Coordinate Descent Iterations in Fast Affine Projection Algorithm

Yuriy Zakharov, *Member, IEEE,* and Felix Albu, *Member, IEEE*

***Abstract*—We propose a new approach for real-time implementation of the fast affine projection (FAP) algorithm. This is based on exploiting the recently introduced dichotomous coordinate descent (DCD) algorithm, which is especially efficient for solving systems of linear equations on real-time hardware and software platforms since it is free of multiplication and division. The numerical stability of the DCD algorithm allows the new combined DCD-FAP algorithm also to be stable. The convergence and complexity of the DCD-FAP algorithm is compared with that of the FAP, Gauss–Seidel FAP (GS-FAP), and modified GS-FAP algorithms in the application to acoustic echo cancellation. The DCD-FAP algorithm demonstrates a performance close to that of the FAP algorithm with ideal matrix inversion and the complexity smaller than that of the Gauss–Seidel FAP algorithms.**

***Index Terms*—Coordinate descent, echo cancellation, fast affine projection, Gauss–Seidel algorithm.**

I. Introduction

THE affine projection (AP) algorithm is an efficient adaptive filtering technique [1]. It allows a higher convergence speed than the normalized least-mean squares (NLMS) algorithm, especially when the excitation signal is colored, as is the case with speech. However, it is complicated for implementation. The fast AP (FAP) algorithm allows a significant simplification [2], but it requires matrix inversion, which is a source of numerical instability, especially when implementing on real-time hardware [e.g., field-programmable gate array (FPGA)] and fixed-point software [e.g., digital signal processor (DSP)] platforms. In the original FAP algorithm, this is due to the fast recursive least-squares (RLS) algorithm. Other iterative techniques were proposed for matrix inversion in the FAP algorithm, in particular, the steepest descent and conjugate gradient techniques [3]. Gauss–Seidel (GS) iterations provide a good solution to the problem; one GS iteration per sample is enough in order to achieve nearly optimal performance [4], [5]. The FAP algorithm based on matrix inversion is computationally efficient only if the step size, which controls the convergence speed and the steady-state output (residual) error, is close to one. To reduce the residual error after the algorithm has converged, the step size should be reduced. Moreover, when the step size is close to one, the FAP algorithm is sensitive to the input noise. As a result, there is still a necessity to find an efficient numerical implementation of the FAP algorithm with an arbitrary step size.

We propose a new approach for real-time implementation of the FAP algorithm. This is based on exploiting the recently introduced dichotomous coordinate descent (DCD) algorithm [6] for solving systems of linear equations. The numerical stability of the DCD algorithm allows the new combined DCD-FAP algorithm also to be stable. The DCD algorithm is free of multiplication and division. Its complexity is low compared with the whole complexity of adaptive filtering. We compare the performance of the new algorithm with the NLMS algorithm, the FAP algorithm based on an ideal matrix inversion (ideal FAP) or GS iterations (GS-FAP), and a modified GS-FAP algorithm based on the solution of a linear system in application to acoustic echo cancellation.

II. FAP Algorithm

The FAP algorithm is defined as follows [2], [5]. Let the system output be $y_n = \mathbf{X}_n^T\mathbf{h} + w_n$, where $\mathbf{X}$ is the excitation signal matrix, $\mathbf{h}$ is an unknown impulse response, and w_n is additive white noise, and assume that for $n < 0$, we have

$$\mathbf{h}_n = \mathbf{0}, \mathbf{E}_n = \mathbf{0}, \mathbf{R}_n = \delta\mathbf{I}, \boldsymbol{\alpha}_n = \mathbf{0}, \mathbf{r}_n = [\delta, \mathbf{0}]^T. \quad (1)$$

At each sample $n \geq 0$

$$\mathbf{r}_n = \mathbf{r}_{n-1} + x_n\boldsymbol{\alpha}_n - x_{n-L}\boldsymbol{\alpha}_{n-L} \quad (2)$$

$$e_n = y_n - \mathbf{x}_n^T\mathbf{h}_{n-1} - \mu\tilde{\mathbf{r}}_n^T\bar{\mathbf{E}}_{n-1} \quad (3)$$

$$\text{update } \mathbf{R}_n \text{ using } \mathbf{r}_n \quad (4)$$

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

$$\boldsymbol{\epsilon}_n = \mathbf{P}_n\mathbf{e}_n \quad \text{or solve} \quad \mathbf{R}_n\boldsymbol{\epsilon}_n = \mathbf{e}_n \quad (6)$$

$$\mathbf{E}_n = \begin{bmatrix} 0 \\ \bar{\mathbf{E}}_{n-1} \end{bmatrix} + \boldsymbol{\epsilon}_n \quad (7)$$

$$\mathbf{h}_n = \mathbf{h}_{n-1} + \mu\mathbf{x}_{n-N+1}E_{N-1,n} \quad (8)$$

where n is the time index, $\mathbf{h}_n = [h_{0,n}, \ldots, h_{L-1,n}]^T$ is an adaptive weight vector of a length L, μ is the step size, and $(\cdot)^T$ denotes the matrix transpose. The $L \times N$ excitation signal matrix $\mathbf{X}_n$ has the structure $\mathbf{X}_n = [\mathbf{x}_n, \mathbf{x}_{n-1}, \ldots, \mathbf{x}_{n-N+1}]$, where $\mathbf{x}_n = [x_n, \ldots, x_{n-L+1}]^T$, $\mathbf{P}_n = \mathbf{R}_n^{-1}$ is the inverse of a $N \times N$ regularized autocorrelation matrix of the excitation signal $\mathbf{R}_n = \mathbf{X}_n^T\mathbf{X}_n + \delta\mathbf{I}$, δ is the regularization parameter, $\mathbf{I}$ is the $N \times N$ identity matrix, $\mathbf{E}_n$ is an $N \times 1$ vector consisting of a sum of the fast normalized residual echo $\boldsymbol{\epsilon}_n$, $E_{N-1,n}$ is the

Manuscript received June 29, 2004; revised November 11, 2004. This work was supported in part by the European Commission under CAPANINA Project FP6-IST-2003-506745. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Zhi Ding.

Y. Zakharov is with the University of York, York, YO10 5DD U.K.

F. Albu is with the Politehnica University of Bucharest, Bucharest, 060032 Romania.

Digital Object Identifier 10.1109/LSP.2005.843765

last element of $\mathbf{E}_n, \bar{\mathbf{E}}_n$ is a vector consisting of the uppermost $N-1$ elements of $\mathbf{E}_n, \bar{\mathbf{e}}_n$ consists of $N-1$ upper elements of $\mathbf{e}_n, \boldsymbol{\alpha}_n = [x_n, \ldots, x_{n-N+1}]^T$, and the vector $\tilde{\mathbf{r}}_n$ contains $N-1$ lower elements of $\mathbf{r}_n$. In step (4), $\mathbf{R}_n$ is updated by replacing the first row and column with elements of $\mathbf{r}_n$, while the bottom-right $(N-1)\times(N-1)$ submatrix is replaced with the top-left $(N-1)\times(N-1)$ submatrix of $\mathbf{R}_{n-1}$.

The complexity of FAP algorithms is $2L+f(N)$ multiply–accumulate (MAC) operations per sample. The term $2L$ is for steps (3) and (8), while the term $f(N)$, which does not depend on L, is for the other steps, among which step (6) is computationally most demanding. In an ideal FAP algorithm, with a direct matrix inversion, $f(N)=\mathcal{O}(N^3)$. The fast RLS algorithm allows reduction in the complexity of the matrix inversion, resulting in $f(N)=20N$; however, it shows numerical instability [2].

If $\mu \approx 1$, then $\mathbf{e}_n \approx e_n \mathbf{b}$, where $\mathbf{b}=[1,0,\ldots,0]^T$, and the first column $\mathbf{p}_n$ of the matrix $\mathbf{P}_n$ is only required for calculating $\boldsymbol{\epsilon}_n$ in (6). Since $L \gg N$, the matrix $\mathbf{R}_n$ is slowly varying in time, as is the solution $\mathbf{p}_n$ of the system. Assuming that we have already obtained an accurate estimate of the vector $\mathbf{p}_{n-1}$ for sample $(n-1)$, one GS iteration per sample is enough for nearly optimal performance [4], [5]. The GS-FAP algorithm is based on one update of $\mathbf{p}_n$ at every sample. This is equivalent to solving the system $\mathbf{R}_n\mathbf{p}_n=\mathbf{b}$ with one GS iteration

$$p_{n,i}=\frac{1}{(\mathbf{R}_n)_{ii}}\left(b_i-\sum_{j=0}^{i-1}(\mathbf{R}_n)_{ij}p_{(n-1),j}-\sum_{j=i+1}^{N-1}(\mathbf{R}_n)_{ij}p_{n,j}\right)$$

where b_i is the ith element of the vector $\mathbf{b}$, $p_{n,i}$ is ith element of $\mathbf{p}_n$, $(\mathbf{R}_n)_{ij}$ is the (i,j)th element of $\mathbf{R}_n$, and $i=0,\ldots,N-1$. Thus, the FAP algorithm based on matrix inversion benefits from distribution of the calculation in time. However, for an arbitrary μ, the matrix inverse will require the solution of N systems of N equations [3]. This is computationally expensive, even with such a distribution of the calculation.

For an arbitrary μ, the solution of the system of equations in (6) might be preferable to matrix inversion. Unfortunately, calculations associated with the solution of the system cannot be distributed in time; a relatively accurate solution should be found at every sample period. The DCD algorithm allows the solution to be obtained without any multiplications and divisions; instead, it uses shift-accumulation (SAC) operations; the latter makes it attractive when a real-time implementation is required.

III. DCD Algorithm

The system of equations to be solved is $\mathbf{R}\boldsymbol{\epsilon}=\mathbf{e}$, where, for clarity, we have omitted the lower index associated with the sample period. For the solution of the system, the DCD algorithm [6] is used. The algorithm is based on binary representation of elements of the solution vector with M_b bits within an amplitude range $[-H,H]$. It starts an iterative approximation of the solution vector $\boldsymbol{\epsilon}$ from the most significant bit. Once the most significant bit has been found for all vector elements, the algorithm starts updating the next less significant bit, and so on. If a bit update happens (such an iteration is called "successful"), the vector $\mathbf{e}$ is also updated. The complexity of the algorithm is mainly due to "successful" iterations. To limit the complexity (with an uncertain error of the solution), a limit for the number of "successful" iterations N_{upd} is predefined. If there is no such limit or the parameter N_{upd} is high enough, the accuracy of the solution is $2^{-M_b}H$. Thus, parameters of the DCD algorithm are as follows: M_b—number of bits used for representation of elements of the vector $\boldsymbol{\epsilon}$ within an amplitude range $[-H,H]$ and N_{upd}—the maximum number of "successful" iterations, at which the solution vector is updated. Denote ϵ_p and e_p elements of vectors $\boldsymbol{\epsilon}$ and $\mathbf{e}$, respectively. The DCD algorithm can be implemented as follows.

Initialization: $\boldsymbol{\epsilon}=0, d=H, q=0$.
for $m=1:M_b$
  $d=d/2$
(1) Flag $=0$
  for $p=0:N-1$
    if $|e_p|>(d/2)[\mathbf{R}]_{pp}$, then
      Flag $=1, q=q+1$
      $\varepsilon_p=\varepsilon_p+\mathrm{sgn}(e_p)\cdot d$
      $\mathbf{e}=\mathbf{e}-\mathrm{sgn}(e_p)\cdot d\cdot\mathbf{R}(:,p)$
    if $q>N_{\mathrm{upd}}$, then the algorithm stops
  end of the p-loop
  if Flag $=1$, then go to (1)
end of the m-loop

The DCD algorithm guarantees convergence to the true solution if elements of the true solution vector $\boldsymbol{\epsilon}$ are within the interval $[-H,H]$. It is seen from the algorithm description that if H is a power of two, then multiplications by factors of power of two are only used; these can be replaced by bit shifts. Thus, the DCD algorithm can be implemented without explicit multiplications and divisions, which are well known to require a significantly higher chip area and power consumption in hardware implementation than addition and bit-shift operations. Moreover, divisions are often the source of numerical instability. The complexity of the DCD algorithm for a particular system of equations depends on many factors. However, for given N_{upd} and M_b, the peak (worst-case scenario) complexity can be shown to be

$$N(2N_{\mathrm{upd}}+M_b)\ \text{SACs}. \tag{9}$$

IV. Numerical Results

Now, we consider acoustic echo cancellation in the following scenario. The room acoustic impulse response has a length $L=512$. The excitation signal is speech sampled at the frequency 8 kHz with a 16-bit resolution. In all FAP algorithms, the affine projection order is $N=8$, and the step size is $\mu=1/8$. The parameters of the DCD algorithm are set to $H=10^{-5}$ and $M_b=16$, while the number of updates N_{upd}, which controls the algorithm complexity and solution accuracy, is varying.

Fig. 1 shows the simulation results for scenarios with noise 30 dB down from the echo, i.e., the signal-to-noise ratio $(\mathrm{SNR})=30$ dB. Experimental plots have been obtained by averaging the misalignment $\|\mathbf{h}-\mathbf{h}_n\|^2/\|\mathbf{h}\|^2$ in 100 trials. In each trial, new excitation speech and noise signals are used. It is seen that even with one "successful" update

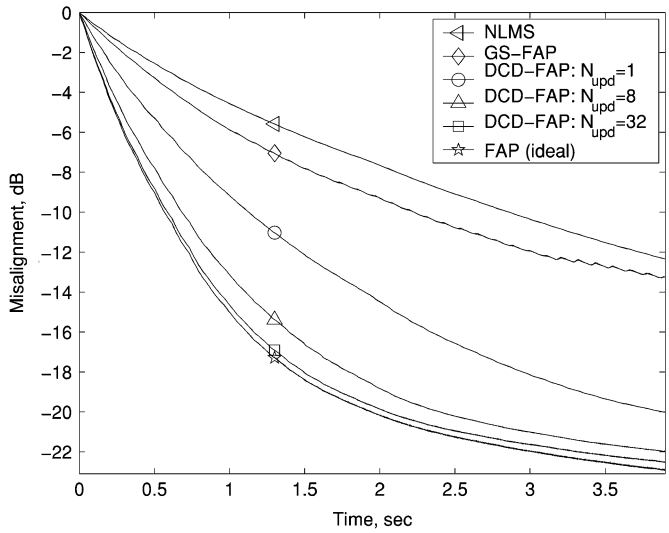

Fig. 1. Comparison of the misalignment (dB) for the ideal FAP, GS-FAP, and DCD-FAP algorithms. $N = 8, \mu = 1/8$, SNR $= 30$ dB.

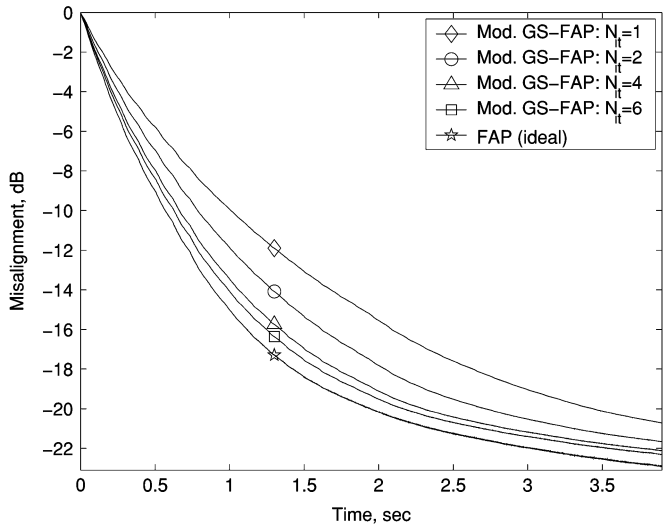

Fig. 2. Comparison of the misalignment (dB) for the ideal FAP and modified GS-FAP algorithms. $N = 8, \mu = 1/8$, SNR $= 30$ dB.

$(N_{\rm upd} = 1)$, the DCD-FAP algorithm provides a significantly better performance than the NLMS and GS-FAP algorithms. The performance of the DCD-FAP algorithm improves as $N_{\rm upd}$ increases, and for $N_{\rm upd} = 32$, it is close to the performance of the ideal FAP algorithm. Simulations for SNR $= 20$ dB (not shown here) have demonstrated that $N_{\rm upd} = 8$ "successful" updates provide nearly the same performance as the ideal FAP algorithm.

Note that GS iterations can also be used to solve the system of equations in (6); we call such a combined algorithm the modified GS-FAP algorithm. In this case, the number of iterations $N_{\rm it}$ may need to be more than one. Fig. 2 shows how the performance of the modified GS-FAP algorithm is improved with the number of GS iterations and approaches that of the ideal FAP algorithm in the same scenario as in Fig. 1. It is seen that the modified GS-FAP algorithm with one iteration provides a better performance than the GS-FAP algorithm and approximately the same as the DCD-FAP algorithm with one "successful" update (see Fig. 1). For $N_{\rm it} \geq 4$, we obtain a performance close to that of the ideal FAP algorithm.

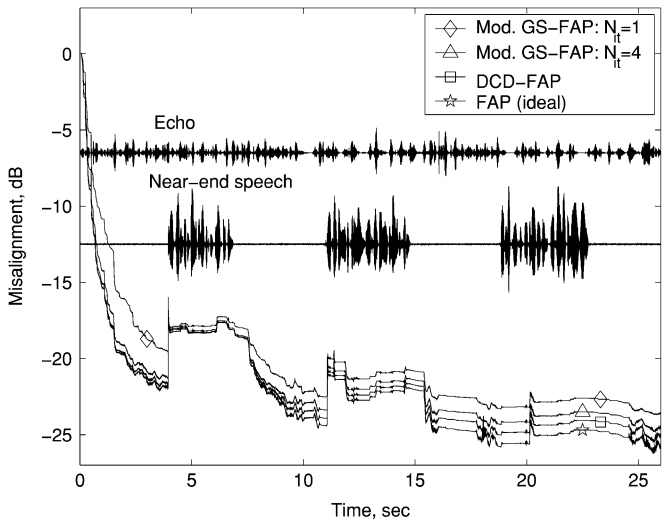

Fig. 3. Comparison of the misalignment (dB) for the ideal FAP, DCD-FAP $(N_{\rm upd} = 32)$, and modified GS-FAP ($N_{\rm it} = 1$ and $N_{\rm it} = 4$) algorithms with time-varying regularization in a double-talk scenario. $N = 8, \mu = 1/8$, SNR $= 30$ dB.

In noisy conditions, especially in double-talk scenarios, regularization is a necessary part of adaptive algorithms. The adaptation should be slowed down in the presence of intensive near-end signals. Fig. 3 compares the misalignment for the ideal FAP, DCD-FAP ($N_{\rm upd} = 32$) and modified GS-FAP ($N_{\rm it} = 1$ and $N_{\rm it} = 4$) algorithms in a double-talk scenario. In this scenario, the echo power is 11 dB below the near-end speech and SNR $= 30$ dB. The regularization parameter δ varies according to a simple technique based on approaches described in [5] and [7]: $\delta = \max(\delta_{\min}, \delta_n)$, where $\delta_{\min} = 10^8, \delta_n = \rho_x(n)$ if $\rho_x(n) > \gamma\rho_y(n)$ or $\delta_n = 20L\rho_y(n)$ otherwise, $\gamma = 8$, and $\rho_x(n)$ and $\rho_y(n)$ are time-averaged powers of signals x_n and y_n, respectively. For averaging, attack-release filters were used with instantaneous attack and a slow-release time constant of 1 s, as in [5]. It is seen that the ideal FAP and DCD-FAP algorithms demonstrate performance with the misalignment difference being less than 1 dB. The modified GS-FAP algorithm with $N_{\rm it} = 1$ shows poorer performance, especially at the initial stage of the adaptation. For $N_{\rm it} = 4$, its performance approaches that of the DCD-FAP algorithm with $N_{\rm upd} = 32$. Note that with $N_{\rm upd} = 8$, the performance of the DCD-FAP algorithm (not shown here) is the same as that of the modified GS-FAP algorithm with $N_{\rm it} = 4$. The overall echo attenuation (ERLE) over the time interval between the 2nd and 26th seconds is approximately 17 dB for all three algorithms.

Fig. 4 shows results for the same scenario as in Fig. 3, with a lower SNR of 20 dB. In this case, the ideal FAP, DCD-FAP $(N_{\rm upd} = 32)$, and modified GS-FAP $(N_{\rm it} = 4)$ algorithms demonstrate approximately equal performance. Note that the same performance is provided by the DCD-FAP algorithm with $N_{\rm upd} = 8$ "successful" updates (not shown here). The modified GS-FAP algorithm with one iteration $N_{\rm it} = 1$ shows a slower convergence at the initial stage of the adaptation.

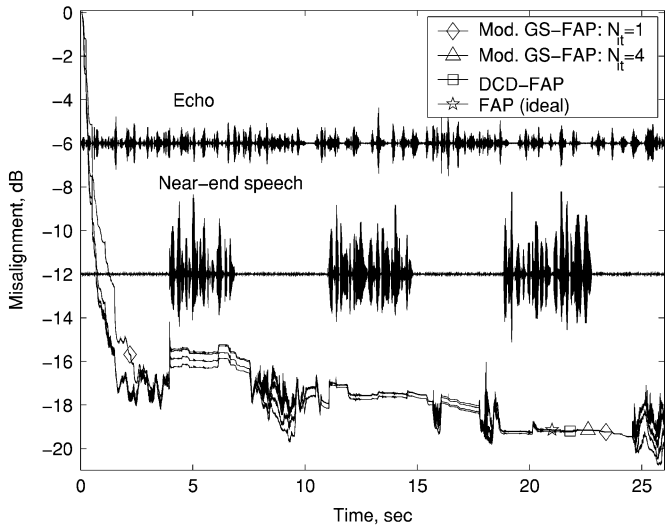

Fig. 4. Comparison of the misalignment (dB) for the ideal FAP, DCD-FAP ($N_{\rm upd} = 32$), and modified GS-FAP ($N_{\rm it} = 1$ and $N_{\rm it} = 4$) algorithms with time-varying regularization in a double-talk scenario. $N = 8, \mu = 1/8$, SNR $= 20$ dB.

In all trials, the complexity of the DCD algorithm for $N_{\rm upd} = 1, 8$, and 32 has not exceeded 128, 248, and 568 SACs, respectively. These are close to the peak DCD complexity from (9): 144, 256, and 592 SACs, respectively. The average complexity has been smaller: 75, 166, and 452 SACs, respectively. Simulations with smaller M_b down to $M_b = 8$ (not shown here) produced a performance close to that for $M_b = 16$. As the peak DCD complexity depends mainly on $N_{\rm upd}$, its reduction, in the case of $M_b = 8$, has not been significant (88, 200, and 472 SACs, respectively), but the average complexity has been reduced greatly (67, 114, and 135 SACs, respectively).

For step (6), the modified GS-FAP algorithm requires approximately $N_{\rm it}N^2$ MACs and $N_{\rm it}N$ divisions; for $N_{\rm it} = 4$ and $N = 8$, this results in 256 MACs and 32 divisions. For the same performance, the DCD-FAP algorithm requires a maximum of 256 SACs and no multiplication or division. Note that, in our example, steps (3) and (8) of the adaptive filtering require approximately $2L = 1024$ MACs per sample, which is significantly greater than the DCD peak complexity. MAC and division operations required for GS iterations are well known to be more complicated for hardware implementation than addition and bit-shift operations required for DCD iterations. Moreover, division operations are a source of numerical instability, and it is preferable to avoid them in real-time systems.

V. Conclusion

We have proposed a new approach for real-time implementation of the fast affine projection algorithm. This is based on the application of the dichotomous coordinate descent iterations for solving systems of linear equations in the FAP algorithm. As the DCD algorithm does not require explicit multiplications and divisions, it is well suited for implementation on real-time hardware and fixed-point software platforms, providing a computationally stable DCD-FAP algorithm. We have compared the performance of the new algorithm with the ideal FAP and Gauss–Seidel FAP algorithms in application to acoustic echo cancellation. It has been shown that the DCD-FAP algorithm demonstrates performance close to that of the ideal FAP algorithm and the complexity smaller than that of the Gauss–Seidel FAP algorithms.

References

[1] A. H. Sayed, *Fundamentals of Adaptive Filtering*. New York: Wiley, 2003.
[2] S. L. Gay and S. Tavathia, "The fast affine projection algorithm," in *Proc. ICASSP*, 1995, pp. 3023–3026.
[3] H. Ding, "Stable Adaptive Filter and Method," Int. Patent Application WO 0038319, Jun. 29, 2000.
[4] F. Albu, J. Kadlec, N. Coleman, and A. Fagan, "The Gauss-Seidel fast affine projection algorithm," in *Proc. SIPS*, San Diego, CA, Oct. 2002, pp. 109–114.
[5] E. Chau, H. Sheikhzadeh, and R. L. Brennan, "Complexity reduction and regularization of a fast affine projection algorithm for oversampled subband adaptive filters," in *Proc. ICASSP*, vol. 5, Montreal, QC, Canada, May 2004, pp. 109–112.
[6] Y. V. Zakharov and T. C. Tozer, "Multiplication-free iterative algorithm for LS problem," *Electron. Lett.*, vol. 40, no. 9, pp. 567–569, Apr. 2004.
[7] H. Sheikhzadeh, R. L. Brennan, and K. R. L. Whyte, "Near-end distortion in over-sampled subband adaptive implementation of affine projection algorithm," in *Proc. EUSIPCO*, Vienna, Austria, Sep. 2004, pp. 413–416.