1-1-1996

Dynamically Regularized Fast RLS with Application to Echo Cancellation

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

DYNAMICALLY REGULARIZED FAST RLS WITH APPLICATION TO ECHO CANCELLATION

Steven L. Gay

Acoustics Research Department
AT&T Bell Laboratories
600 Mountain Avenue, Murray Hill, New Jersey 07974
email: slg@research.att.com

Abstract: This paper introduces a dynamically regularized fast recursive least squares (DR-FRLS) adaptive filtering algorithm. Numerically stabilized FRLS algorithms exhibit reliable and fast convergence with low complexity even when the excitation signal is highly self-correlated. FRLS still suffers from instability, however, when the condition number of the implicit excitation sample covariance matrix is very high. DR-FRLS, overcomes this problem with a regularization process which only increases the computational complexity by 50%. The benefits of regularization include: 1) the ability to use small forgetting factors resulting in improved tracking ability and 2) better convergence over the standard regularization technique of noise injection. Also, DR-FRLS allows the degree of regularization to be modified quickly without restarting the algorithm.

The application of DR-FRLS to stabilizing the fast affine projection (FAP) algorithm is also discussed.

1. Introduction

One of the more promising classes of adaptive filtering algorithms for acoustic echo cancellation is fast recursive least squares[1] [2] (FRLS). These exhibit reasonably low computational complexity combined with fast convergence even when the excitation signal is highly colored. Versions with improved numerical stability have appeared in the past few years[3] [4] yet, even these suffer from instability which occurs when the excitation signal's sample covariance matrix, $\mathbf{R}_n$, is poorly conditioned. This situation may arise from the excitation signal's actual statistics (it may be highly self-correlated) or from the use of an insufficiently long data window in $\mathbf{R}_n$'s estimation. Since RLS and FRLS rely on the explicit or implicit inversion (respectively) of $\mathbf{R}_n$, both become unduly susceptible to system measurement noise and/or numerical errors from finite precision arithmetic when $\mathbf{R}_n$ has some small eigenvalues.

Regularization is a common technique used in least squares methods whereby a matrix such as $\delta\mathbf{I}_N$ is added to $\mathbf{R}_n$ prior to inversion. Here, δ is a small positive number and $\mathbf{I}_N$ is the N dimensional identity matrix. This establishes δ as the lower bound for the minimum eigenvalue of the resulting matrix, stabilizing the solution (if δ is big enough) at the price of biasing the least squares solution slightly.

Here, we introduce a technique for regularizing the $\mathbf{R}_n$ inverse in such a way that the $O(N)$ complexity of the FRLS algorithms may be retained. Moreover, the degree of regularization (the size of δ) may be changed in real-time without restarting the adaptive filter, resulting in a *dynamically regularized* FRLS (DR-FRLS) adaptive filtering algorithm.

2. Regularization Refresh

RLS and FRLS are efficient ways of implementing the following algorithm,

$$e_n = d_n - \underline{x}_n^T \underline{h}_{n-1} \tag{1}$$

$$\underline{h}_n = \underline{h}_{n-1} + \mathbf{R}_n^{-1} \underline{x}_n e_n. \tag{2}$$

In acoustic echo cancellation parlance, the scalars, vectors, and matrix of (1) and (2) are defined as follows:

- d_n is the *desired* signal. It consists of both the echo and any other background acoustic signal, y_n, picked up by the microphone.
- x_n is the excitation signal and is assumed to equal 0 for $n<0$ and $\underline{x}_n$ is the N-length excitation vector,

$$\underline{x}_n = [x_n, x_{n-1}, \dots, x_{n-N+1}]^T. \tag{3}$$

- $\underline{h}_n$ is the N-length adaptive filter coefficient vector,

$$\underline{h}_n = [h_{1,n}, h_{2,n}, \dots, h_{N,n}]^T. \tag{4}$$

- e_n is the a priori error, or residual echo.
- $\mathbf{R}_n$ is the N-by-N sample covariance matrix of $\{x_n\}$.

Various windows can be applied to the data used to estimate $\mathbf{R}_n$. The exponential window is popular since it allows rank-one updating from sample period to sample period. Specifically,

$$\mathbf{R}_n = \lambda^{n+1}\delta_0 \mathbf{D}_\lambda + \sum_{i=0}^{n} \lambda^i \underline{x}_{n-i} \underline{x}_{n-i}^T \tag{5}$$

$$= \lambda \mathbf{R}_{n-1} + \underline{x}_n \underline{x}_n^T \tag{6}$$

where, λ is the *forgetting factor* selected within the range $0 << \lambda \leq 1$, δ_0 is the initial regularization,

$$\mathbf{D}_\lambda = diag\left\{\lambda^{N-1},\ \lambda^{N-2},\ \ldots,\ \lambda,\ 1\right\}, \tag{7}$$

and

$$\mathbf{R}_{-1} = \delta_0 \mathbf{D}_\lambda. \tag{8}$$

Exploitation of the rank-one update of $\mathbf{R}_n$ of equation (6) led to $O(N^2)$ complexity RLS from classical least squares methods ($O(N^3)$ complexity) and the exploitation of the shift invariant nature of $\underline{x}_n$ led to $O(N)$ complexity FRLS from RLS[1][2].

With $\mathbf{R}_n$ as defined in (5) through (8), $\delta_0 \mathbf{D}_\lambda$ serves to initially regularize the inverse in (2), but according to the first term in (5) its effect diminishes with time. Adding an appropriately scaled version of $\delta \mathbf{I}_N$ to $\mathbf{R}_n$ prior to inversion each sample period would indeed regularize the least squares solution, but that would require an additional rank N update each sample period, eliminating the computational benefit of the rank-one update in (6). An alternative, is to add an approximation to $\delta \mathbf{I}_N$, $\mathbf{D}_n$, which itself is updated each sample period with a rank-one update matrix constructed from the outer product of a shift invariant vector. Then, (2) can be modified to

$$\underline{h}_n = \underline{h}_{n-1} + \mathbf{R}_{x,n}^{-1} \underline{x}_n e_n \tag{9}$$

where

$$\mathbf{R}_{x,n} = \mathbf{D}_n + \mathbf{R}_n. \tag{10}$$

With both, $\mathbf{D}_n$ and $\mathbf{R}_n$ being maintained by rank-one updates, $\mathbf{R}_{x,n}$ requires a rank-two update. This will increase the computational complexity somewhat over those algorithms using only $\mathbf{R}_n$, but with the benefit of regularization.

It is desirable that $\mathbf{D}_n$ be constructed recursively, using the outer product of a vector composed of a shift invariant signal such that the eigenvalues of $\mathbf{D}_n$ are updated, or *refreshed* as often as possible. Accordingly, let us define the vector

$$\underline{p}_n = [0,\ 0,\ \ldots,\ 0,\ 1,\ 0,\ \ldots,\ 0]^T \tag{11}$$

where all elements in the vector are zero except for a one in position $1 + [n]_{mod\ N}$. In addition we introduce two signals ϕ_n and ξ_n which will control the size of the regularization in the adaptive filter. ϕ_n determines whether ξ_n^2 will slightly inflate (when $\phi_n = 1$) or deflate (when $\phi_n = -1$) the regularization matrix. We can now define the regularization update to be

$$\mathbf{D}_n = \sum_{i=0}^{n} \lambda^i \phi_n \xi_n^2 \underline{p}_{n-i} \underline{p}_{n-i}^T \tag{12}$$

$$= \lambda \mathbf{D}_{n-1} + \phi_n \xi_n^2 \underline{p}_n \underline{p}_n^T \tag{13}$$

with $\mathbf{D}_{-1} = 0$ as the initial condition. If we further restrict the sample periods that ϕ_n and ξ_n may change values to those where the $\underline{p}_n$ vector has its only non-zero value in the first position, $\phi_n \xi_n \underline{p}_n$ becomes shift invariant, and an $O(N)$ algorithm may then be derived.

This $\mathbf{D}_n$ is very similar to that proposed by Ljung and Soderstrom[5] with the exceptions that here we exploit 1) the shift invariance of $\phi_n \xi_n \underline{p}_n$ to get to $O(N)$ complexity and 2) the potential deflationary effects of ϕ_n enabling us to manipulate the degree of regularization more easily.

If ϕ_n and ξ_n are fixed, then the i^{th} diagonal element of $\mathbf{D}_n$ will reach a steady state of

$$d_{i,n} = \phi_n \xi_n^2 \frac{\lambda^{[n-i+1]_{mod\ N}}}{1 - \lambda^N} \tag{14}$$

Equation (14) shows that the regularization provided by the i^{th} diagonal element of $\mathbf{D}_n$ varies periodically due to the periodic nature of the regularization update. For reasonable values of λ and N the condition number of $\mathbf{D}_n$ can easily be restricted to the range of 1.1 to 1.4.

3. Dynamically Regularized FTF

In this section we define the dynamically regularized fast transversal filter (FTF), an FRLS algorithm, which also incorporates numerical stabilization. The rank-two update of $\mathbf{R}_{x,n}$ can alternately be viewed as two rank-one updates, where an intermediate *regularization refreshed* covariance matrix $\mathbf{R}_{p,n}$ is defined as,

$$\mathbf{R}_{p,n} = \lambda \mathbf{R}_{x,n-1} + \phi_n \xi_n^2 \underline{p}_n \underline{p}_n^T \tag{15}$$

and then, the *data updated covariance matrix* is

$$\mathbf{R}_{x,n} = \mathbf{R}_{p,n} + \underline{x}_n \underline{x}_n^T. \tag{16}$$

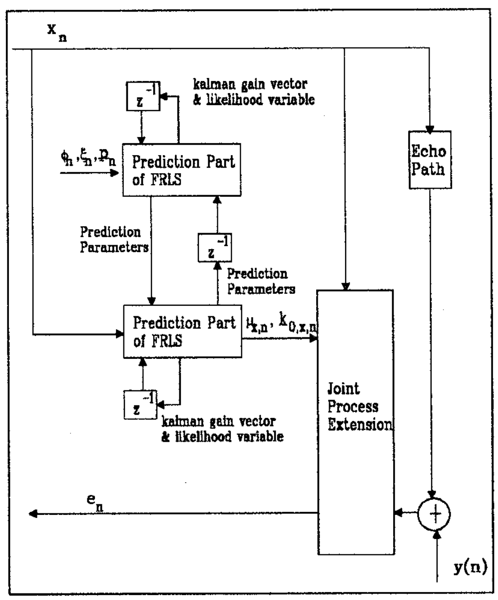

Figure 1. DR-FRLS Block Diagram

Using (16) and the matrix inversion lemma we can express the *data a posteriori kalman gain vector,* as

$$\underline{k}_{1,x,n} = \mathbf{R}_{x,n}^{-1} \underline{x}_n = \tag{18}$$
$$= (\mathbf{R}_{p,n}^{-1} - \mathbf{R}_{p,n}^{-1} \underline{x}_n (1 + \underline{x}_n^T \mathbf{R}_{p,n}^{-1} \underline{x}_n)^{-1} \underline{x}_n^T \mathbf{R}_{p,n}^{-1}) \underline{x}_n.$$

Now, define the *data a priori kalman gain vector* as

$$\underline{k}_{0,x,n} = \mathbf{R}_{p,n}^{-1}\underline{x}_n \qquad (19)$$

and the *data likelihood variable* as

$$\mu_{x,n} = (1 + \underline{k}_{0,x,n}^T \underline{x}_n)^{-1}. \qquad (20)$$

Then, after some manipulation, (18) becomes,

$$\underline{k}_{1,x,n} = \mathbf{R}_{x,n}^{-1}\underline{x}_n = \underline{k}_{0,x,n}\mu_{x,n}. \qquad (21)$$

Using (21) in (9) the coefficient update becomes,

$$\underline{h}_n = \underline{h}_{n-1} + \underline{k}_{0,x,n}\mu_{x,n}e_n. \qquad (22)$$

The remaining computations are all directed toward the stable, efficient computation of $\underline{k}_{0,x,n}$ and $\mu_{x,n}$ from sample period to sample period. These steps are summarized in the DR-FRLS block diagram of figure 1 and are detailed in tables 1 and 2.

FRLS is often separated into two parts. The prediction part generates the kalman gain vector and the likelihood variable which are sent to the joint process extension part. The joint process extension uses the output of the prediction part together with the excitation signal, x_n, and the desired signal, d_n, to generate the a priori error output, e_n. Normally, the prediction part also maintains forward and backward prediction vectors and their corresponding prediction error energies as internal variables. DR-FRLS has an additional prediction part for the regularization process. It is driven by the shift invariant sequence $\phi_n \xi_n p_n$. The prediction parts corresponding to the data and regularization processes influence each other via the prediction parameters as shown in the block diagram. Each maintains its own independent kalman gain vectors and likelihood variables. Both prediction parts use error feed-back for stabilization[4]. In addition, the likelihood variable estimates are stabilized using the multi-channel, multi-experiment method of Slock and Kailath[6].

The total complexity of DR-FRLS is 12N multiplies per sample period, only 50% more than stabilized FTF[4]. The data related prediction part requires 6N multiplies while the sparse nature of $\underline{p}_n$ allows the computation of its prediction part to be reduced to 4N multiplies. An additional 2N multiplications are required for the joint process extension.

4. A Simulation

In Figure 2 the convergence of the misalignment (the normalized coefficient error) in dB is shown for DR-FRLS and another common regularization approach called *noise injection.* In noise injection, a white noise signal is added to x_n just prior to its input into the prediction part of the FRLS adaptive filter, regularizing the sample covariance matrix. In the simulation of Figure 2 the excitation signal was a 5 second speech signal, the variance of the noise injection signal was $4.5\sigma_x^2/N$, N was 1000, $\lambda = (1 - 1/3N)$, $E_{b,x,-1} = \delta = 8.4\sigma_x^2$ and the echo-signal to background-noise ratio, SNR_{EB}, was 30 dB. The DR-FRLS simulation used the same values and in addition, $\xi_n^2 = 2.8\sigma_x^2$ and $\phi_n = 1$. The figure shows that DR-FRLS converges faster and to a lower final error level than noise injection.

5. Application to FAP

The fast affine projection (FAP) adaptive filtering algorithm[7] [8] is a low complexity, fast converging adaptive filter which is particularly useful in acoustic echo cancellation applications. If N is the length of the adaptive filter and P is the order of the affine projection algorithm, FAP's computational complexity is $2N + 20P$ multiplications per sample period. Typically, $P << N$ in acoustic echo cancellation applications. FAP uses the prediction part of a P'th order N-length sliding windowed FRLS to supply forward and backward prediction vectors and prediction error energies. The initial values of the prediction error energies provide regularization to the implicit sample covariance matrix inversion. An advantage of the sliding window is that this regularization does not diminish with time. A disadvantage though, is that the prediction parameters experience a slow build-up of numerical errors over time. One way to combat this problem is to modify the window on the FRLS algorithm to include a forgetting factor slightly less than one, producing a non-rectangular sliding window on the data. Thus, numerical errors dissipate with the forgetting factors. Unfortunately, other numerical errors arise when the forgetting factor is less than one, but these can be controlled using stabilized FRLS techniques[4]. Another side-effect is that the forgetting factor will cause the initial regularization to be forgotten together with the numerical errors. With the application of the DR-FRLS technique, though, the regularization can be refreshed each sample period. The resulting complexity is $2N + 26P$ multiplies per sample period.

6. Conclusions

This paper introduces dynamically regularized FRLS (DR-FRLS), a process for regularizing FRLS adaptive filters at the cost of a 50% increase in computational complexity. The degree of regularization can be modified easily without restarting the algorithm. Simulations indicate that DR-FRLS has better convergence performance that the noise-injection approach. Finally, the application of DR-FRLS to FAP was also discussed.

7. Acknowledgement

The author would like to thank Dr. Juergen Cezanne of AT&T Bell Laboratories for many useful discussions.

Table 1: DR-FRLS Initialization

$\underline{a}_{x,-1} = [1,0,\ldots,0]^T$	$\underline{b}_{x,-1} = [0,\ldots,0,1]^T$
$E_{b,x,-1} = \delta$	$E_{a,x,-1} = \lambda^{N-1}E_{b,x,-1}$
$\bar{\mu}_{x,-1} = 1$	$\bar{\mu}_{p,-1} = 1$
$\underline{\bar{k}}_{0,p,-1} = \underline{0}$	$\underline{\bar{k}}_{0,x,-1} = \underline{0}$
$\underline{x}_{-1} = \underline{0}$	$\underline{y}_{-1} = \underline{0}$
$\underline{p}_{-1} = \underline{0}$	
$K_1 = 1.5,\ K_2 = 2.5,\ K_4 = 0,\ K_5 = 1$	

Table 2: DR-FRLS

Prediction Part For Regularization Update	Multiplies
1) $e_{0,p,n,a} = \underline{a}^T_{x,n-1} \xi_n \underline{p}_n$	
2) $k_{0,p,n,1} = E^{-1}_{a,x,n-1} \lambda^{-1} e_{0,p,n,a}$	
3) $\underline{k}_{0,p,n} = [0, \underline{\bar{k}}^T_{0,p,n-1}]^T + \underline{a}_{x,n-1} k_{0,p,n,1}$	N
4) $\mu^{-1}_{p,n} = \bar{\mu}^{-1}_{p,n-1} + \phi_n e_{0,p,n,a} k_{0,p,n,1}$	
5) $e^f_{0,p,n,b} = \underline{b}^T_{x,n-1} \xi_n \underline{p}_n$	
6) $k^f_{0,p,n,N} = \lambda^{-1} E^{-1}_{b,x,n-1} e^f_{0,p,n,b}$	
7) extract $k^s_{0,p,n,N}$ from $\underline{k}_{0,p,n}$	
8) $e^s_{0,p,n,b} = \lambda E_{b,x,n-1} k^s_{0,p,n,N}$	
9) $e^{(j)}_{0,p,n,b} = K_j e^f_{0,p,n,b} + (1-K_j) e^s_{0,p,n,b} \quad j=1,2,5$	
10) $k_{0,p,n,N} = K_4 k^f_{0,p,n,N} + (1-K_4) k^s_{0,p,n,N}$	
11) $[\underline{\bar{k}}^T_{0,p,n}, 0]^T = \underline{k}_{0,p,n} - \underline{b}_{x,n-1} k_{0,p,n,N}$	N
12) $\bar{\mu}^{-1}_{p,n} = \mu^{-1}_{p,n} - \phi_n e^{(5)}_{0,p,n,b} k^s_{0,p,n,N}$	
13) $e_{1,p,n,a} = \bar{\mu}_{p,n-1} e_{0,p,n,a}$	
14) $\underline{a}_{p,n} = \underline{a}_{x,n-1} - \phi_n [0, \underline{\bar{k}}^T_{0,p,n-1}]^T e_{1,p,n,a}$	N
15) $E^{-1}_{a,p,n} = \lambda^{-1} E^{-1}_{a,x,n-1} - \phi_n k^2_{0,p,n,1} \mu_{p,n}$	
16) $e^{(j)}_{1,p,n,b} = \bar{\mu}_{p,n} e^{(j)}_{0,p,n,b} \quad j=1,2$	
17) $\underline{b}_{p,n} = \underline{b}_{x,n-1} - \phi_n [\underline{\bar{k}}^T_{0,p,n}, 0]^T e^{(1)}_{1,p,n,b}$	N
18) $E_{b,p,n} = \lambda E_{b,x,n-1} + \phi_n e^{(2)}_{0,p,n,b} e^{(2)}_{1,p,n,b}$	
Prediction Part For Data Update:	
19) $e_{0,x,n,a} = \underline{a}^T_{p,n} \underline{x}_n$	N
20) $k_{0,x,n,1} = E^{-1}_{a,p,n} e_{0,x,n,a}$	
21) $\underline{k}_{0,x,n} = [0, \underline{\bar{k}}^T_{0,x,n-1}]^T + \underline{a}_{p,n} k_{0,x,n,1}$	N
22) $\mu^{-1}_{x,n} = \bar{\mu}^{-1}_{x,n-1} + e_{0,x,n,a} k_{0,x,n,1}$	
23) $e^f_{0,x,n,b} = \underline{b}^T_{p,n} \underline{x}_n$	N
24) $k^f_{0,x,n,N} = E^{-1}_{b,p,n} e^f_{0,x,n,b}$	
25) extract $k^s_{0,x,n,N}$ from $\underline{k}_{0,x,n}$	
26) $e^s_{0,x,n,b} = E_{b,p,n} k^s_{0,x,n,N}$	
27) $e^{(j)}_{0,x,n,b} = K_j e^f_{0,x,n,b} + (1-K_j) e^s_{0,x,n,b} \quad j=1,2,5$	
28) $k_{0,x,n,N} = K_4 k^f_{0,x,n,N} + (1-K_4) k^s_{0,x,n,N}$	
29) $[\underline{\bar{k}}^T_{0,x,n}, 0]^T = \underline{k}_{0,x,n} - \underline{b}_{p,n} k_{0,x,n,N}$	N
30) $\bar{\mu}^{-1}_{x,n} = \mu^{-1}_{x,n} - e^{(5)}_{0,x,n,b} k^s_{0,x,n,N}$	
31) $e_{1,x,n,a} = \bar{\mu}_{x,n-1} e_{0,x,n,a}$	
32) $\underline{a}_{x,n} = \underline{a}_{p,n} - [0, \underline{\bar{k}}^T_{0,x,n-1}]^T e_{1,x,n,a}$	N
33) $E^{-1}_{a,x,n} = E^{-1}_{a,p,n} - k^2_{0,x,n,1} \mu_{x,n}$	
34) $e^{(j)}_{1,x,n,b} = \bar{\mu}_{x,n} e^{(j)}_{0,x,n,b} \quad j=1,2$	
35) $\underline{b}_{x,n} = \underline{b}_{p,n} - [\underline{\bar{k}}^T_{0,x,n}, 0]^T e^{(1)}_{1,x,n,b}$	N
36) $E_{b,x,n} = E_{b,p,n} + e^{(2)}_{0,x,n,b} e^{(2)}_{1,x,n,b}$	
Likelihood Variable Stabilization:	
37) if n is odd: $\bar{\mu}_{p,n} = \lambda^{N-1} \bar{\mu}^{-1}_{x,n} E_{b,x,n} / E_{a,x,n}$	
38) if n is even: $\bar{\mu}_{x,n} = \lambda^{N-1} \bar{\mu}^{-1}_{p,n} E_{b,x,n} / E_{a,x,n}$	
Joint Process Extension:	
39) $e_{0,n} = d_n - \underline{h}^T_n \underline{x}_n$	N
40) $\underline{h}_n = \underline{h}_{n-1} + \mu_{x,n} \underline{k}_{0,x,n} e_{0,n}$	N
Total Complexity:	12N

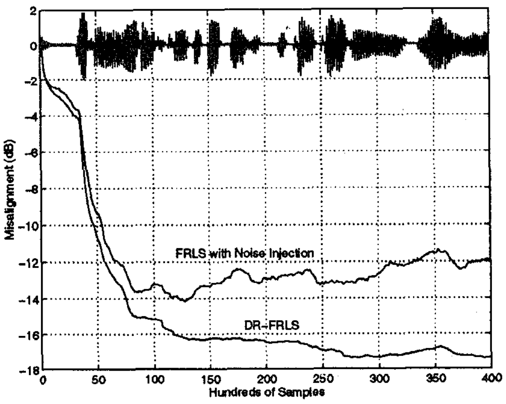

Figure 2. Convergence of DR-FRLS versus FRLS with Noise Injection

REFERENCES

1. Cioffi, J. M., and Kailath, T. "Fast Recursive Least Squares Transversal Filters for Adaptive Processing," IEEE Trans. Acoust., Speech, Signal Process., Assp-34, 304 (1984).

2. Carayannis, G., Manolakis, D., Kalouptsidis, N., "A Fast Sequential Algorithm for Least Squares Filtering and Prediction," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-31, no. 6, pp. 1394-1402, 1983.

3. Benallal, A., Gilloire, A., "A New Method to Stabilize Fast RLS Algorithms Based on a First-Order Model of the Propagation of Numerical Errors," 1988 Intl. Conf. on Acoustics, Speech, and Signal Proc., New York, N.Y., April, 1988.

4. 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.

5. Ljung, L. and Soderstrom, T., "Theory and Practice of Recursive Identification," The MIT Press, Cambridge, MA, 1983

6. Slock, D. T. M. and Kailath, T., "A Modular Multichannel Multiexperiment Fast Transversal Filter RLS Algorithm," Signal Processing, Elsevier Science Publishers, Vol. 28, pp. 25-45 (1992)

7. Gay, Steven L. and Tavathia, S., "The Fast Affine Projection Algorithm," Proceedings of Intl. Conf. on Acoustics, Speech, and Signal Proc., Detroit, MI, May 1995.

8. Tanaka, M., Kaneda, Y., Makino, S., Kojima, J., "Fast Projection Algorithm and Its Step Size Control," Proceedings of Intl. Conf. on Acoustics, Speech, and Signal Proc., Detroit, MI, May 1995.