A FAST GAUSS-NEWTON PARALLEL-CASCADE ADAPTIVE TRUNCATED VOLTERRA FILTER

Thomas M. Panicker and V. John Mathews
Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112.

ABSTRACT

This paper introduces a computationally efficient Gauss-Newton type adaptation algorithm for parallel-cascade realizations of truncated Volterra systems with arbitrary, but finite order nonlinearity. Parallel-cascade realizations implement higher-order Volterra systems using parallel and multiplicative combinations of lower-order Volterra systems. The complexity of our system is comparable to the complexity of the system model itself, and is considerably less than that of the fast RLS Volterra filters. Results of experiments comparing the Gauss-Newton method with a competing structure with similar computational complexity as well as demonstrating the capability of parallel-cascade systems to approximate truncated Volterra systems are also included in the paper.

1. INTRODUCTION

The output of a homogeneous and causal pth order Volterra system with N-sample memory is related to its input as [1]

$$y(n) = \sum_{m_p=m_{p-1}}^{N-1} \cdots \sum_{m_1=0}^{N-1} h_p(m_1,\cdots,m_p) \times x(n-m_1)\cdots x(n-m_p). \quad (1)$$

In the above equation, we have explicitly made use of the invariance of the coefficients with respect to permutations of their indices. It has been shown in [2] that any system with input-output relationship as in (1) can be represented exactly using a parallel combination of components as shown in Figure 1. The advantages of realizing higher-order Volterra systems using parallel-cascade structures are also discussed in [2].

2. PARALLEL-CASCADE REALIZATION

The output of the parallel-cascade system of Figure 1 can be expressed as

$$y(n) = \sum_{i=1}^{r} \sigma_i \left[\mathbf{X}_{N,l}^T(n)\mathbf{U}_i\right]\left[\mathbf{X}_{N,p-l}^T(n)\mathbf{V}_i\right] = \sum_{i=1}^{r} \sigma_i y_{l,i}(n) y_{p-l,i}(n), \quad (2)$$

where $l < p$ and the vector $\mathbf{X}_{N,p_1}(n)$ has $\begin{pmatrix} N+p_1-1 \\ p_1 \end{pmatrix}$ elements and contains all possible p_1th order product signals of the form $x(n-k_1)x(n-k_2)\cdots x(n-k_{p_1})$, and

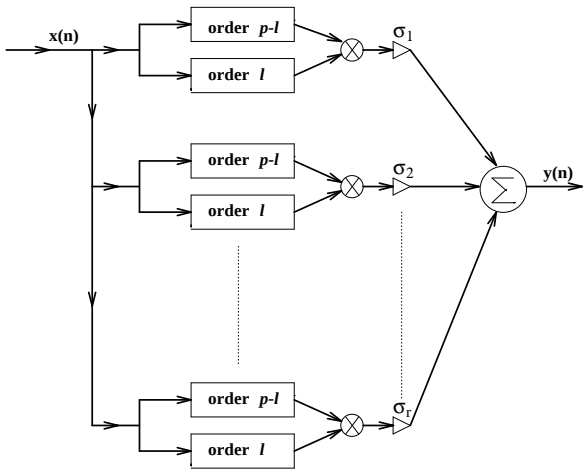

Figure 1. A parallel-cascade realization of a pth order Volterra kernel. Each block represents a Volterra system of the order shown within.

$0 \le k_1 \le k_2 \le \cdots \le k_{p_1} \le N-1$. The vectors $\mathbf{U}_i$ and $\mathbf{V}_i$ are coefficient vectors of appropriate dimensions. The signal $y_{l,i}(n)$ in (2) is defined as the output of a homogeneous lth order Volterra system given by

$$y_{l,i}(n) = \mathbf{X}_{N,l}^T(n)\mathbf{U}_i. \quad (3)$$

The signal $y_{p-l,i}(n)$ is also defined in a similar manner.

The objective of this paper is to derive a computationally efficient Gauss-Newton type adaptation algorithm for truncated Volterra filters when the system model is realized using a parallel-cascade structure. The computational complexity of our technique is comparable to the complexity of the system model itself. Experimental results presented later in this paper show that the fast algorithm exhibit fast convergence characteristics when compared with the LMS adaptive Volterra filters. Realizations of the adaptive filter that employs fewer than the maximum necessary number of parallel branches often perform similarly to more complex algorithms with larger number of branches.

3. GAUSS-NEWTON RECURSION FOR PARALLEL-CASCADE VOLTERRA SYSTEMS

Consider the problem of estimating a desired response signal $d(n)$ as the output of a homogeneous pth order adaptive Volterra system employing the parallel-cascade structure with r branches. The output of the adaptive filter may

be written as

$$\begin{aligned}\hat{d}(n) &= \sum_{i=1}^{r} \sigma_i(n) y_{l,i}(n) y_{p-l,i}(n) \\ &= \sum_{i=1}^{r} \sigma_i(n) \left[\mathbf{X}_{N,l}^T(n)\mathbf{U}_i(n)\right] \\ &\quad \times \left[\mathbf{X}_{N,p-l}^T(n)\mathbf{V}_i(n)\right]. \end{aligned} \tag{4}$$

Let $e(n) = d(n) - \hat{d}(n)$ denote the estimation error at time n. Our objective is to develop a Gauss-Newton adaptive filter that attempts to reduce

$$\xi(n) = \frac{1}{2} E\left\{ \mid e(n) \mid^2 \right\} \tag{5}$$

at each iteration. Taking the partial derivative of $\xi(n)$ with respect to $\mathbf{U}_m(n)$, we get

$$\begin{aligned}\frac{\partial \xi(n)}{\partial \mathbf{U}_m(n)} &= -E\left\{ e(n)\sigma_m(n)\mathbf{X}_{N,l}(n)\mathbf{X}_{N,p-l}^T(n)\mathbf{V}_m(n) \right\} \\ &= -E\left\{ e(n)\sigma_m(n) y_{m,p-l}(n)\mathbf{X}_{N,l}(n) \right\}. \end{aligned} \tag{6}$$

The estimation problem may now be considered as one in which we attempt to estimate $d(n)$ using $\sigma_m(n) y_{m,p-l}(n)\mathbf{X}_{N,l}(n)$. A Gauss-Newton adaptation algorithm [3] may be derived for this case as

$$\mathbf{U}_m(n+1) = \mathbf{U}_m(n) + \mu \mathbf{R}_{u,m}^{-1}(n)\mathbf{X}_{m,N,l}(n)e(n), \tag{7}$$

where μ is a convergence parameter, $\mathbf{X}_{m,N,l}(n)$ is defined as

$$\mathbf{X}_{m,N,l}(n) = \sigma_m(n) y_{m,p-l}(n)\mathbf{X}_{N,l}(n), \tag{8}$$

and $\mathbf{R}_{u,m}(n)$ is an estimate of $E\left\{ \mathbf{X}_{m,N,l}(n)\mathbf{X}_{m,N,l}^T(n) \right\}$ obtained in our system as

$$\mathbf{R}_{u,m}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{X}_{m,N,l}(i)\mathbf{X}_{m,N,l}^T(i). \tag{9}$$

Let us define a gain vector $\mathbf{k}_{u,m}(n)$ for each branch as

$$\mathbf{k}_{u,m}(n) = \mathbf{R}_{u,m}^{-1}(n)\mathbf{X}_{m,N,l}(n). \tag{10}$$

Substituting (10) in (7), the update equation becomes

$$\mathbf{U}_m(n+1) = \mathbf{U}_m(n) + \mu \mathbf{k}_{u,m}(n)e(n). \tag{11}$$

The gain vector $\mathbf{k}_{u,m}(n)$ can be computed using the matrix inversion lemma as is typically done in the derivation of conventional RLS algorithms [4]. It is straightforward to develop similar update equations for $\mathbf{V}_m(n)$ and $\sigma_m(n)$. The update equation for $\mathbf{V}_m(n)$ is given by

$$\mathbf{V}_m(n+1) = \mathbf{V}_m(n) + \mu \mathbf{k}_{v,m}(n)e(n), \tag{12}$$

where

$$k_{v,m}(n) = \mathbf{R}_{v,m}^{-1}(n)\mathbf{X}_{m,N,p-l}(n). \tag{13}$$

In (13), $\mathbf{X}_{m,N,p-l}(n)$ is given by

$$\mathbf{X}_{m,N,p-l}(n) = \sigma_m(n) y_{m,l}(n)\mathbf{X}_{N,p-l}(n). \tag{14}$$

Table 1. Gauss-Newton algorithm for parallel-cascade Volterra filter.

Initialization		
for $m = 1 : r$,		
$\mathbf{P}_{u,m}(0)$	=	$\delta^{-1}\mathbf{I}$
$\mathbf{P}_{v,m}(0)$	=	$\delta^{-1}\mathbf{I}$
$P_{\sigma,m}(0)$	=	δ^{-1}
$u_{i,m}(0)$	=	$\begin{cases} 1 & ;\text{if } i = m \\ 0 & ;\text{otherwise} \end{cases}$
$v_{i,m}(0)$	=	$\begin{cases} 1 & ;\text{if } i = m \\ 0 & ;\text{otherwise} \end{cases}$
$\sigma_m(0)$	=	1
Main Routine		
for $m = 1 : r$,		
$\mathbf{X}_{m,N,l}(n)$	=	$\sigma_m(n)\left[\mathbf{X}_{N,p-l}^T(n)\mathbf{V}_m(n)\right]\mathbf{X}_{N,l}(n)$
$\pi_{u,m}(n)$	=	$\mathbf{X}_{m,N,l}^T(n)\mathbf{P}_{u,m}(n-1)$
$\kappa_{u,m}(n)$	=	$\lambda + \pi_{u,m}(n)\mathbf{X}_{m,N,l}(n)$
$\mathbf{k}_{u,m}(n)$	=	$\frac{\mathbf{P}_{u,m}(n-1)\mathbf{X}_{m,N,l}(n)}{\kappa_{u,m}(n)}$
$\mathbf{P}'_{u,m}(n-1)$	=	$\mathbf{k}_{u,m}\pi_{u,m}(n)$
$\mathbf{P}_{u,m}(n)$	=	$\frac{1}{\lambda}\left(\mathbf{P}_{u,m}(n-1) - \mathbf{P}'_{u,m}(n-1)\right)$
$\mathbf{P}_{u,m}(n)$	=	$\frac{\mathbf{P}_{u,m}(n) + \mathbf{P}_{u,m}^T(n)}{2}$
Similarly update		$\{\mathbf{k}_{v,m}(n)$ and $k_{\sigma,m}(n); m = 1, 2, \cdots, r\}$
$e(n)$	=	$d(n) - \sum_{m=1}^{r} \sigma_m(n)\left[\mathbf{X}_{N,l}^T(n)\mathbf{U}_m(n)\right]$
		$\times \left[\mathbf{X}_{N,p-l}^T(n)\mathbf{V}_m(n)\right]$
for $m = 1 : r$,		
$\mathbf{U}_m(n+1)$	=	$\mathbf{U}_m(n) + \mu\mathbf{k}_{u,m}(n)e(n)$
$\mathbf{V}_m(n+1)$	=	$\mathbf{V}_m(n) + \mu\mathbf{k}_{v,m}(n)e(n)$
$\sigma_m(n+1)$	=	$\sigma_m(n) + \mu k_{\sigma,m}(n)e(n)$
δ	=	a small positive constant
$\mathbf{I}$	=	Identity Matrix

and $\mathbf{R}_{v,m}(n)$ is defined as

$$\mathbf{R}_{v,m}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{X}_{m,N,p-l}(i)\mathbf{X}_{m,N,p-l}^T(i). \tag{15}$$

Similarly, the update equation for $\sigma_m(n)$ is given by

$$\sigma_m(n+1) = \sigma_m(n) + \mu k_{\sigma,m}(n)e(n), \tag{16}$$

where

$$k_{\sigma,m}(n) = R_{\sigma,m}^{-1}(n)X_{\sigma,m}(n) \tag{17}$$

and $X_{\sigma,m}(n)$ and $R_{\sigma,m}(n)$ in the above equation are given by

$$X_{\sigma,m}(n) = y_{m,l}(n)y_{m,p-l}(n) \tag{18}$$

and

$$R_{\sigma,m}(n) = \sum_{i=1}^{n} \lambda^{n-i} X_{\sigma,m}^2(i), \tag{19}$$

respectively. In all the experiments described later we chose $\mu = (1 - \lambda)$, such that the tracking of the autocorrelation matrix as well as the adaptation of the coefficients have comparable speeds.

Table 1 describes the Gauss-Newton algorithm for updating the coefficient vectors. In Table 1, $u_{i,j}(n)$ and $v_{i,j}(n)$ represent the jth element of the coefficient vector $\mathbf{U}_i(n)$

and $\mathbf{V}_i(n)$, respectively. Similarly, the matrices $\mathbf{P}_{u,m}(n)$ and $\mathbf{P}_{v,m}(n)$ represent the matrices $\mathbf{R}^{-1}_{u,m}(n)$ and $\mathbf{R}^{-1}_{v,m}(n)$, respectively. The scalar $P_{\sigma,m}(n)$ represent $R^{-1}_{\sigma,m}(n)$.

4. A FAST GAUSS-NEWTON RECURSION FOR PARALLEL-CASCADE VOLTERRA SYSTEMS

The Gauss-Newton algorithm described above is computationally demanding since the gain vector has to be evaluated for each branch. In this section, we derive a fast Gauss-Newton method for our problem by making certain judicious approximations. Specifically, the approximations are such that only one gain vector need to be calculated for updating all $\mathbf{U}_m(n)$'s. Similarly, $\mathbf{V}_m(n)$'s can be updated using a second gain vector.

Let us rewrite the estimation error $e(n)$ as

$$e(n) = d_m(n) - \hat{d}_m(n), \tag{20}$$

where $d_m(n)$ and $\hat{d}_m(n)$ are given by

$$d_m(n) = d(n) - \sum_{k=1,k\neq m}^{r} \sigma_k(n) y_{l,k}(n) y_{p-l,k}(n) \tag{21}$$

and

$$\hat{d}_m(n) = \sigma_m(n) y_{l,m}(n) y_{p-l,m}(n), \tag{22}$$

respectively. The estimation error in (20) can be written as

$$e(n) = \alpha_{u,m}(n)[\frac{d_m(n)}{\alpha_{u,m}(n)} - \mathbf{U}^T_m(n)\mathbf{X}_{N,l}(n)], \tag{23}$$

where

$$\begin{aligned} \alpha_{u,m}(n) &= \sigma_m(n) y_{p-l,m}(n) \\ &= \sigma_m(n)[\mathbf{X}^T_{N,p-l}(n)\mathbf{V}_m(n)]. \end{aligned} \tag{24}$$

The cost function in (5) can be written using the above definitions as

$$\xi(n) = \frac{1}{2} E\left\{ \alpha^2_{u,m}(n) \left| \frac{d_m(n)}{\alpha_{u,m}(n)} - \mathbf{U}^T_m(n)\mathbf{X}_{N,l}(n) \right|^2 \right\}. \tag{25}$$

The estimation problem can now be viewed as one in which we attempt to estimate $d_m(n)/\alpha_{u,m}(n)$ with the data vector $\mathbf{X}_{N,l}(n)$. The update equation for $\mathbf{U}_m(n)$ can therefore be written as

$$\mathbf{U}_m(n+1) = \mathbf{U}_m(n) + \mu \mathbf{R}^{-1}_l(n)\mathbf{X}_{N,l}(n)\alpha^2_{u,m}(n)e'_{u,m}(n), \tag{26}$$

where $e'_{u,m}(n)$ is given by

$$e'_{u,m}(n) = \frac{d_m(n)}{\alpha_{u,m}(n)} - \mathbf{U}^T_m(n)\mathbf{X}_{N,l}(n) \tag{27}$$

and matrix $\mathbf{R}_l(n)$ is given by

$$\mathbf{R}_l(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{X}_{N,l}(i)\mathbf{X}^T_{N,l}(i). \tag{28}$$

It is straightforward to see that $e'_{u,m}(n)$ and $e(n)$ are related as

$$\alpha^2_{u,m}(n)e'_{u,m}(n) = \alpha_{u,m}(n)e(n). \tag{29}$$

We can write the update equation for $\mathbf{U}_m(n)$ using (29) as

$$\mathbf{U}_m(n+1) = \mathbf{U}_m(n) + \mu\alpha_{u,m}(n)\mathbf{k}_u(n)e(n), \tag{30}$$

where

$$\mathbf{k}_u(n) = \mathbf{R}^{-1}_l(n)\mathbf{X}_{N,l}(n). \tag{31}$$

It is important to recognize that the recursions obtained above is only an approximation to the system derived in the previous section. This is because the weighting factor $\alpha^2_{u,m}(n)$ in the definition of the cost function in (25) depends on the coefficients of the adaptive filter and we have neglected such dependencies in the derivation of the update equations. However, we will see in the experimental results that there is not a significant loss of performance with the use of this approximation.

Similarly, the update equation for $\mathbf{V}_m(n)$ is given by

$$\mathbf{V}_m(n+1) = \mathbf{V}_m(n) + \mu\alpha_{v,m}(n)\mathbf{k}_v(n)e(n), \tag{32}$$

where $\alpha_{v,m}(n)$ and $\mathbf{k}_v(n)$ are given by

$$\alpha_{v,m}(n) = \sigma_m(n) y_{p-l,m}(n) \tag{33}$$

and

$$\mathbf{k}_v(n) = \mathbf{R}^{-1}_{p-l}(n)\mathbf{X}_{N,p-l}(n), \tag{34}$$

respectively. In (34), $\mathbf{R}_{p-l}(n)$ is the least-squares estimate of the autocorrelation matrix of $\mathbf{X}_{N,p-l}(n)$ obtained as

$$\mathbf{R}_{p-l}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{X}_{N,p-l}(i)\mathbf{X}^T_{N,p-l}(i). \tag{35}$$

The update equation for $\sigma_m(n)$ is given by

$$\sigma_m(n+1) = \sigma_m(n) + \mu\alpha_{\sigma,m}(n)e(n), \tag{36}$$

where $\alpha_{\sigma,m}(n)$ is given by

$$\alpha_{\sigma,m}(n) = y_{l,m}(n)y_{p-l,m}(n). \tag{37}$$

The advantage of this approach is that the gain vector $\mathbf{k}_u(n)$ and $\mathbf{k}_v(n)$ is the same for all the $\mathbf{U}_i(n)$'s and $\mathbf{V}_i(n)$'s corresponding to different branches. Furthermore, it is possible to update $\mathbf{k}_u(n)$ and $\mathbf{k}_v(n)$ using the forward and backward prediction errors associated with the data vectors $\mathbf{X}_{N,l}(n)$ and $\mathbf{X}_{N,p-l}(n)$ similar to the derivation of the fast RLS adaptive filters [5].

5. COMPUTATIONAL COMPLEXITY

The computational complexity associated with a fast RLS adaptive filter implemented in direct form structure is $O(N^{2p-1})$ arithmetical operations for a pth order Volterra system model with N-sample memory. For the fast Gauss-Newton parallel-cascade structure, the complexity depends on the particular decomposition employed. A pth order filter realized using r, $p/2$th order branches require $rO(N^{p/2}) + O(N^p)$ arithmetical operations per iteration. This complexity is comparable to the complexity of the system model.

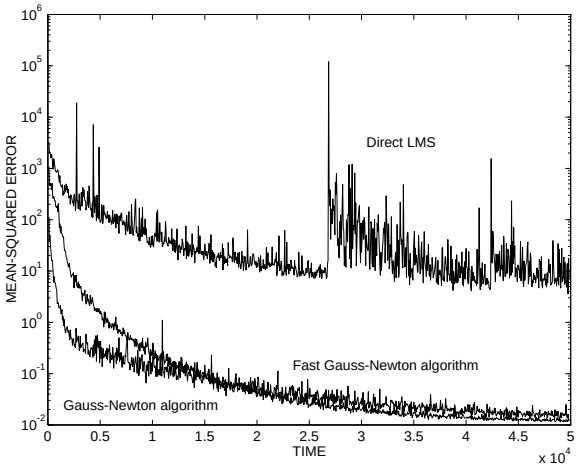

Figure 2. Mean-squared error of the adaptive filters for colored Gaussian input and measurement noise with variance 0.01.

6. SIMULATION EXAMPLES

Two experiments were conducted to evaluate the performance of the algorithm in identifying an unknown, homogeneous fourth-order truncated Volterra filter from measurements of input and output signals in a stationary operating environment. The coefficients of the unknown system were given by

$$h(k_1,k_2,k_3,k_4) = \frac{20.88}{2\pi[1.5^4 + a_1^4 + a_2^4 + a_3^4 + a_4^4]^{3/4}} + u(k_1,k_2,k_3,k_4), \quad (38)$$

where $a_i = (k_i - 1)$, $0 \leq k_1,k_2,k_3,k_4 \leq 4$ and $u(k_1,k_2,k_3,k_4)$ was a uniformly distributed random variable in the range [-0.1,+0.1] that is also symmetric in its indices k_1,k_2,k_3 and k_4. The maximum number of branches in the parallel-cascade realization of the above filter is fifteen. The input signal was a colored Gaussian signal with unit variance and zero mean value. The desired response signal was generated by processing the signal with the system of (38) and then corrupting the output with an uncorrelated white Gaussian noise sequence with zero mean value and variance 0.01. Figure 2 displays a plot of the mean-squared estimation error of direct form LMS and the two Gauss-Newton parallel-cascade algorithms developed in this paper. The results were obtained by averaging the results of one hundred independent experiments. The LMS Volterra filter employed a step size $\mu = 8.3 \times 10^{-7}$. The Gauss-Newton methods used parameter values of $\lambda = 0.99$ and $\mu = 0.01$. These values were selected so that the first version of the Gauss-Newton method and the LMS adaptive filter resulted in almost identical steady-state-errors. The adaptation techniques developed in this paper perform significantly better than the direct form LMS Volterra filter for this particular experiment. The performance of the fast Gauss-Newton algorithm is particularly noteworthy since its computational complexity is of the same order as that of the LMS adaptive filter.

Figure 3 compares the performance of the fast Gauss-Newton algorithm employing ten and fifteen branches. It is difficult to see two separate curves in this plot, indicating that the performance degradation is not significant when

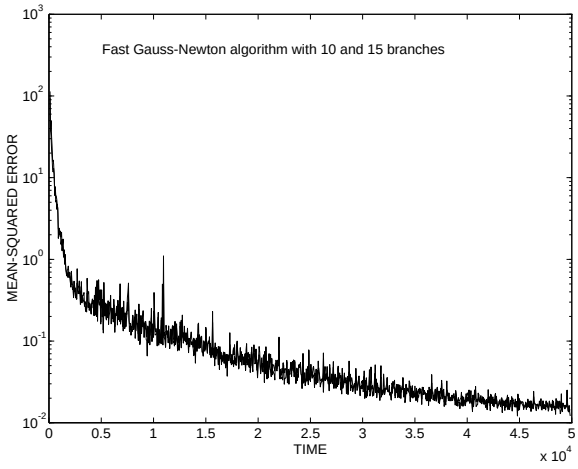

Figure 3. Mean-squared error of the adaptive filters for colored Gaussian input and measurement noise with variance 0.01 with reduced number of branches.

only ten branches are employed in this particular experiment. The steady-state errors evaluated over the last ten thousand samples showed a slightly higher value for the system employing ten branches than the one that used fifteen branches.

7. CONCLUSIONS

This paper presented two Gauss-Newton algorithms for adaptive parallel-cascade truncated Volterra filters. The computational complexity of the fast Gauss-Newton method is comparable to that of the complexity of the system model itself. This, along with its superior performance over the LMS adaptive filter makes this method an attractive option in applications of adaptive Volterra filters.

REFERENCES

[1] M. Schetzen, "The Volterra and Wiener theories of nonlinear systems" New York: Wiley, 1980.

[2] T. M. Panicker and V. J. Mathews, "Parallel-cascade realizations and approximations of truncated Volterra systems," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Volume 3, pp. 1590-1593, Atlanta, Georgia, 1996.

[3] L. Ljung, "System Identification - Theory for the user" Prentice-Hall, Inc., Englewood Cliffs, New Jersy 07632, 1987.

[4] S. Haykin, "Adaptive Filter Theory" third edition, Prentice Hall, Englewood Cliffs, New Jersy, 1996.

[5] J. Lee and V. J. Mathews, "A fast least squares adaptive second-order Volterra filter and its performance analysis" *IEEE Trans. Signal Proc.*, vol. 41, no. 3, pp. 1087-1102, March 1993.