven L. Gay

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

Abstract: This paper introduces a new adaptive filtering algorithm called fast affine projections (FAP). Its main attributes include RLS (recursive least squares) like convergence and tracking with NLMS (normalized least mean squares) like complexity. This mix of complexity and performance is similar to the recently introduced Fast Newton Transversal Filter (FNTF) algorithm. While FAP shares some similar properties with FNTF it is derived from a different perspective, namely the generalization of the affine projection interpretation of NLMS. FAP relies on a sliding windowed fast RLS (FRLS) algorithm to generate forward and backward prediction vectors and expected prediction error energies. Since sliding windowed FRLS algorithms easily incorporate regularization of the implicit inverse of the covariance matrix, FAP is regularized as well.

1. Introduction

Recently, the Fast Newton Transversal Filter (FNTF) algorithm[1], a generalization of both the normalized least mean squares[2] (NLMS) and the recursive least squares[3] (RLS) algorithms has been introduced. FNTF exploits an assumption that the process excitation signal is the output of an N'th order autoregressive model. Under such assumptions, linear predictors, such as those used in the fast RLS[4] (FRLS) algorithms, may be limited to length N-1 with no loss of performance. FNTF uses this property to reduce computational complexity from 7L (where L is the length of the joint process estimation) to 2L+7N. This includes an additional memory requirement of N(L-N). Alternately, the additional memory usage may be avoided with an increase in complexity to 2L+12N. When N=1, FNTF reverts to the NLMS algorithm and when N=L, FNTF becomes FRLS.

Another direction in the generalization of NLMS is the so-called affine projection algorithm (APA)[5] [6]. Under this interpretation, each tap update of NLMS is viewed as a one dimensional affine projection. APA considers the case where projections are made in multiple dimensions per tap update. As with RLS and FNTF, APA requires an implicit inverse of the excitation signal's covariance matrix, although with APA the dimension of the covariance matrix is the same as the dimension of the projection, N, not the length of the joint process estimation, L. Using techniques similar to those which led to FRLS from RLS, a fast version of APA, FAP may be derived. Moreover, FAP relies on a sliding windowed FRLS[7] to generate forward and backward prediction vectors and expected prediction error energies. Since sliding windowed FRLS algorithms easily incorporate regularization of the covariance matrix inverse[8], FAP is regularized as well. The complexity of FAP is roughly that of FNTF, 2L+14N, however FAP does not require significantly greater memory than NLMS. In many applications, especially those involving speech, L is often much larger than the required N. The echo cancellation problem is a good example of such an application and the algorithms developed in this paper are presented in that framework. Simulations indicate that FAP's convergence and tracking of system changes are comparable in speed to FRLS methods when N is greater than the order of the excitation signal model.

2. The Affine Projection Algorithm

The affine projection algorithm, in a relaxed and regularized form, is defined by the following two equations:

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

$$\underline{h}_n = \underline{h}_{n-M} + \mu \mathbf{X}_n \left[\mathbf{X}_n^t \mathbf{X}_n + \delta \mathbf{I} \right]^{-1} \underline{e}_n . \tag{2}$$

This is a block algorithm where the block size is N. The excitation signal matrix, $\mathbf{X}_n$, is L by N and has the structure,

$$\mathbf{X}_n = \left[\underline{x}_n, \underline{x}_{n-1}, \cdots \underline{x}_{n-(N-1)} \right] \tag{3}$$

where the $\underline{x}_n = \left[x_n, \cdots x_{n-L+1} \right]^t$ 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, $\underline{y}_n$.

$$\underline{s}_n = \mathbf{X}_n^t \underline{h}_{ep} + \underline{y}_n . \tag{4}$$

The step-size parameter, μ is the relaxation factor as in NLMS. The algorithm is stable for $0 \le \mu < 2$.

The scalar δ is the regularization parameter for the autocorrelation matrix inverse. 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.

At each iteration, the data in the algorithm moves forward by M samples. Usually M is set to the block size, N, to mitigate the $O(N^2)$ computational complexity of the calculation of the vector

$$\underline{\varepsilon}_n = \left[\mathbf{X}_n^t \mathbf{X}_n + \delta \mathbf{I} \right]^{-1} \underline{e}_n = \left[\varepsilon_{0,n}, \cdots, \varepsilon_{N-1,n} \right]^t . \tag{5}$$

However faster convergence and/or a lower converged system misadjustment can be achieved if M is set to a lower value. FAP achieves $O(N)$ complexity for the calculation of $\underline{\varepsilon}_n$ when M is set to one.

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

Projections Onto An Affine Subspace and Convergence

By manipulating equations (1), (2), and (4), and assuming that δ is small enough to be ignored and the additive system noise, $\underline{y}_n$ is zero, the APA tap update can be expressed as,

$$\underline{h}_n = \mathbf{Q}_n \underline{h}_{n-M} + \mathbf{P}_n \underline{h}_{ep} \tag{7}$$

where

$$\mathbf{Q}_n = \mathbf{I} - \mu \mathbf{X}_n \left[\mathbf{X}_n' \mathbf{X}_n\right]^{-1} \mathbf{X}_n' = \mathbf{U}_n \, diag\left\{1-\mu, \cdots, 1-\mu, 1, \cdots, 1\right\} \mathbf{U}_n', \tag{8}$$

$$\mathbf{P}_n = \mathbf{I} - \mathbf{Q}_n, \tag{9}$$

and the diagonal matrix in (8) has N $(1-\mu)$'s and L-N 1's along the diagonal. The matrices, $\mathbf{Q}_n$ and $\mathbf{P}_n$ represent projection matrices onto orthogonal subspaces when $\mu = 1$ and relaxed projection matrices when $0 < \mu < 1$. Thus (7), and therefore (1) and (2), represent the (relaxed) projection of $\underline{h}_{n-M}$ onto the affine subspace defined by the (relaxed) linear projection matrix, $\mathbf{Q}_n$, and the offset vector, $\mathbf{P}_n \underline{h}_{ep}$.

Equation (7) provides insight into the convergence of $\underline{h}_n$ to $\underline{h}_{ep}$. Assume that $\mu = 1$. As N increases from 1 toward L, the contribution to $\underline{h}_n$ from $\underline{h}_{n-M}$ decreases because the nullity of $\mathbf{Q}_n$ is increasing, while the contribution from $\underline{h}_{ep}$ increases because the rank of $\mathbf{P}_n$ is increasing. In principle, when $N = L$, $\underline{h}_n$ should converge to $\underline{h}_{ep}$ in one step, since $\mathbf{Q}_n$ has a rank of zero and $\mathbf{P}_n$ a rank of L. In practice however, as N approaches L the condition number of the matrix, $\mathbf{X}_n' \mathbf{X}_n$ begins to grow. As a result, the inverse of $\mathbf{X}_n' \mathbf{X}_n$ becomes more and more dubious and must be replaced with either a regularized or pseudo-inverse.

The Connection Between APA, LS, and RLS

This section shows the connection between APA, least squares (LS), and RLS. Using the matrix inversion lemma it can be shown that the APA tap update described in (2) can be written as[9]

$$\underline{h}_n = \underline{h}_{n-M} + \mu \left[\mathbf{X}_n \mathbf{X}_n' + \delta \mathbf{I}\right]^{-1} \mathbf{X}_n \underline{e}_n \tag{11}$$

Note that $\mathbf{X}_n \mathbf{X}_n'$ is an L by L rank deficient estimate of the autocorrelation matrix and that its inverse is regularized by the matrix $\delta \mathbf{I}$. Now, consider the case where the sample advance, M, is one and consider the vector $\underline{e}_n$. By definition,

$$\underline{e}_n = \underline{s}_n - \mathbf{X}_n' \underline{h}_{n-1} = \begin{bmatrix} s_n - \underline{x}_n' \underline{h}_{n-1} \\ \overline{\underline{s}}_{n-1} - \overline{\mathbf{X}}_{n-1}' \underline{h}_{n-1} \end{bmatrix}. \tag{12}$$

Where the matrix $\overline{\mathbf{X}}_{n-1}$ has dimension L by (N-1) and consists of the N-1 left-most (newest) columns of $\mathbf{X}_{n-1}$ and the N-1 length vector $\overline{\underline{s}}_{n-1}$ consists of the N-1 upper (newest) elements of the vector $\underline{s}_{n-1}$.

First, consider the lower $N-1$ elements of (12). Define the a posteriori residual echo vector for sample period n-1, $\underline{e}_{1,n-1}$ as

$$\underline{e}_{1,n-1} = \underline{s}_{n-1} - \mathbf{X}_{n-1}' \underline{h}_{n-1} = \left[\mathbf{I} - \mu \mathbf{X}_{n-1}' \mathbf{X}_{n-1} \left[\mathbf{X}_{n-1}' \mathbf{X}_{n-1} + \delta \mathbf{I}\right]^{-1}\right] \underline{e}_{n-1}. \tag{13}$$

Now make the approximation,

$$\mathbf{X}_{n-1}' \mathbf{X}_{n-1} + \delta \mathbf{I} \approx \mathbf{X}_{n-1}' \mathbf{X}_{n-1} \tag{14}$$

which is valid as long as δ is significantly smaller than the eigenvalues of $\mathbf{X}_{n-1}' \mathbf{X}_{n-1}$. Of course, this means that $\delta \mathbf{I}$ no longer regularizes the inverse of the N by N sample autocorrelation matrix. However it still regularizes the rank deficient L by L sample autocorrelation matrix inverse of (11). Using this approximation,

$$\underline{e}_{1,n-1} \approx (1-\mu) \underline{e}_{n-1}. \tag{15}$$

Recognizing that the lower $N-1$ elements of (12) are the same as the upper $N-1$ elements of (13), (15) can be used to express $\underline{e}_n$ as

$$\underline{e}_n \approx \begin{bmatrix} s_n - \underline{x}_n' \underline{h}_{n-1} \\ (1-\mu) \overline{\underline{e}}_{n-1} \end{bmatrix} = \begin{bmatrix} e_n \\ (1-\mu) \overline{\underline{e}}_{n-1} \end{bmatrix}. \tag{16}$$

Where $\overline{\underline{e}}_{n-1}$ is an N-1 length vector containing the uppermost elements of $\underline{e}_{n-1}$. This is a key approximation and is essential in the derivation of FAP in the next section. For $\mu = 1$,

$$\underline{e}_n \approx \begin{bmatrix} e_n \\ \underline{0} \end{bmatrix}. \tag{17}$$

Using (17) in (11),

$$\underline{h}_n = \underline{h}_{n-1} + \left[\mathbf{X}_n \mathbf{X}_n' + \delta \mathbf{I}\right]^{-1} \underline{x}_n e_n. \tag{18}$$

Equation (18) is a regularized, rank deficient least squares tap update. If $N = n$, $\delta = 0$, and the matrix inversion lemma is applied to a rank-one update of the inverted matrix in (18), (18) becomes the growing windowed RLS algorithm.

Indeed, equation (18) can be used as an alternative starting point to our algorithm and FAP (with $\mu = 1$) can be thought of as a fast, regularized, rank deficient least squares algorithm. One advantage of this interpretation is that one can work backwards from (18) and obtain (11) where an alternative definition for $\underline{e}_n$, namely, $\underline{e}_n = \begin{bmatrix} e_n \\ \underline{0} \end{bmatrix}$ is used. As a side benefit, this obviates the need for the approximation in (14) and δ can once again be chosen large enough to regularize small eigenvalues in $\mathbf{X}_n' \mathbf{X}_n$. While the two algorithms represented by relations (2) and (18) are slightly different, they yield the same convergence curves with only a modification of the parameter δ.

3. The Fast Affine Projection Algorithm

In this section two fast APA algorithms are derived, one with relaxation, $0 < \mu < 1$, and one without relaxation, $\mu = 1$. In both cases the sample advance, M, is set to one.

Both the relaxed and non-relaxed algorithms can be thought of as close approximations to a fast APA when δ is set to a small value compared to the smaller eigenvalues of $\mathbf{X}_n' \mathbf{X}_n$. Alternately, the fast algorithm without relaxation can be thought of as an exact fast regularized rank deficient least squares algorithm. These philosophical considerations are a direct result of whether relation (16) is taken as an approximation or an equality.

In the echo cancellation problem, as in other problems, the residual echo, $\underline{e}_n$ is the directly observed part of the algorithm, and the adaptive filter taps, $\underline{h}_n$ are "buried," so to speak, in the RAM of some signal processor somewhere.

Therefore, it is permissible to maintain any form of $\underline{h}_n$ that is convenient in the algorithm, as long as the first sample of $\underline{e}_n$ is not modified in any way from that of equation (1). This is the basis of FAP. The fidelity of $\underline{e}_n$ is maintained at each sample period, but $\underline{h}_n$ is not. Another vector, $\underline{h}_n$ is maintained, where only the last column of $\mathbf{X}_n$ is weighted and accumulated into $\underline{h}_n$ each sample period. Thus, the computational complexity of the tap-update process is no more complex than NLMS, L operations. In addition, the residual echo vector calculation is shown to be recursive with $L+O(N)$ complexity.

Using (5) in (2) the APA tap update can be expressed as,

$$\underline{h}_n = \underline{h}_{n-1} + \mu \mathbf{X}_n \underline{\varepsilon}_n . \tag{20}$$

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,

$$\underline{h}_n = \underline{h}_0 + \mu \sum_{i=0}^{n-1} \mathbf{X}_{n-i} \underline{\varepsilon}_{n-i} . \tag{21}$$

Now, expand the vector/matrix multiplication,

$$\underline{h}_n = \underline{h}_0 + \mu \sum_{i=0}^{n-1} \sum_{j=0}^{N-1} \underline{x}_{n-j-i} \varepsilon_{j,n-i} . \tag{22}$$

Assuming that $x_n = 0$ for $n \le 0$, it can be shown that (22) can be rewritten as,

$$\underline{h}_n = \underline{h}_0 + \mu \sum_{k=0}^{N-1} \underline{x}_{n-k} \sum_{j=0}^{k} \varepsilon_{j,n-k+j} + \mu \sum_{k=N}^{n-1} \underline{x}_{n-k} \sum_{j=0}^{N-1} \varepsilon_{j,n-k+j} . \tag{23}$$

If the first term and the second pair of summations on the right side of (23) are defined as

$$\hat{\underline{h}}_{n-1} = \underline{h}_0 + \mu \sum_{k=N}^{n-1} \underline{x}_{n-k} \sum_{j=0}^{N-1} \varepsilon_{j,n-k+j} \tag{24}$$

and the first pair of summations in (23) is recognized as a vector-matrix multiplication,

$$\mathbf{X}_n \underline{E}_n = \mu \sum_{k=0}^{N-1} \underline{x}_{n-k} \sum_{j=0}^{k} \varepsilon_{j,n-k+j} \tag{25}$$

where,

$$\underline{E}_n = \mu \begin{bmatrix} \varepsilon_{0,n} \\ \varepsilon_{1,n} + \varepsilon_{0,n-1} \\ \vdots \\ \varepsilon_{N-1,n} + \varepsilon_{N-2,n-1} + \cdots + \varepsilon_{0,n-(N-1)} \end{bmatrix} \tag{26}$$

then, (23) can be expressed as

$$\underline{h}_n = \hat{\underline{h}}_{n-1} + \mathbf{X}_n \underline{E}_n . \tag{27}$$

It is easily seen from (24) that

$$\hat{\underline{h}}_n = \hat{\underline{h}}_{n-1} + \mu \underline{x}_{n-(N-1)} \sum_{j=0}^{N-1} \varepsilon_{j,n-N+1+j} \tag{28}$$

$$= \hat{\underline{h}}_{n-1} + \underline{x}_{n-(N-1)} E_{N-1,n} \tag{29}$$

Using (29) in (27) the current echo path estimate can alternately be expressed as

$$\underline{h}_n = \hat{\underline{h}}_n + \bar{\mathbf{X}}_n \bar{\underline{E}}_n \tag{30}$$

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

Observing (26) it is seen that $\underline{E}_n$ can also be calculated recursively. By inspection,

$$\underline{E}_n = \mu \underline{\varepsilon}_n + \begin{bmatrix} 0 \\ \bar{\underline{E}}_{n-1} \end{bmatrix} . \tag{31}$$

Now, consider the relationship between $\underline{e}_n$ and $\underline{e}_{n-1}$. Using (16) 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 (30) for $\underline{h}_{n-1}$ in the first row of (16), e_n is,

$$e_n = \hat{e}_n - \bar{\underline{r}}'_{xx,n} \bar{\underline{E}}_{n-1} \tag{33}$$

where

$$\hat{e}_n = s_n - \underline{x}'_n \hat{\underline{h}}_{n-1} , \tag{34}$$

$$\bar{\underline{r}}_{xx,n} = \bar{\underline{r}}_{xx,n-1} + x_n \bar{\underline{\alpha}}_n - x_{n-L} \bar{\underline{\alpha}}_{n-L} , \tag{35}$$

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

To efficiently compute (31) one needs to find a recursion for the vector $\underline{\varepsilon}_n$. Define $\mathbf{R}_n = \mathbf{X}'_n \mathbf{X}_n$ and 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 $\bar{\mathbf{R}}_n$ 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:

$$\mathbf{R}_n^{-1} = \begin{bmatrix} 0 & \underline{0}' \\ \underline{0} & \bar{\mathbf{R}}_n^{-1} \end{bmatrix} + \frac{1}{E_{a,n}} \underline{a}_n \underline{a}'_n \tag{36}$$

$$= \begin{bmatrix} \bar{\mathbf{R}}_n^{-1} & \underline{0} \\ \underline{0}' & 0 \end{bmatrix} + \frac{1}{E_{b,n}} \underline{b}_n \underline{b}'_n \tag{37}$$

and the definitions,

$$\bar{\underline{\varepsilon}}_n = \bar{\mathbf{R}}_n^{-1} \bar{\underline{e}}_n \tag{38}$$

(where $\bar{\underline{e}}_n$ is an N-1 length vector containing the N-1 lower elements of $\underline{e}_n$) and

$$\bar{\underline{\varepsilon}}_n = \bar{\mathbf{R}}_n^{-1} \bar{\underline{e}}_n , \tag{39}$$

one can multiply (36) from the right by $\underline{e}_n$ and use (5) and (38) to obtain,

$$\underline{\varepsilon}_n = \begin{bmatrix} 0 \\ \bar{\underline{\varepsilon}}_n \end{bmatrix} + \frac{1}{E_{a,n}} \underline{a}_n \underline{a}'_n \underline{e}_n . \tag{40}$$

Similarly, multiplying (37) from the right by $\underline{e}_n$, using (5) and (39), and solving for $\begin{bmatrix} \bar{\underline{\varepsilon}}_n \\ 0 \end{bmatrix}$,

$$\begin{bmatrix} \bar{\underline{\varepsilon}}_n \\ 0 \end{bmatrix} = \underline{\varepsilon}_n - \frac{1}{E_{b,n}} \underline{b}_n \underline{b}'_n \underline{e}_n . \tag{41}$$

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[7].

The relations derived above can be brought together in the relaxed FAP algorithm as follows:

1. Use sliding windowed Fast Kalman or Fast Transversal Filter algorithms to update $E_{a,n}$, $E_{b,n}$, $\underline{a}_n$, and $\underline{b}_n$.

2. $\tilde{r}_{xx,n}=\tilde{r}_{xx,n-1}+x_n\tilde{\alpha}_n-x_{n-L}\tilde{\alpha}_{n-L}$
3. $\hat{e}_n=s_n-\underline{x}_n^t\hat{\underline{h}}_{n-1}$
4. $e_n=\hat{e}_n-\tilde{r}_{xx,n}^t\bar{\underline{E}}_{n-1}$
5. $\underline{e}_n=\begin{bmatrix} e_n \\ (1-\mu)\bar{\underline{e}}_{n-1}\end{bmatrix}$
6. $\underline{\varepsilon}_n=\begin{bmatrix}0\\ \bar{\underline{\varepsilon}}_n\end{bmatrix}+\frac{1}{E_{a,n}}\underline{a}_n\underline{a}_n^t\underline{e}_n$
7. $\begin{bmatrix}\bar{\underline{\varepsilon}}_n\\ 0\end{bmatrix}=\underline{\varepsilon}_n-\frac{1}{E_{b,n}}\underline{b}_n\underline{b}_n^t\underline{e}_n$
8. $\underline{E}_n=\begin{bmatrix}0\\ \bar{\underline{E}}_{n-1}\end{bmatrix}+\mu\underline{\varepsilon}_n$
9. $\hat{\underline{h}}_n=\hat{\underline{h}}_{n-1}+\underline{x}_{n-(N-1)}E_{N-1,n}$
10. $\bar{\underline{\varepsilon}}_{n+1}=(1-\mu)\bar{\underline{\varepsilon}}_n$

Step 1 is of complexity 10N when FTF is used. 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.

If relaxation is eliminated, that is, μ is set to one, considerable savings in complexity can be realized. FAP without relaxation may be written as follows:

1. Use sliding windowed Fast Kalman or Fast Transversal Filter algorithms to update $E_{a,n}$ and $\underline{a}_n$.
2. $\tilde{r}_{xx,n}=\tilde{r}_{xx,n-1}+x_n\tilde{\alpha}_n-x_{n-L}\tilde{\alpha}_{n-L}$
3. $\hat{e}_n=s_n-\underline{x}_n^t\hat{\underline{h}}_{n-1}$
4. $e_n=\hat{e}_n-\tilde{r}_{xx,n}^t\bar{\underline{E}}_{n-1}$
5. $\underline{E}_n=\begin{bmatrix}0\\ \bar{\underline{E}}_{n-1}\end{bmatrix}+\frac{e_n}{E_{a,n}}\underline{a}_n$
6. $\hat{\underline{h}}_n=\hat{\underline{h}}_{n-1}+\underline{x}_{n-(N-1)}E_{N-1,n}$

Here, steps 3 and 6 are still complexity L, step 2 is of complexity 2N, and steps 4 and 5 are of complexity N. Taking into account the sliding windowed FTF, the total complexity is 2L+14N.

4. Simulations

Figure 1 shows a comparison of initial convergence between FTF (Fast Transversal Filter, an FRLS technique) and FAP where the system misadjustment is plotted versus time. The echo path of length, $L=100$, is fixed and the additive noise, y_n, is 40 dB down from the of the echo , $\underline{h}_{ep}^t\underline{x}_n$. The excitation signal is a colored noise sequence generated by sending white Gaussian noise through the filter $1/(1-.98z^{-11})$. Soft initialization was used for both algorithms. For FTF, $E_{a,0}$ and $E_{b,0}$ were both set to $2\sigma_x^2$ (where σ_x^2 is the power of x_n) and λ, the forgetting factor was set to 0.9995. For FAP, $E_{a,0}$ and $E_{b,0}$ were set to $8\sigma_x^2$ and N was set to 12. FAP converges at roughly the same rate as FTF with 2L complexity versus 7L complexity, respectively. Both FAP and FTF converge faster than NLMS.

5. Acknowledgements

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

REFERENCES

1. G. V. Moustakides, S. Theordoridis, "Fast Newton Transversal Filters -- A New Class of Adaptive Estimation Algorithms." IEEE Trans. on Signal Proc. Vol 39, No. 10, Oct. 1991.
2. B. Widrow, S. D. Stearns, "Adaptive Signal Processing," Prentice-Hall, Inc. Englewood Cliffs, N.J. 1985.
3. Orfanidis, "Optimum Signal Processing, An Introduction," MacMillan, New York, 1985.
4. 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.
5. 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.
6. P. C. W. Sommen, "Adaptive Filtering Methods," Ph.D. Dissertation, Technische Universiteit Eindhoven, June 1992.
7. 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.
8. A. Houacine, "Regularized Fast Recursive Least Squares Algorithms for Adaptive Filtering," IEEE Trans. Signal Processing, Vol. 39, No. 4, April. 1991.
9. D. Morgan, AT&T Bell Laboratories, Murray Hill, N.J., personal communication, April 1993.

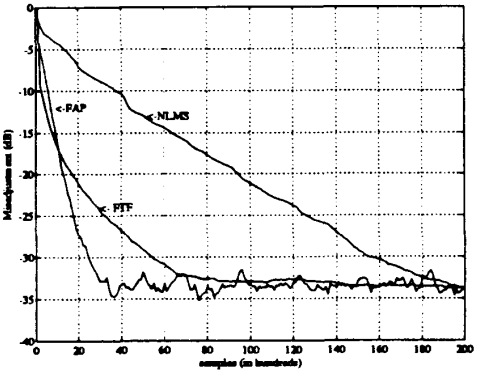

Figure 1. FAP versus FTF and NLMS with colored excitation