A FREQUENCY-DOMAIN ADAPTIVE POLE-ZERO FILTER WITH APPLICATIONS

John J. Shynk [*] Richard P. Gooch [**] Bernard Widrow [*]

[*]Dept. of Electrical Engineering
Stanford University
Stanford, CA 94305

[**]Applied Signal Technology, Inc.
1185 Bordeaux, Suite D
Sunnyvale, CA 94086

Abstract - An adaptive pole-zero filter implemented in the frequency domain is described. The filter structure is shown to be a parallel combination of first-order sections which permits simple monitoring of the filter poles so that stability can be ensured. Applications in system identification and adaptive array processing are discussed and computer simulation results are given.

INTRODUCTION

Over the last several years, adaptive pole-zero filtering has become an important area of research. The majority of adaptive pole-zero filters discussed in the literature have been direct-form realizations [1,2], although one exception to this is the cascade form introduced by David [3]. Unfortunately, almost all of these adaptive pole-zero filters have problems related to algorithm instability and slow convergence to the optimal solution. These problems have been severe enough to prevent the widespread use and acceptance of adaptive pole-zero filtering even though there are many applications for which a pole-zero filter could provide significantly better performance than an all-zero filter having the same number of coefficients.

Recently, the authors have introduced an adaptive pole-zero filter implemented in the frequency domain which appears to be quite robust [4]. The structure is similar to parallel-form filter realizations and as such is less sensitive to finite-precision effects (than direct forms) and is amenable to hardware (e.g., VLSI) implementation. In addition, the structure permits simple monitoring of the transfer function poles without the large complexity generally attributed to stability checking and projection of unstable poles.

FREQUENCY-DOMAIN ADAPTIVE POLE-ZERO FILTER

The frequency-domain adaptive pole-zero filter (FDAF) is displayed in Fig. 1. Observe that the filter takes a discrete Fourier transform (DFT) of the tapped-delay-line (TDL) output vector at each instant of time to split the input signal $x(n)$ into N *orthogonal* [*] signals $u_k(n)$. Each complex signal $u_k(n)$ is then independently filtered by a first-order subfilter comprised of a single pole and a single zero. Each subfilter is denoted by its z-transform as

$$H_k(n,z) = \frac{b_k(n) + c_k(n)z^{-1}}{1 - a_k(n)z^{-1}} \qquad (1)$$

where $\{a_k(n)\,,\, b_k(n)\,,\, c_k(n)\}$ are adjustable coefficients. The outputs of all subfilters are added to generate the overall adaptive filter output $y(n)$. The output signal $y(n)$ is then compared to the desired response signal $d(n)$ to generate an error signal $e(n)$ that is used by an adaptive algorithm to adjust the filter weights to minimize the mean-square of the error (MSE).

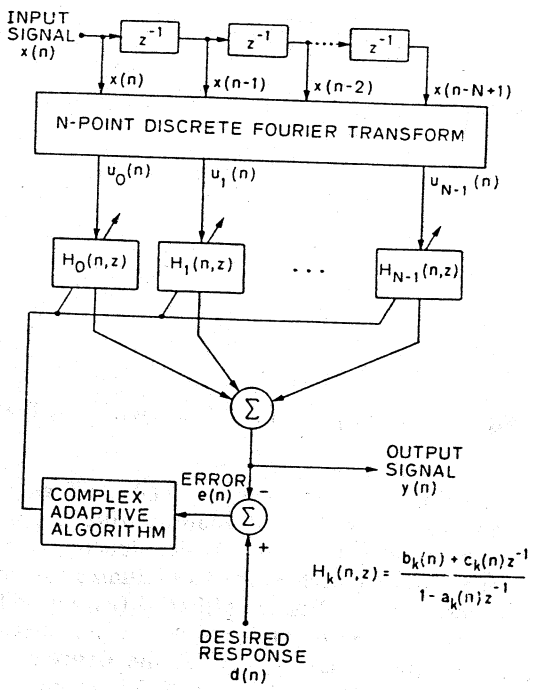


Fig. 1 Frequency-domain adaptive pole-zero filter.

The parallel structure of this filter has several advantages over the direct form of conventional pole-

[*] Precisely speaking, the signals $u_k(n)$ are only approximately orthogonal since their spectra have a small amount of overlap. The amount of overlap can be reduced by increasing N and by windowing [5].

zero adaptive filters. Because the poles $\{a_k(n)\}$ appear explicitly in the first-order subfilters $H_k(n,z)$, it is possible to directly monitor their trajectories during adaptation so that potential instabilities are easily prevented. If one pole attempts to move outside the unit circle, the adaptive update for that pole is simply ignored. Alternatively, an unstable pole could be reflected back inside the unit circle (e.g., by forming its minimum phase equivalent [5]). The update of all other stable poles is performed however, thus insuring a change of the filter state for the next update so that the algorithm is less likely to lock up [2].

It is possible to implement the TDL/DFT structure using a fast Fourier transform (FFT) algorithm. However, since the TDL/DFT structure is essentially a bank of bandpass filters, a more efficient implementation is to use a frequency sampling (FS) structure [5] such as that shown in Fig. 2. The FS structure has a computational complexity of order N per sample compared to order $N \log N$ of the FFT approach. Referring to Fig. 2, the comb filter $1 - z^{-N}$ has N zeros equally spaced around the unit circle, one of which is canceled by a pole at $z = W_N^k$, where $W_N = e^{j2\pi/N}$. Thus, the kth bandpass filter is centered at $\omega = 2\pi k/N$ with an approximate bandwidth of $2\pi/N$.

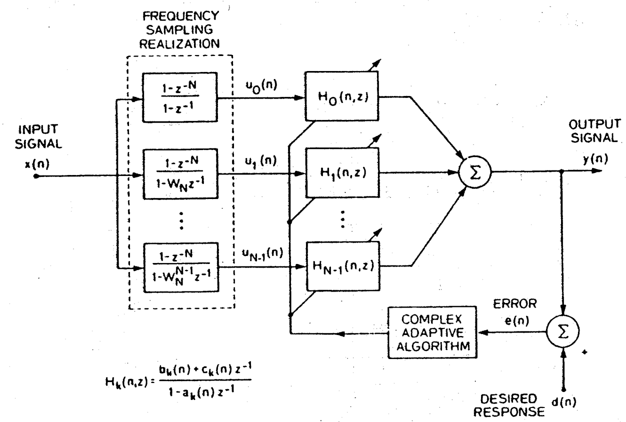

FREQUENCY SAMPLING REALIZATION
INPUT SIGNAL
$x(n)$
$\frac{1-z^{-N}}{1-z^{-1}}$
$\frac{1-z^{-N}}{1-W_N z^{-1}}$
$\frac{1-z^{-N}}{1-W_N^{N-1} z^{-1}}$
$u_0(n)$
$u_1(n)$
$u_{N-1}(n)$
$H_0(n,z)$
$H_1(n,z)$
$H_{N-1}(n,z)$
Σ
OUTPUT SIGNAL
$y(n)$
ERROR
$e(n)$
COMPLEX ADAPTIVE ALGORITHM
Σ
+
DESIRED RESPONSE
$d(n)$
$H_k(n,z) = \frac{b_k(n) + c_k(n) z^{-1}}{1 - a_k(n) z^{-1}}$

Fig. 2 Frequency-sampling realization of the FDAF.

We can therefore view the TDL/DFT as a means of converting a wideband adaptive filter into several *narrowband* adaptive filters which function somewhat independently of each other to minimize the common output error $e(n)$. Each subfilter $H_k(n,z)$ modifies the spectrum of $u_k(n)$ in such a way that the spectrum of $y(n)$ attempts to match that of the desired response $d(n)$. If the spectrum $S_d(z)$ of $d(n)$ is rational and the order of the denominator polynomial of $S_d(z)$ is less than or equal to $2N$, then the FDAF can exactly match $S_d(z)$. In other words the FDAF can exactly model an arbitrary rational system of order N. Details of this are given in [4].

Note however that both the TDL/DFT structure and the FS structure have two bins centered on the real axis so that, if $x(n)$ is real, then $u_0(n)$ and $u_{N/2-1}(n)$ will be real. It turns out that the adaptive algorithm used to update a subfilter will not generate complex coefficients if the subfilter input is real. Consequently, the FDAF would require a DFT with order greater than N to exactly model an Nth order system that has no real poles.

It is possible to overcome this problem by using a *generalized* DFT in place of the standard DFT. In this case, the inputs to the subfilters are given by

$$u_k(n) = \frac{1}{N} \sum_{m=0}^{N-1} x(n-m)\, W_N^{-m(k+k_0)} \qquad (2)$$

where $-1/2 \leq k_0 \leq 1/2$. Equation (2) corresponds to the usual DFT where the frequency index has been shifted by the possibly noninteger-valued k_0. This form of the DFT is more desirable since the bin centers can be shifted arbitrarily around the unit circle. In particular, k_0 can be chosen so that no bin centers lie on the real axis.

The adaptive algorithm used to update the filter coefficients is a modified form of the recursive Gauss-Newton algorithm [6]. The coefficient update expression for each subfilter is given by

$$\theta_k(n+1) = \theta_k(n) + \alpha R_k^{-1}(n+1) \psi_k(n) e^*(n) \ , \qquad (3)$$

where the superscript * denotes complex conjugate and the scalar α is a constant that controls the algorithm convergence rate. The coefficient vector $\theta_k(n)$ is defined by

$$\theta_k(n) = \Big(a_k(n) \, , b_k(n) \, , c_k(n) \Big)^T \ , \qquad (4)$$

and the signal vector $\psi_k(n)$ is given by

$$\psi_k(n) = \Big(y_k^f(n-1) \, , u_k^f(n) \, , u_k^f(n-1) \Big)^T \qquad (5)$$

with elements formed by filtering the subfilter input and output signals by the pole polynomial of (1), i.e.

$$y_k^f(n) = y_k(n) + a_k^*(n)\, y_k^f(n-1) \qquad (6a)$$

$$u_k^f(n) = u_k(n) + a_k^*(n)\, u_k^f(n-1) \ . \qquad (6b)$$

The matrix $R_k(n+1)$ is the Hessian matrix and is computed as follows:

$$R_k(n+1) = (1-\alpha)\, R_k(n) + \alpha \psi_k(n) \psi_k(n)^H \ , \qquad (7)$$

where the superscript H refers to conjugate transpose.

Because the DFT outputs $u_k(n)$ are orthogonal, it is possible to adapt the coefficients of each subfilter independently of the other subfilters, and still have satisfactory convergence of the adaptive algorithm. Consequently, the full $N \times N$ block Hessian matrix can be replaced by an $N \times N$ block *diagonal* matrix where each block has rank three and corresponds to the Hessian matrix for each subfilter (i.e., $R_k(n)$ of (7)). This permits calculation of the inverse Hessian to be accomplished by inverting N (3x3) submatrices.

It is important to note that the Hessian matrix could be left out of the update (3) altogether in order to reduce the algorithm's computational complexity. Unfortunately simulations have shown that this reduction in complexity occurs at the expense of slower convergence.

The initial coefficient values are generally chosen to be zero and the initial Hessian matrix is given by $R_k(0) = \delta I$, where I is the identity matrix and δ is a scalar chosen to approximate the power of $u_k(n)$.

APPLICATION IN SYSTEM IDENTIFICATION

In system identification, the desired response signal and the adaptive filter input are often assumed to be generated as follows:

$$d(n) = G(z)\,x(n) + v(n) \tag{8}$$

where $x(n)$ and $v(n)$ are zero-mean, mutually uncorrelated signals, and $G(z)$ is the system to be identified.

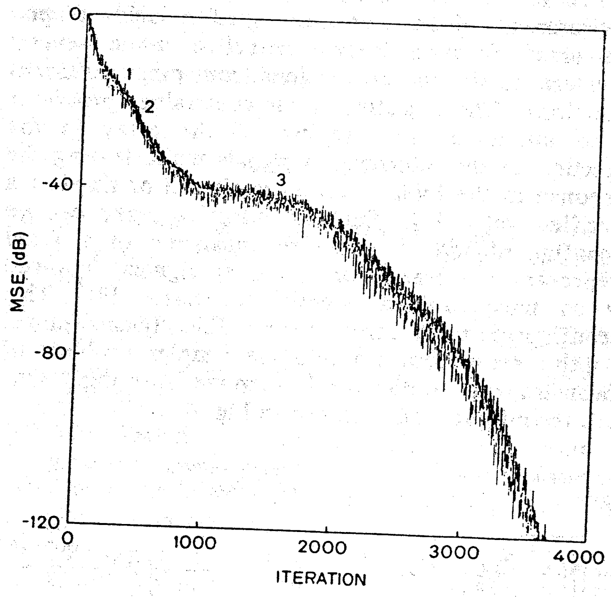
A system identification example where $G(z)$ had poles close to the unit circle was presented in [4]. There we demonstrated that the FDAF performed remarkably well under extremely difficult conditions. In this section, we present results showing the MSE and the trajectories of the filter poles during adaptation.

The particular system $G(z)$ to be identified was

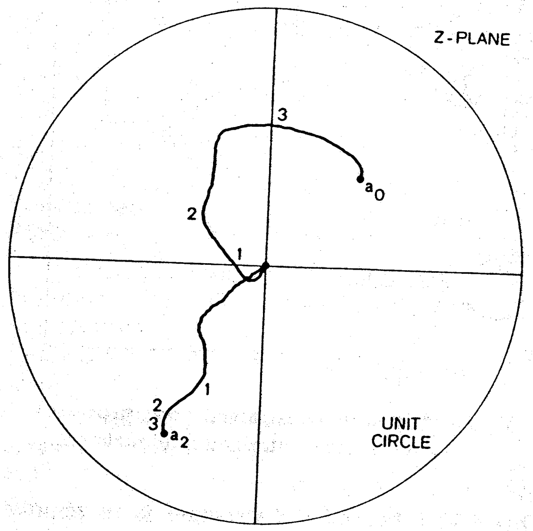
$$G(z) = \frac{K(1-.9z^{-1})(1-.81z^{-2})}{(1-.71z^{-1}+.25z^{-2})(1+.75z^{-1}+.56z^{-2})} \tag{9}$$

which has four complex poles at $p_{1,2} = 0.50\angle\pm45°$ and $p_{3,4} = 0.75\angle\pm120°$, and three zeros at $z_{1,2} = 0.90\angle\pm90°$ and $z_3 = 0.90$. The input signal $x(n)$ was a real white noise process and the constant K was chosen such that $G(z)\,x(n)$ had unit variance. There was no additive noise $v(n)$. The specific FDAF parameters were $\alpha = .01$, $N = 4$, $\delta = 0.25$, and $k_0 = 0.5$. Instead of updating the coefficients independently as given by (3), they were updated simultaneously using the full block-matrix generalization of (7).

Fig. 3a shows the MSE learning curve and Fig. 3b shows the pole trajectories during adaptation. These curves were obtained by averaging the results of 25 independent computer simulation runs. The trajectories of only two poles are shown since the other two poles are simply the complex conjugates of those shown. Three locations have been marked on each trajectory for comparison with the MSE learning curve. Observe that the MSE essentially converges to zero (i.e., less than -120 dB) by iteration 4000 and that the poles converge to those of $G(z)$. Also note that the MSE decreases quickly to -40dB by iteration 1000, even though the poles are not close to convergence.



(a)



(b)

Fig. 3 (a) MSE learning curve and (b) pole trajectories.

Between iterations 1000 and 1500 the trajectory of pole a_0 changes direction toward the corresponding pole of $G(z)$; during this change, the MSE convergence rate is somewhat reduced. Beyond iteration 2000, the MSE again decreases quickly as the poles converge toward the ideal values to give zero MSE. Although not shown here, the coefficients $\{b_k(n), c_k(n)\}$ have also converged to those predicted by the partial fraction expansion of [4] where the DFT of (2) has been used.

APPLICATION IN ADAPTIVE ARRAY PROCESSING

A constrained array processor operates in an environment where a desired signal is incident upon the array from a known direction while several interference signals are incident from other unknown directions. The objective of the constrained processor is to minimize the response of the array in the direction of the interference signals while leaving the response in the look direction unaltered or fixed at a specified value [7]. Such an adaptive array can be reconfigured into a structure consisting of a fixed preprocessor operating on the array signals, followed by an unconstrained adaptive processor [8]. This reconfiguration transforms the linearly-constrained adaptive array problem into the simpler problem of unconstrained multichannel adaptive filtering. One such reconfiguration is shown in Fig. 4.

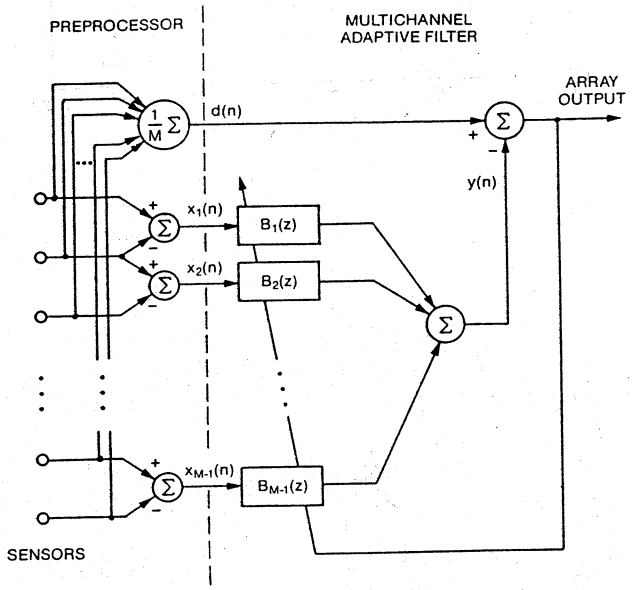


Fig. 4 A linearly constrained array processor using unconstrained minimization.

The basic idea behind the structure is to remove the look-direction signal from the adaptive array filter inputs $x_1(n)$ through $x_{M-1}(n)$ and then add it back into the array output by summing all of the sensor signals. The look-direction constraint is thus ensured to be fixed (at one) regardless of the state of the array filters $B_1(z)$ through $B_{M-1}(z)$. Interference signals on the other hand appear in the $x_k(n)$ and $d(n)$ inputs and can therefore be canceled from the array output by appropriately adapting the filter weights.

Conventional array processing uses all-zero digital filters for the array filters $\{B_k(z)\}$. It can be shown however, that the optimal broadband array weighting for a linearly-constrained array contains a set of pole-like resonances [9]. This naturally motivates the use of *pole-zero* filters instead of the usual tapped-delay-line (all-zero) filters.

Fig. 5 shows an adaptive pole-zero array processor using the FDAF where, for convenience, the fixed preprocessor of Fig. 4 has not been shown.

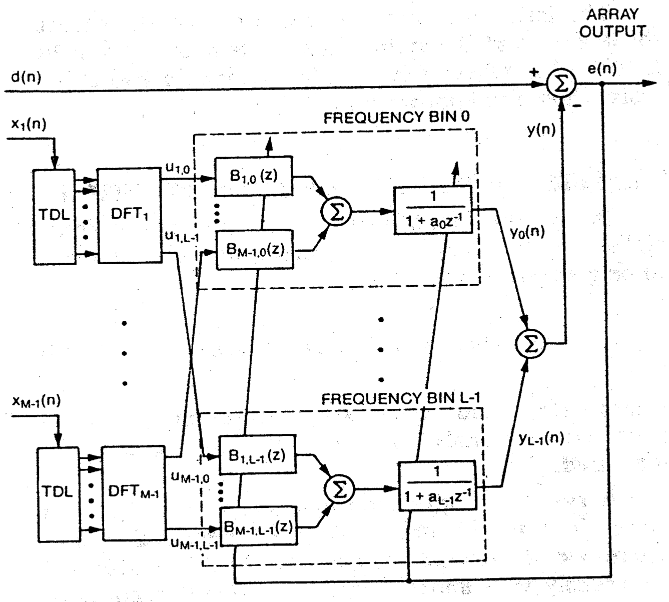


Fig. 5 Frequency-domain adaptive pole-zero array processor.

Observe that each input signal $x_k(n)$ $(k = 1,\ldots, M-1)$ is first processed by a DFT to produce L orthogonal output signals $u_{km}(n)$ $(m = 0,\ldots, L-1)$. The DFT signals $\{u_{km}(n)\}$ are then grouped by frequency bins and filtered by a bank of $M-1$ all-zero digital filters with transfer functions

$$B_{km}(z) = b_{km0} + b_{km1}z^{-1} , \qquad (10)$$

where $\{b_{km0}, b_{km1}\}$ are the adaptive coefficients. The outputs of the filters are then summed and filtered by a single-pole filter with transfer function $1/(1 + a_m z^{-1})$ to produce the intermediate output signal $y_m(n)$. The overall output $y(n)$ is obtained by summing these intermediate signals, which is then subtracted from $d(n)$ to give the array output $e(n)$. An adaptive algorithm similar to that of (3) is then used to update the adaptive coefficients.

We now present simulation results comparing the signal-to-interference ratio (SIR) convergence curve of the array processor of Fig. 5 with that of a conventional array processor of similar complexity. A wideband interference signal with 50% relative bandwidth was generated by passing white noise through a fourth-order Butterworth filter having a normalized frequency of $\omega_0 = \pi/2$ rads and a bandwidth of $\pi/2$ rads. This signal impinged upon an array consisting of two sensors ($M = 2$) separated by a distance equivalent to one-half wavelength at the signal's center frequency. The bearing angle of the interference signal was chosen to be 20° off the look direction with a power level 20 dB above that of the look-direction signal. There was no thermal noise.

The tap spacing for all arrays was also chosen to be one-half wavelength at the signal's center frequency. The FDAF was comprised of four poles ($L = 4$) and four zeros ($M = 2$) for a total of nine coefficients. In order to be of similar complexity as the FDAF, the filter of the all-zero array was chosen to have nine feedforward coefficients. The recursive-least-squares (RLS) algorithm was used to adapt the all-zero array and (3) (generalized in an obvious way) was used to adapt the pole-zero array. The time constant of both adaptive algorithms was set at 100 samples (i.e., $\alpha = 0.99$) and the Hessian matrices were initialized to the identity matrix times the known signal power.

Fig. 6 shows the SIR convergence curves resulting from averaging 500 independent computer simulations.

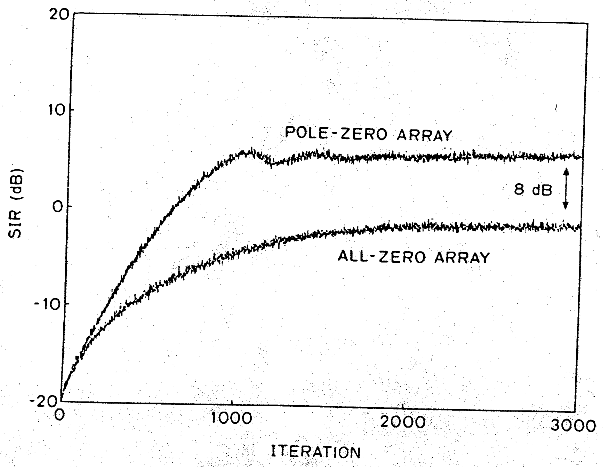


Fig. 6 SIR convergence curves of the pole-zero and the all-zero array processors.

Notice that the convergence rates of the all-zero and pole-zero arrays are comparable, however the converged SIR of the pole-zero array exceeds that of the all-zero array by about 8 dB. Thus, the FDAF generates a better approximation of the optimal weighting than an all-zero array.

A more complete description of the FDAF array processor and additional simulations are given in [9].

CONCLUSIONS

A frequency-domain adaptive pole-zero filter (FDAF) has been discussed. As a result of the parallel structure, the filter is always stable and is less sensitive to coefficient quantization (than direct forms). Furthermore, the filter can model an arbitrary rational system. Computer simulations were presented demonstrating that the poles of the FDAF converge to those of the system being identified (Fig. 3). Application of the FDAF to array processing was also discussed, and computer simulations were presented to show that the FDAF performs significantly better, in terms of increased SIR, than conventional all-zero adaptive array processing (Fig. 6).

REFERENCES

[1] B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1985.

[2] C.R. Johnson, Jr., "Adaptive IIR filtering: Current results and open issues," *IEEE Trans. Inform. Theory*, vol. IT-30, no. 2, pp. 237-250, Mar. 1984.

[3] R.A. David, "IIR adaptive algorithms based on gradient search techniques," Dept. of Elec. Engin., Stanford Univ., Stanford, CA, Aug. 1981 (Ph.D. dissertation).

[4] J.J. Shynk and R.P. Gooch, "Frequency-domain adaptive pole-zero filtering," *Proc. IEEE*, vol. 73, no. 10, pp. 1526-1528, Oct. 1985.

[5] L.R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

[6] J.J. Shynk, "A complex adaptive algorithm for IIR filtering," *IEEE Trans. Acoust., Speech, Signal Proc.*, in review.

[7] O.L. Frost, III, "An algorithm for linearly constrained adaptive array processing," *Proc. IEEE*, vol. 60, no. 8, pp. 926-935, Aug. 1972.

[8] L.J. Griffiths and C.W. Jim, "An alternative approach to linearly constrained adaptive beamforming," *IEEE Trans. Antennas and Propagation*, vol. AP-30, no. 1, pp. 27-34, Jan. 1982.

[9] R.P. Gooch and J.J. Shynk, "Wideband adaptive array processing using pole-zero digital filters," *IEEE Trans. Antennas and Propagation*, Special issue on adaptive array processing, to appear Mar. 1986.

A FREQUENCY-DOMAIN ADAPTIVE POLE-ZERO FILTER WITH APPLICATIONS

John J. Shynk, Richard P. Gooch, and Bernard Widrow

Reprinted from the NINETEENTH ASILOMAR
CONFERENCE ON CIRCUITS, SYSTEMS AND COMPUTERS
November 6-8, 1985 Pacific Grove, CA