Adaptive Pole Estimation

ARYE NEHORAI, SENIOR MEMBER, IEEE, AND DAVID STARER, MEMBER, IEEE

Abstract—A new adaptive algorithm is developed for on-line estimation of the poles of autoregressive (AR) processes. The method estimates the poles directly from the data without intermediate estimation of the AR coefficients or polynomial factorization. It converges rapidly, is computationally efficient, and attains the Cramer-Rao bound (CRB) asymptotically. A closed-form expression for the asymptotic CRB is provided. Convergence to the true solution is proved, and methods are discussed for extending the algorithm for use with more general (e.g., ARMA) models. Numerical examples are presented to demonstrate the performance of the algorithm.

I. INTRODUCTION

SYSTEMS are usually identified in terms of the coefficients of their characteristic polynomials (or transfer functions). However, one is often more interested in the roots of these polynomials than in their coefficients. For example, we state the following.

1) In speech analysis, the positions of the roots of the autoregressive (AR) polynomial (i.e., the poles) determine the formant frequencies and bandwidths—these are the perceptually important parameters (see, e.g., [7]).

2) In speech coding, transmission of parameters over noisy communication channels requires that the signal should be parametrized in a robust form which can tolerate disturbances added in the transmission process. The quality of the speech is dependent (among other things) on the positions of the poles of the AR model. However, the pole positions are sensitive functions of the AR coefficients, and small perturbations in the coefficients can cause the poles to shift significantly. Transmission of the coefficients over noisy channels can therefore lead to a substantial loss in quality. A more robust approach is to transmit the actual pole positions, and this would involve parametrizing the speech in terms of its poles at the transmitter.

3) In sensor array processing, direction-of-arrival estimation and tracking using a uniform linear array can be formulated as an adaptive polynomial rooting problem (see, e.g., [3] and [6]).

4) In biomedical engineering, the trajectory of the roots of the AR model of EEG signals has been used [8] to predict the onset of seizure.

In situations such as these, representation of the signals or systems in terms of their poles (or zeros) is more appropriate than representation in terms of coefficients.

Traditionally, the roots are located in a two-stage procedure; initially the polynomial coefficients are estimated using conventional identification techniques such as the Recursive Least Squares (RLS) algorithm, and then the roots of the resulting polynomial are found using standard factorization schemes such as Müller’s method (see, e.g., [9]). However, this approach is often computationally too intense to apply to on-line tracking of system roots where updated estimates are required at every sampling interval.

Thus, there exists a need for an algorithm which can satisfy the two simultaneous requirements of a) directly providing estimates of a system’s poles (rather than the coefficients) and of b) providing new estimates as each data sample is received. The requirement for on-line estimates implicitly calls for an algorithm which uses computationally efficient updating procedures and a fixed amount of memory, regardless of the number of samples processed. For stationary processes, the estimates are required to become more accurate with each update. For nonstationary processes, the estimates are required to track the time-varying parameters.

This paper (see also [5]) presents an algorithm which satisfies the above requirements. The method estimates the poles without explicitly estimating and factorizing the AR polynomial. It is based on a direct parametrization of the AR process in terms of its poles rather than its coefficients, and uses the Gauss–Newton recursive prediction error (RPE) method [10]–[12]. This algorithm uses an exact expression for the gradient, and converges considerably faster than previous pole estimation methods based on the LMS algorithm [3]. In addition, it is asymptotically statistically efficient and is computationally more efficient than conventional two-stage pole estimation methods.

When compared to relaxation-type methods for estimating poles (see, e.g., [2]), the algorithm has the advantage of possessing well-understood convergence characteristics. In relaxation methods, parts of the parameter vector are estimated individually; for example, the system could be separated into a cascade of second-order sections, and the parameters of the individual sections could be estimated consecutively. Off-line relaxation methods are guaranteed to converge to a local minimum. However, on-line relaxation methods are only approximations to the off-line cases and their convergence is not necessarily as-
sured. In contrast, the convergence properties of the proposed method for the simultaneous on-line estimation of the entire parameter vector are well understood. Convergence analysis of the method is provided, and it is proved that the algorithm converges to the true poles of the process.

The paper is organized as follows. In the remainder of this section, the pole estimation problem is formulated. The derivation of the basic pole estimation algorithm is presented in Section II, and its convergence is analyzed in Section III. In Section IV, a closed-form expression for the Cramer-Rao Bound (CRB) for any unbiased AR pole estimator is derived. The algorithm is extended in Section V in order to deal with more general (e.g., ARMA) processes. In Section VI, numerical simulation results are presented to demonstrate that the algorithm attains the CRB asymptotically, is computationally efficient, and can readily be extended. Section VII concludes the paper. Appendices present closed-form expressions for certain mappings and gradients required in the text. For completeness, an alternative two-step algorithm for pole estimation based on coefficient matching is presented in Appendix C.

A. Problem Statement

An nth-order autoregressive process is modeled as

\[ y(t) = \frac{1}{A(q^{-1})} e(t) \]  

(1.1)

where \( e(t) \) is zero-mean white noise with variance \( \sigma^2 \), and \( A(q^{-1}) \) denotes a monic polynomial in the delay operator \( q^{-1} \). The polynomial \( A(q^{-1}) \) can be represented as a sum or as a product (cascade form) as follows:

\[ A(q^{-1}) = \sum_{i=0}^{n} a_i q^{-i} = \prod_{i=1}^{n} (1 - \lambda_i q^{-1}). \]  

(1.2)

The parameters \( \lambda_i, i = 1, \cdots, n \), are the poles of the system and are assumed to have magnitude less than unity. This is equivalent to assuming that \( y(t) \) is stationary. The coefficients \( a_i \) are assumed to be real, implying that the poles should be real or occur in complex conjugate pairs.

The purpose of the proposed algorithm is to obtain accurate on-line estimates of the poles \( \lambda_i \) in a computationally efficient way from measurements of the data \( y(t) \).

II. DIRECT ADAPTIVE POLE ESTIMATION

A. Parametrization

Suppose that \( A(q^{-1}) \) has \( m \) pairs of complex conjugate roots and \( p \) real roots such that \( n = 2m + p \). Let \( \lambda_{1k}, \lambda_{2k}, \cdots, \lambda_{mk}, k = 1, \cdots, m \), be the \( k \)th pair of complex conjugate roots, and let \( \lambda_{jk}, j = 1, \cdots, p \), be the \( j \)th real root. Then \( A(q^{-1}) \) can be regarded as a cascade of first- and second-order filter sections, i.e.,

\[ A(q^{-1}) = \prod_{j=1}^{p} F_j(q^{-1}) \prod_{k=1}^{m} S_k(q^{-1}), \]  

(2.1a)

where

\[ F_j(q^{-1}) = (1 - \zeta_j q^{-1}) \]  

(2.1b)

\[ S_k(q^{-1}) = (1 - \lambda_k q^{-1})(1 - \lambda_k^* q^{-1}). \]  

(2.1c)

Define \( \rho_k \) and \( \omega_k \) to be the radius and positive angle of the \( k \)th pair of complex conjugate roots of \( A(q^{-1}) \) so that

\[ \lambda_k = \rho_k e^{j \omega_k}, \]  

(2.2)

where \( j = \sqrt{-1} \). The unknown parameter vector is defined as

\[ \theta = [\rho_1^T \omega_1^T \zeta_1^T]^T \]  

(2.3)

where the superscript \( T \) denotes matrix transposition and

\[ \rho = [\rho_1, \rho_2, \cdots, \rho_m]^T, \]  

(2.3a)

\[ \omega = [\omega_1, \omega_2, \cdots, \omega_m]^T, \]  

(2.3b)

\[ \zeta = [\zeta_1, \zeta_2, \cdots, \zeta_p]^T. \]  

(2.3c)

The polar representation is the one favored in this application since the parameters have a physical interpretation related to the positions and bandwidths of spectral peaks. Equivalent parametrization in terms of rectangular or other coordinates is also possible (see also Section V-A).

It is assumed that the order \( n \) of the process, as well as the proportion of real and complex poles, is known. If the order is not known, it can be determined using conventional techniques (such as the Akaike Information Criterion). The case of unknown proportion of real and complex poles can be overcome by appropriate parametrization, and Section V-A discusses this issue. In most practical cases, however, this proportion is known. For example, in speech processing, the commonly used LPC model accounts for the vocal tract resonances and hence the poles appear in complex conjugate pairs (see, e.g., [7]). The anti-aliasing filtering applied to the speech before sampling could add a known number of real poles.

B. Algorithm Derivation

The RPE algorithm with quadratic error cost function estimates \( \theta \) by recursively minimizing

\[ V(\theta) = \gamma(t) \sum_{k=1}^{t} \left( \prod_{j=k+1}^{t} \omega(j) \right) e^2(k, \theta). \]  

(2.4)

where \( \epsilon(k, \theta) \) is the prediction error, \( \omega(j) \) is a weighting factor discussed in Section II-C, and \( \gamma(t) \) is a normalizing factor. More details regarding this cost function can be found, for example, in [10].

To implement the algorithm, expressions for the prediction error and the error gradient with respect to \( \theta \) are needed. These are derived in the following subsections.

1) The Prediction Error: The prediction error \( \epsilon(t) \) of the AR process (1.1) is given by

\[ \epsilon(t) = y(t) + \sum_{i=1}^{n} a_i y(t - i) \]

\[ = y(t) - a^T \varphi(t) \]  

(2.5a)
where \( a \) is the vector of polynomial coefficients
\[
a = [a_1, a_2, \ldots, a_n]^T
\]  
(2.5b)
and \( \varphi(t) \) is the regression vector
\[
\varphi(t) = [-y(t-1), -y(t-2), \ldots, -y(t-n)]^T.
\]  
(2.5c)

2) The Error Gradient: Let the error gradient vector be defined as
\[
\psi'(t) = -\frac{\partial \psi(t)}{\partial \theta^T} = -\left[ \frac{\partial \psi(t)}{\partial \theta_1}, \ldots, \frac{\partial \psi(t)}{\partial \theta_n} \right].
\]  
(2.6)

Applying the chain rule, this can be written as
\[
\psi'(t) = -\frac{\partial \psi(t)}{\partial a^T} \frac{\partial a}{\partial \theta^T}
\]  
(2.7a)
where, from (2.5a)
\[
\frac{\partial \psi(t)}{\partial a} = -\varphi'(t)
\]  
(2.7b)
and, for polar coordinates, \( \partial a / \partial \theta^T \) is given by the partitioned matrix
\[
\frac{\partial a}{\partial \theta^T} = \begin{bmatrix}
\frac{\partial a_1}{\partial \theta_1} & \frac{\partial a_2}{\partial \theta_1} & \cdots & \frac{\partial a_m}{\partial \theta_1} \\
\frac{\partial a_1}{\partial \theta_2} & \frac{\partial a_2}{\partial \theta_2} & \cdots & \frac{\partial a_m}{\partial \theta_2} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial a_1}{\partial \theta_n} & \frac{\partial a_2}{\partial \theta_n} & \cdots & \frac{\partial a_m}{\partial \theta_n}
\end{bmatrix}.
\]  
(2.8)

For simplicity of exposition, it is assumed in the following that all poles are distinct. However, the discussion can be extended readily to deal with multiple poles.

Entries in the submatrix \( \partial a / \partial \theta^T \) can be found by differentiating (2.1a) with respect to \( \rho_i \) to give
\[
\frac{\partial A(q^{-1})}{\partial \rho_i} = \delta A_i(q^{-1}) \prod_{i=1}^{m} S_i(q^{-1}) \prod_{i=1}^{\rho} F_i(q^{-1})
\]  
(2.9)
but, from (1.2)
\[
\frac{\partial A(q^{-1})}{\partial \rho_i} = \sum_{i=0}^{n} \frac{\partial a_i}{\partial \rho_i} q^{-i}.
\]  
(2.10)

In polar coordinates
\[
S_i(q^{-1}) = 1 - 2\rho_i \cos \omega_i q^{-1} + \rho_i^2 q^{-2}
\]  
(2.11)
so that
\[
\frac{\partial S_i(q^{-1})}{\partial \rho_i} = -2 \cos \omega_i q^{-1} + 2\rho_i q^{-2}.
\]  
(2.12)

Equating the right-hand sides of (2.9) and (2.10), multiplying throughout by \( \delta S_i(q^{-1}) \), and making use of the definition (1.2) of \( A(q^{-1}) \) gives
\[
S_i(q^{-1}) \sum_{i=0}^{n} \frac{\partial a_i}{\partial \rho_i} q^{-i} = \delta S_i(q^{-1}) \sum_{i=0}^{n} a_i q^{-i}.
\]  
(2.13)

It is now possible to extract a recursive expression for the derivative by equating coefficients of \( q^{-i} \) in the above equation. Performing this operation leads to
\[
\frac{\partial a_i}{\partial \rho_j} = 2\rho_i \cos \omega_i \frac{\partial a_{i-1}}{\partial \rho_j} - \rho_i \frac{\partial a_i}{\partial \rho_j}
\]
\[-2 \cos \omega_i a_{i-1} + 2\rho_i a_{i-2}
\]  
(2.14a)
for \( 2 \leq i \leq n, 1 \leq k \leq m \), with boundary conditions
\[
\frac{\partial a_0}{\partial \rho_j} = 0, \quad \frac{\partial a_1}{\partial \rho_j} = -2 \cos \omega_i.
\]  
(2.14b)

The above equations define a second-order recursion which is used to calculate the gradient of the polynomial coefficients with respect to the pole radii, enabling the submatrix \( \partial a / \partial \theta^T \) to be built. These recursive expressions avoid redundant calculations and are computationally efficient.

The submatrix \( \partial a / \partial \theta^T \) is found in an exactly analogous manner, and this procedure leads to the recursion
\[
\frac{\partial a_i}{\partial \omega_j} = 2\rho_i \cos \omega_i \frac{\partial a_{i-1}}{\partial \omega_j} - \rho_i \frac{\partial a_i}{\partial \omega_j}
\]
\[+2\rho_i \sin \omega_i a_{i-1}
\]  
(2.15a)
for \( 2 \leq i \leq n, 1 \leq k \leq m \), with boundary conditions
\[
\frac{\partial a_0}{\partial \omega_j} = 0, \quad \frac{\partial a_1}{\partial \omega_j} = 2\rho_i \sin \omega_i.
\]  
(2.15b)

The submatrix \( \partial a / \partial \theta^T \) can also be found by using the above method, giving
\[
\frac{\partial a_i}{\partial \xi_k} = \xi_k \frac{\partial a_{i-1}}{\partial \xi_k} - a_{i-1}
\]  
(2.16a)
subject to the boundary condition
\[
\frac{\partial a_0}{\partial \xi_k} = 0.
\]  
(2.16b)

Similar recursions for complex poles can also be found [13].

C. The Direct Adaptive Pole Estimation Algorithm

The Gauss–Newton RPE algorithm for direct pole estimation using the prediction error and its gradient derived in the previous section is given in Table I.

Parametrization: The algorithm in Table I is a general pole estimator which is applicable for different parametrizations (e.g., polar, rectangular, or second-order section coefficients). The only parameter-dependent steps are those marked with asterisks. This independence of the specific form of parametrization can be exploited to yield root estimation methods that are well suited to particular situations. Examples are given in Section V. In the case of polar coordinates, \( \hat{a}(t) \) is calculated from \( \hat{\theta}(t) \) using the recursion given in (A.6), and \( \partial a / \partial \theta^T \) is calculated using (2.14a)-(2.16b).

Admissible Parameters: In the algorithm, the estimate should be projected into the admissible parameter set \( \mathfrak{D} \)
TABLE I

<table>
<thead>
<tr>
<th>Adaptive AR Pole Estimation Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize ( \hat{\theta}(0), \hat{\phi}(0), P(0), \phi(t), \psi(t), \zeta(t), w(t), w_\infty, w_\infty(\infty) )</td>
</tr>
<tr>
<td>For ( t = 1, 2, \cdots ) do</td>
</tr>
<tr>
<td>( \hat{\theta}(t) = \hat{\theta}(t) - \hat{\theta}(t) (t - 1) \hat{\psi}(t) )</td>
</tr>
<tr>
<td>( L(t) = (\phi(t - 1) \psi(t))/w(t) + \psi'(t) P(t - 1) \psi(t) )</td>
</tr>
<tr>
<td>( P(t) = (P(t - 1) - L(t) \psi(t) P(t - 1) \psi(t))/w(t) )</td>
</tr>
<tr>
<td>( \hat{\theta}(t) = { \hat{\theta}(t - 1) + L(t) \hat{\theta}(t) }_w )</td>
</tr>
<tr>
<td>Calculate ( \hat{\theta}(t) ) from ( \hat{\theta}(t) )</td>
</tr>
<tr>
<td>( \phi(t + 1) = { -\phi(t), -\psi(t), \cdots, -\psi(t - n + 1) }^T )</td>
</tr>
<tr>
<td>( \psi(t + 1) = [\psi(t), \psi(t), \cdots, \psi(t - n + 1)] )</td>
</tr>
<tr>
<td>( w(t + 1) = w(\infty) - (w(\infty) - w(t))w_\infty )</td>
</tr>
</tbody>
</table>

which, in our case, consists of the set of all distinct poles (see also Section III). The projection is indicated by the subscript \( \Sigma \).

Initial Values: The recommended initial values for the algorithm are: \( P(0) = cI \), where \( c \) is some constant and \( \phi(1) = \psi(1) = 0 \). The initial values for the pole estimates can be assigned using any a priori knowledge concerning their likely positions. In the absence of such knowledge, a possible choice is to distribute the poles uniformly around a circle, inside and concentric with the unit disk. For most parametrizations, to ensure convergence, \( \hat{\theta} \) should not be initialized with zeros.

Forgetting Factor for Stationary Processes: For stationary processes, the recommended values for the forgetting factor are \( \omega_0 = 0.99 \), \( w(1) = 0.95 \), and \( w(\infty) = 1 \) (see, e.g., [10]).

Forgetting Factor for Nonstationary Processes: There are numerous standard methods which can be used to enable the RPE algorithm to cope with nonstationary processes. These are discussed in, for example, [10, pp. 274-278] and [14, pp. 62–68]. Tracking time-varying parameters involves several compromises which are well documented in these references and the problem is therefore not addressed in this paper.

Number of Computations: In considering the number of computations required, use is made of the flop defined by Golub and Van Loan [15] to be the amount of computation associated with the statement \( s := s + a_i b_j \). The only extra computational burden imposed by the above algorithm in addition to that required by the RLS method is associated with the three steps of calculating the coefficients \( \hat{\theta}(t) \), evaluating the Jacobian \( \partial a / \partial \theta^T \), and forming \( \psi(t + 1) \). Consider an \( m \times n \)-order system consisting of \( n/2 \) complex-conjugate pole pairs—the worst case in terms of number of computations.

1) Calculation of the coefficients from the poles using (A.6) requires \( n^2 \) flops.
2) Calculation of the Jacobian matrix \( \partial a / \partial \theta^T \) requires \( 5n^2 \) flops. Five flops are required to perform the recursion (2.14a), and the recursion needs to be repeated for \( i = 1, 2, \ldots, n \) and \( k = 1, 2, \ldots, n/2 \). Thus, evaluation of the matrix \( \partial a / \partial \theta^T \) requires \( 5n^2 / 2 \) flops. A similar number of operations is required for computation of \( \partial a / \partial \theta^T \).
3) Formation of \( \psi(t + 1) \) involves a matrix-vector product which requires \( n^2 \) flops.

Thus, the algorithm requires a total of \( 7n^2 \) flops per update in addition to computations needed by the RLS method. The conventional two-stage method, wherein the AR polynomial is updated and factorized at each time interval, is more computationally intense since the second stage (performing the factorization) is nonlinear and requires an iterative solution. Unlike the proposed algorithm, the number of computations for the two-stage method is not fixed. A numerical comparison of the relative computational complexity of conventional pole estimation and the method proposed in this paper is presented in Section VI-D.

III. CONVERGENCE ANALYSIS

It is well known that the error surface used in the least-squares identification of the AR polynomial coefficients is quadratic and therefore possesses a unique global minimum with no false local minima. However, the identification of the AR poles is a nonlinear problem and the error surface is no longer quadratic. The question thus arises as to whether or not there are false local minima to which the algorithm can converge. It will be shown in this section that no such minima exist except for trivial labeling ambiguities and easily avoided subspaces.

It has been shown (see, e.g., [10, Section 4.4]) that, asymptotically (i.e., for long data records), the RPE algorithm converges to a local minimum of the criterion function defined by

\[
V(\theta) = Ex^T(t, \theta).
\]  

(3.1)

The local minima of this function correspond to the set of values of \( \theta \) satisfying

\[
\frac{\partial V(\theta)}{\partial \theta^T} = \frac{\partial V(\theta)}{\partial a} \frac{\partial a}{\partial \lambda} \frac{\partial \lambda}{\partial \theta^T} = 0.
\]  

(3.2)

Each of the three terms above will now be examined individually.

The first term is expanded by differentiating (3.1) and making use of (2.5a) to give

\[
\frac{\partial V(\theta)}{\partial a} = [Ra - r]^T \]  

(3.3a)

where

\[
R = E[\psi(t) \psi'(t)]
\]  

(3.3b)

\[
r = E[\psi(t) \psi(t)]
\]  

(3.3c)

The second term is investigated in Appendix B (see also [16] and [17]) where it is shown that for systems with distinct, simple poles

\[
\frac{\partial a}{\partial \lambda} = -HV
\]  

(3.4a)
where the Hankel matrix $H$ is given by

$$
H = \begin{bmatrix}
1 & a_1 & \cdots & a_{n-1} \\
\vdots & & & \vdots \\
1 & a_1 & \cdots & a_{n-1} \\
\end{bmatrix}
$$

and the Vandermonde matrix $V$ is defined as

$$
V = \begin{bmatrix}
\lambda_1^{n-1} & \cdots & \lambda_1^{n-1} \\
\vdots & \ddots & \vdots \\
\lambda_1 & \cdots & \lambda_n \\
1 & \cdots & 1 \\
\end{bmatrix}
$$

The third term depends on the particular parametrization chosen. For example, for complex-conjugate poles in polar representation where

$$
\lambda = [\rho_1 e^{-i\omega_1}, \ldots, \rho_n e^{-i\omega_n}, \rho_1 e^{i\omega_1}, \ldots, \rho_n e^{i\omega_n}]^T
$$

and

$$
\theta = [\rho_1, \ldots, \rho_n, \omega_1, \ldots, \omega_n]
$$

the derivative is given by the partitioned matrix

$$
\frac{\partial \lambda}{\partial \theta} = \begin{bmatrix}
G_{11} & G_{12} \\
G_{21} & G_{22}
\end{bmatrix} = G
$$

where

$$
G_{11} = \text{diag} \{ e^{i\omega_1} \}^m, \quad G_{12} = \text{diag} \{ +i\rho_k e^{i\omega_1} \}^m_{k=1},
$$

$$
G_{21} = \text{diag} \{ e^{-i\omega_1} \}^m, \quad G_{22} = \text{diag} \{ -i\rho_k e^{-i\omega_1} \}^m_{k=1}.
$$

Equation (3.2) can now be rewritten as

$$
[Ra - r]HVG = 0. \tag{3.7}
$$

This equation can be satisfied either when $Ra = r$ or when $HVG$ is singular. The solution $Ra = r$ yields the true polynomial coefficient vector—the unique global minimum of the usual quadratic criterion in terms of $a$.

Consider now the product $HVG$.

- The Hankel matrix $H$ is clearly invertible. The Vandermonde matrix $V$ is invertible for distinct simple poles, but is singular for multiple poles.
- The matrix $G$ is square and invertible for all values of $\omega_k$, but will contain a zero column if and only if any of the $\rho_k$ is zero. Thus, poles at the origin cause $G$ to become singular.

The parameter space therefore contains three classes of region where the gradient can be zero:

1) regions where $Ra = r$;
2) regions where $\lambda_j = \lambda_k$ for $j \neq k$;
3) regions where $\rho_k = |\lambda_k| = 0$ for one or more $k$.

Regions 2 and 3 are inadmissible and easily recognized. The admissible parameter set $3\pi$ discussed in Section II-C therefore consists of all points in the parameter space except regions 2 and 3. The algorithm can be constrained easily to avoid these regions by disallowing both multiple-pole estimates and estimates at the origin. Projecting the estimates into the admissible parameter set is equivalent to imposing these constraints.

Thus, in the admissible parameter set, the product $HVG$ has full rank. Therefore, the only solution of (3.7) is $a = R^{-1}r$, and this is the true value of the polynomial coefficient vector—the roots of this polynomial are the only points in the parameter space which satisfy (3.7). Since polynomial roots are unique, no false local minima exist.

Although the cost function has no false local minima, it does exhibit multiple minima; in fact there are $n!$ of them. These, however, all correspond to valid parameter estimates and come about due to pole labeling (permutation) ambiguities. For example, the polynomial $A(q^{-1}) = (1 - \theta_1 q^{-1})(1 - \theta_2 q^{-1})$ is described equally well by the parameter vector $\theta^T = [\theta_1, \theta_2]$ as by the permuted parameter vector $\theta^T = [\theta_2, \theta_1]$. The presence of multiple true minima is, in fact, an advantage since the parameter values need not be adjusted substantially from any initial position in order to locate a minimum. This accelerates the convergence of the algorithm. The algorithm's rapid convergence to any one of $n!$ global minima is a surprising, but pleasing, property of the highly nonlinear parametrization. Note that the RPE algorithm cannot converge to saddle points since they are unstable.

**Remark:** The requirement for the imposition of constraints is actually not as stringent as may be inferred from the analysis above. In extensive computer simulations, the situation has never arisen where it has been necessary to impose the constraints. Several factors explain this: 1) Provided that the system does not contain multiple poles, driving noise will tend to separate estimates that accidently coalesce. 2) Attempting to estimate poles at the origin is equivalent to overparametrizing the model. In this case, the corresponding angles are arbitrary, but the radii converge to zero.

**IV. CRAMER-RAO BOUND FOR AR POLE ESTIMATES**

In this section, we derive a closed-form expression for the asymptotic Cramer-Rao lower bound for the AR pole estimates. The results are useful for evaluating the performance of any unbiased estimator of the AR poles and are not limited to the particular algorithm discussed in this paper.

The CRB on the covariance of such an estimator can be written

$$
CRB(\theta) = J_\theta^{-1} \tag{4.1}
$$

where $J_\theta$ denotes the Fisher information matrix associated with $\theta$. In our problem,

$$
J_\theta = \left[ \frac{\partial \theta^T}{\partial \theta} \right] \left[ \frac{\partial \theta}{\partial \theta^T} \right] \tag{4.2}
$$
where $J_a$ is the FIM associated with the polynomial coefficient vector $a$. It is well known (see, e.g., [10]–[12]) that, asymptotically, $J_a$ is given by

$$J_a = \frac{1}{\sigma^2} {\mathcal N}\left[ \nu(t) \nu^T(t) \right]$$

(4.3)

where $N$ is the number of data samples processed. Using the Gohberg–Semenov formula (see, e.g., [18] and [19]), one can write

$$J_a = N \left[ A_1 A_1^T - A_2 A_2^T \right]^{-1}$$

(4.4a)

where

$$A_1 = \begin{bmatrix} 1 & \cdots & \cdots & 1 \\ 1 & \cdots & \cdots & 1 \\ \vdots & \ddots & \ddots & \vdots \\ 1 & \cdots & \cdots & 1 \\ a_1 & \cdots & \cdots & a_1 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n-1} & \cdots & \cdots & a_{n-1} \\ a_n & \cdots & \cdots & a_n \end{bmatrix}$$

(4.4b)

$$A_2 = \begin{bmatrix} a_1 & \cdots & \cdots & a_1 \\ a_2 & \cdots & \cdots & a_2 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n-1} & \cdots & \cdots & a_{n-1} \\ a_n & \cdots & \cdots & a_n \end{bmatrix}$$

(4.4c)

The matrix $\frac{\partial a}{\partial \theta^T}$ used in the transformation (4.2) can be found with the aid of (3.4a) and (3.6a). For the polar parameter vector $\theta$ defined in (2.3), it can be shown that

$$\frac{\partial a}{\partial \theta^T} = 2A_1 B$$

(4.5)

where $B$ is a square matrix which depends on the particular pole parametrization chosen and is nonsingular for simple poles inside the unit circuit (note that $2A_1 B = -HVG$). For polar coordinates $B$ is given by

$$B = [B_\rho B_\omega B_\iota]$$

(4.6a)

where the individual elements of the submatrices are

$$[B_\rho]_{jk} = -\rho^{j-1} \cos(j \omega_k)$$

(4.6b)

$$[B_\omega]_{jk} = \rho^{j-1} \sin(j \omega_k)$$

(4.6c)

$$[B_\iota]_{jk} = -\frac{1}{2} \delta_{jk}$$

(4.6d)

for $1 \leq j \leq n$, $1 \leq k \leq m$, and $1 \leq l \leq p$.

Combining the above results, the CRB is found to be

$$\text{CRB} (\theta) = \frac{1}{4N} B^{-1} A_1^{-1} \left[ A_1 A_1^T - A_2 A_2^T \right] A_1^{-T} B^{-T}$$

(4.7)

where the superscript $-T$ denotes the operations of transposition and inversion. Let

$$A = A_1^{-1} A_2.$$
neously, whereas in the previous methods, the second-order section parameters were updated individually. Simultaneously updating all elements of the parameter vector is important to ensure on-line convergence. In addition, the exact gradient is used here.

The gradient can be obtained using the method outlined in Section II-B-2, resulting in the following recursions:

\[
\frac{\partial a_i}{\partial a_j} = a_{i-1} - a_j \frac{\partial a_{i-1}}{\partial a_j} - \beta_j \frac{\partial a_{i-2}}{\partial a_i} \tag{5.3a}
\]

for \(2 \leq i \leq 2m\) and \(1 \leq j \leq m\) with boundary conditions

\[
\frac{\partial a_0}{\partial a_i} = 0, \quad \frac{\partial a_i}{\partial a_i} = 1 \tag{5.3b}
\]

and

\[
\frac{\partial a_i}{\partial \beta_j} = a_{i-1} - a_j \frac{\partial a_{i-1}}{\partial \beta_j} - \beta_j \frac{\partial a_{i-2}}{\partial \beta_j} \tag{5.4a}
\]

for \(2 \leq i \leq 2m\) and \(1 \leq j \leq m\) with boundary conditions

\[
\frac{\partial a_0}{\partial \beta_i} = 0, \quad \frac{\partial a_i}{\partial \beta_i} = 0 \tag{5.4b}
\]

The algorithm given in Table I can now be applied with the recursions above used for the calculation of \(\partial a / \partial \theta^T\), and (5.1a), (5.1b), and (5.2) used to calculate \(\hat{a}(t)\) from \(\hat{\theta}(t)\) (i.e., (5.1a) to (5.2)) used for the first step marked with an asterisk in Table I, and (5.3a) to (5.4b) used for the second step).

The poles are calculated analytically from \(\hat{\theta}\) as the solutions of \(\lambda^2 + \dot{\alpha}_k \lambda + \dot{\beta}_k = 0\) for \(1 \leq k \leq m\). If the discriminant \(\dot{\alpha}_k^2 - 4\dot{\beta}_k\) is positive, the \(k\)th section represents a pair of real poles, whereas if it is negative, the section represents a pair of complex conjugate poles.

In this parametrization, the initial value of \(\hat{\theta}\) can be zero. This corresponds to poles at \(+1\) and \(-1\). A numerical example of the use of this algorithm when the proportion of real and complex poles is unknown is given in Section VI.

B. Autoregressive Moving-Average (ARMA) Models

The basic pole estimation algorithm in Table I can be extended easily to cater both for more general as well as for more specialized systems. An example of a more general model is the ARMA model defined by

\[
A(q^{-1}) y(t) = C(q^{-1}) e(t) \tag{5.5}
\]

where \(A(q^{-1}) = \sum_{k=1}^{n_a} a_k q^{-k}\) and \(C(q^{-1}) = \sum_{k=1}^{n_c} c_k q^{-k}\) are unknown monic polynomials whose roots are inside the unit circle. This model can be accommodated in our algorithm by extending the parameter vector \(\theta\), for example, as follows:

\[
\theta = [\rho^{(a)} T \rho^{(e)} T \omega^{(e)} T \omega^{(r)} T]^T \tag{5.6}
\]

where \(\rho^{(a)} = [\rho_0^{(a)} \cdots \rho_{m_a}^{(a)}]^T\) and \(\omega^{(a)} = [\omega_1^{(a)} \cdots \omega_{m_a}^{(a)}]^T\) with \(m_a = n_a / 2\), represent the radii and angles of the roots of \(A(q^{-1})\). Similarly, \(\rho^{(e)} = [\rho_1^{(e)} \cdots \rho_{m_e}^{(e)}]^T\) and \(\omega^{(e)} = [\omega_1^{(e)} \cdots \omega_{m_e}^{(e)}]^T\) with \(m_e = n_e / 2\), represent the radii and angles of the roots of \(C(q^{-1})\).

The polynomial coefficients derived from these roots are similarly collected into the extended coefficient vector \(\eta\) given by

\[
\eta = [a^T \ c^T]^T \tag{5.7}
\]

where \(a = [a_0 \cdots a_m]^T\) and \(c = [c_0 \cdots c_n]^T\).

The gradients derived in Section II can now be combined with the RPE algorithm for ARMA coefficient identification (see, e.g., [10]). The resulting algorithm for ARMA pole-zero estimation is given in Table II.

In the case of polar parametrization, computation of the coefficient vector \(\hat{\eta}(t)\) from the parameter vector \(\hat{\theta}(t)\) is accomplished by using recursion (A.6) to find the AR coefficients \(\hat{a}(t)\) from \(\hat{\rho}^{(a)}(t)\) and \(\hat{\omega}^{(a)}(t)\). The same recursion is then used to find the MA coefficients \(\hat{c}(t)\) from \(\hat{\rho}^{(e)}(t)\) and \(\hat{\omega}^{(e)}(t)\).

The Jacobian \(\xi(t) = \partial \hat{\eta}(t) / \partial \hat{\theta}^T(t)\) is a partitioned (2-block x 4-block) matrix with the \(i, j\) block \(\xi_{i,j}\) given by

\[
\xi_{1,i}(t) = \frac{\partial \hat{a}(t)}{\partial \rho^{(a)}(t)} \quad \xi_{1,j}(t) = \frac{\partial \hat{a}(t)}{\partial \omega^{(a)}(t)} \tag{5.8a}
\]

\[
\xi_{2,i}(t) = \frac{\partial \hat{c}(t)}{\partial \rho^{(e)}(t)} \quad \xi_{2,j}(t) = \frac{\partial \hat{c}(t)}{\partial \omega^{(e)}(t)} \tag{5.8b}
\]

and zeros in all other entries. The entries of the matrix \(\partial a / \partial \omega^{(a)}\) are calculated using recursion (2.14a) and the entries of \(\partial a / \partial \omega^{(e)}\) are calculated using (2.15a). Similarly, the entries of \(\partial c / \partial \omega^{(a)}\) are calculated using (2.14a) and those of \(\partial c / \partial \omega^{(e)}\) using (2.15a), but with coefficients \(c_i\) replacing \(a_i\). An example of the use of this algorithm is given in Section VI.

VI. SIMULATION RESULTS

The performances of the proposed algorithms were investigated by using them to identify the roots of several processes. Comparisons to previous algorithms are provided.

A. Speed of Convergence

The first process simulated was the autoregressive process

\[
s(t) = \frac{1}{A(q^{-1})} e(t) \tag{6.1a}
\]

where \(e(t)\) was zero-mean unit-variance white Gaussian noise and the polynomial \(A(q^{-1})\) was given by

\[
A(q^{-1}) = 1 - 0.6q^{-1} + 0.73q^{-2} - 0.44q^{-3} + 0.5256q^{-4} \tag{6.1b}
\]
TABLE II
ADAPTIVE ARMA POLE-ZERO ESTIMATION ALGORITHM

<table>
<thead>
<tr>
<th>Initialize $\hat{\theta}(0), \hat{\phi}(0), P(0), \psi(1), \varphi(1), \phi_r(1), w(1), w_r, w(\infty)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>For $i = 1, 2, \ldots, do$</td>
</tr>
<tr>
<td>$e(t) = y(t) - \psi(t) \hat{\theta}(t - 1)$</td>
</tr>
<tr>
<td>$L(t) = P(t - 1) \psi(t) / |w(t) + \psi(t) P(t - 1) \psi(t)|$</td>
</tr>
<tr>
<td>$P(t) = [P(t - 1) - L(t) \psi(t) P(t - 1)] / w(t)$</td>
</tr>
<tr>
<td>$\hat{\theta}(t) = \hat{\theta}(t - 1) + L(t) \psi(t)$</td>
</tr>
<tr>
<td>* Calculate $\hat{\phi}(t) = [\hat{\phi}(t) \hat{\phi}(t - 1)]^T$ from $\hat{\theta}(t)$</td>
</tr>
<tr>
<td>* Calculate $\hat{\psi}(t) = [\hat{\psi}(t) \hat{\psi}(t - 1)]^T$ from $\hat{\theta}(t)$</td>
</tr>
<tr>
<td>$\bar{\epsilon}(t) = y(t) - \psi(t) \hat{\theta}(t)$</td>
</tr>
<tr>
<td>$y(t) = y(t) - \sum_{i=1}^{n} \hat{\phi}_i t_i$</td>
</tr>
<tr>
<td>$g(t) = \bar{\epsilon}(t) - \sum_{i=1}^{n} \hat{\phi}_i t_i$</td>
</tr>
<tr>
<td>$\varphi(t + 1) = [-y(t), \ldots, -y(t - n_e + 1), \epsilon(t), \ldots, \epsilon(t - n_e + 1)]^T$</td>
</tr>
<tr>
<td>$\psi(t + 1) = [-y(t), \ldots, -y(t - n_e + 1), \epsilon(t), \ldots, \epsilon(t - n_e + 1)]^T$</td>
</tr>
<tr>
<td>$w(t + 1) = w(\infty) - (w(\infty) - w(t))w_r$</td>
</tr>
</tbody>
</table>

This is the same process investigated by Orfanidis and Vail [3], and has poles located at

\[
\rho_L e^{\pm j\theta} = -0.3 \pm 0.8 \quad \text{(6.1c)}
\]

\[
\rho_R e^{\pm j\phi} = 0.6 \pm 0.6. \quad \text{(6.1d)}
\]

The algorithm given in Table I was used to identify this process. Initial conditions for the pole estimates were set with the radii at 0.5 and the positive angles equally distributed between 0 and $\pi$. Thus,

\[
\hat{\theta}(0) = [0.5, 0.5, \pi/3, 2\pi/3]^T
\]

and, correspondingly,

\[
\hat{\phi}(0) = [0, 0.25, 0, 0.0625]^T.
\]

The remaining initial values were: $w(1) = 0.95$, $w_r = 0.99$, $w(\infty) = 1.0$, $\varphi(1) = \psi(1) = 0$, and $P(0) = 100I$.

The results of the simulation, after transformation to rectangular coordinates, are plotted in Fig. 1. It can be seen that the algorithm converges more rapidly than the one described in [3]. This is due to the fact that we use a Gauss–Newton update with an exact expression for the gradient.

B. Statistical Performance

The statistical efficiency of the algorithm was demonstrated by performing Monte Carlo simulations; the experiment described in Section VI-A was repeated 32 times using different driving noise processes. The sample error variances obtained for the radius and angle estimates are shown in Fig. 2. The Cramer–Rao bound, calculated using (4.9), is also plotted. As expected, the variances tend asymptotically to the Cramer–Rao bound. No outliers were observed in the experiments.

C. Tracking Time-Varying Parameters

The tracking ability of the algorithm was tested on a second-order AR process driven by zero-mean unit-variance white noise. The poles of the process were located at a radius of 0.95 and angles of $\pm 1$ radian for the first 499 samples. At the 500th sample, the pole angles were moved instantaneously to $\pm 1.5$ radians. The initial parameter estimates were set at the true parameter values. All other initial values were the same as in Section VI-A except the forgetting factor which was chosen to have $w(1) = 0.95$, $w(\infty) = 0.98$, and $w_r = 0.99$, so that the algorithm remained sensitive to parameter changes. When the transient occurred, $w(t)$ had effectively reached its final value. The angle estimate from one realization is given in Fig. 3, showing that the estimate converged to the new parameter value in roughly 200 samples. Larger values of $w(\infty)$ gave estimates that showed less variance, but which converged slower. The reverse was true for smaller values of $w(\infty)$.

The sample mean of 64 realizations of these simulations is also shown in Fig. 3. In order to compare the rate of convergence of the method proposed here with conventional pole estimation, the same realizations were identified using the recursive least squares (RLS) algorithm to estimate the AR polynomial. This was factorized at every update using Müller’s method. The forgetting factors were the same for both algorithms. The results from the two different identification methods were indistinguishable, showing that the proposed identification algorithm is able to track time-varying parameters at the same rate as conventional pole estimation methods, but requires significantly less computation, as discussed in Section VI-D.

D. Numerical Efficiency

As mentioned above, one of the main advantages of the proposed algorithm is its computational efficiency. To compare the numerical efficiency of the proposed algorithm to more conventional methods, the parameters of the process described in Section VI-A were estimated on a sample-by-sample basis on a personal computer with 12-digit precision. Using the method proposed in Table I, the average time per update was 136 ms. The same process was also identified using the Recursive Least Squares algorithm to update the polynomial coefficients. With each update, Müller’s method (as coded in [9, pp. 122–124]) was used to factorize the latest polynomial estimate. In this case, the average update time is dependent on the required accuracy because Müller’s method is used to calculate roots individually, and is iterated until each root estimate satisfies an accuracy criterion. The average time (in seconds) per update as a function of accuracy is given in Table III where it can be seen that, even for the most inaccurate case, the conventional method is at least seven
Fig. 1. Pole estimates using the algorithm in Table I for the AR process (6.1b). Solid lines indicate simulation results and dotted lines indicate true values.

Fig. 2. Estimation error variances for the process (6.1b). Crosses indicate simulation results and solid lines show the Cramer-Rao bound. Variances of angle estimates are given in radians squared.

Fig. 3. Pole angle estimate in radians and average behavior from 64 realizations for the time-varying second-order process of Section VI-C.
times slower than the method proposed in this paper. In the table, the accuracy is \( \epsilon_1 = \epsilon_2 \) as defined in Algorithm 3.10 of [9]. Note that, while the accuracy of the two-stage method is fixed by the choice of \( \epsilon \), the proposed adaptive pole estimation algorithm will always achieve the CRB asymptotically.

Table III shows that conventional factorization results in computation times longer than those obtained using the algorithm proposed here. This is a consequence of deflation—the process of reducing the polynomial's order by removing the estimated root. Deflation must be performed to ensure that the same root is not found more than once. Before a root is removed by deflation, it should be found with great accuracy (which usually involves several iterations of the estimation process) because, otherwise, the resulting deflated polynomial coefficients will be inaccurate. Since the roots are very sensitive to perturbations in the coefficients, this can result in the roots of the deflated polynomial being grossly inaccurate. Deflation can be used to reduce this effect, but requires more computation. In the absence of deflation, even a slight error in the computation of the first few roots of a high-order polynomial will result in the remaining roots being severely displaced. Thus, one has to use a sufficient number of iterations to ensure that the root to be extracted is estimated accurately enough, which is equivalent to choosing a small value of \( \epsilon_1 \) or \( \epsilon_2 \). This effect is not present when all roots are updated simultaneously as proposed in this paper.

\[ \rho_2 e^{z_{i=1} \omega} = 0.9 e^{2\pi/3}, \quad \gamma_1 = 0.9, \quad \gamma_2 = 0.7 \] (6.2)

and the process was driven by zero-mean unit-variance white Gaussian noise. The initial values used in the identification were the same as those used in Section VI-A, except that the parameter vector estimate was initialized with zeros.

The results of a typical simulation are shown in Fig. 5. The plots show the pole estimates computed (analytically) from the second-order section coefficient estimates. From examination of the angle estimate for the complex-conjugate pair, it is apparent that for the first few samples, these poles were estimated as a real pair. However, the algorithm rapidly corrected itself, yielding the proper complex-conjugate estimate. Fig. 5 also shows the estimates of the real poles.

\section*{Resolution}

A fourth-order AR process with closely spaced poles was simulated for the purpose of demonstrating the resolution of the algorithm. All parameters and initial conditions were the same as described in Section VI-A except that the poles of the process had radii of 0.95 and angles of \( \pm 0.98 \) and \( \pm 1.02 \) radians. Fig. 4 shows the angle estimates obtained in a typical simulation, demonstrating that the algorithm can correctly identify closely spaced poles.

\section*{Processes with Unknown Proportion of Real and Complex Poles}

A fourth-order AR process with two real poles and a pair of complex-conjugate poles was used to test the identification method proposed in Section V-A which, as discussed previously, does not require \( a \) priori knowledge of the relative number of real and complex poles. The poles were located at

\[ \rho_1 e^{z_{i=1} \omega} = 0.9 e^{2\pi/3}, \quad \gamma_1 = 0.9, \quad \gamma_2 = 0.7 \] (6.2)

and the process was driven by zero-mean unit-variance white Gaussian noise. The initial values used in the identification were the same as those used in Section VI-A, except that the parameter vector estimate was initialized with zeros.

The results of a typical simulation are shown in Fig. 5. The plots show the pole estimates computed (analytically) from the second-order section coefficient estimates. From examination of the angle estimate for the complex-conjugate pair, it is apparent that for the first few samples, these poles were estimated as a real pair. However, the algorithm rapidly corrected itself, yielding the proper complex-conjugate estimate. Fig. 5 also shows the estimates of the real poles.

\section*{Identification of ARMA Roots}

The algorithm described in Table III was tested by using it to identify the ARMA(4, 4) process studied by Cadzow [22]. The process obeyed the model (5.5) where \( e(t) \) was
zero-mean unit-variance white Gaussian noise and
\[ A(q^{-1}) = 1 + 0.1q^{-1} + 1.66q^{-2} + 0.093q^{-3} 
+ 0.8649q^{-4} \] 
(6.3a)
\[ C(q^{-1}) = 1.6814 + 0.038q^{-1} + 1.3745q^{-2} 
+ 0.0109q^{-3} + 0.1284q^{-4} \]  
(6.3b)
This process has poles at 0.9644 \( \exp(\pm 1.3619i) \) and 0.9644 \( \exp(\pm 1.8330i) \). The zeros are located at 0.8425 \( \exp(\pm 1.5802i) \) and 0.3280 \( \exp(\pm 1.5811i) \).

The initial conditions were: \( w(1) = 0.97, w_0 = 0.995 \), \( w(\infty) = 1.0, P(0) = 0.1I \), and
\[ \hat{\theta}(0) = [0.5, 0.5, 0.5, 0.5, \pi/3, 2\pi/3, \pi/3, 2\pi/3] \]  
with corresponding initial coefficient vector
\[ \hat{\eta}(0) = [0, 0.25, 0, 0.0625, 0, 0.25, 0, 0.0625] \]  

The results of a typical experiment are shown in Fig. 6. As is typical for ARMA estimation (see, e.g., [10]), the AR parameters—the poles in this case—converge more rapidly than the MA parameters.
VII. CONCLUSION

A new recursive algorithm has been presented for direct estimation of the poles of autoregressive processes. It has been extended for root identification of more general models such as pole-zero estimation of ARMA models. Exactly analogous extensions are possible for dealing with ARMAX models. The algorithm can also be used for the estimation of AR poles in additive noise by combining the results of this paper with those of [23]. A simple method was presented for dealing with systems where the relative number of real and complex poles is unknown.

The algorithm is computationally efficient, requiring \( O(n^2) \) operations per update for an \( n \)-th order system. It exhibits fast convergence due to the use of the Gauss-Newton approach with an exact expression for the gradient. Numerical simulations were presented to demonstrate the algorithm’s computational efficiency and convergence rate.

The method is useful for both stationary and nonstationary processes. For stationary processes, it is useful in applications requiring on-line estimates which become increasingly more accurate as each data sample is received, while simultaneously satisfying the constraints of using fixed memory length and computationally efficient updating procedures. For nonstationary processes, the algorithm is useful for tracking pole movements on line.

A closed-form expression for the CRB on the pole estimates was derived in the paper and simulation results confirm that the algorithm attains the bound asymptotically.

It was shown that, subject to minor constraints, the algorithm converges to a global minimum of the error cost function which corresponds to the true roots of the AR polynomial. This is a surprising, but pleasing, property of the highly nonlinear parametrization. The existence in the cost function of multiple true minima ensures rapid convergence of the algorithm.

APPENDIX A

MAPPING POLYNOMIAL ROOTS TO COEFFICIENTS

The fundamental relationship between the polynomial coefficients \( a_i, i = 0, \ldots, n \), and the roots \( \lambda_i, i = 1, \ldots, n \), is given by (1.2) for arbitrary \( q \). This relationship can also be written in terms of the shift matrix \( Z \) which consists of ones along the principal subdiagonal and zeros elsewhere

\[
Z = \begin{bmatrix}
0 & & & \\
& 1 & & \\
& & \ddots & \\
& & & 1
\end{bmatrix}, \quad (n+1) \times (n+1). \quad (A.1)
\]

In this case, (1.2) becomes

\[
\sum_{i=0}^{n} a_i Z^i = \prod_{i=1}^{n} (I_{n+1} - \lambda_i Z). \quad (A.2a)
\]

or, equivalently,

\[
T = \prod_{i=1}^{n} \Phi(\lambda_i) \quad (A.2b)
\]

where \( T \) is the lower triangular Toeplitz matrix defined as

\[
T = \sum_{i=0}^{n} a_i Z^i = \begin{bmatrix}
1 & & & \\
& a_1 & 1 & \\
& & \ddots & \\
& & & a_n
\end{bmatrix} \quad (A.2c)
\]

and \( \Phi(\lambda_i) \) is an \((n+1) \times (n+1)\) lower bidiagonal Toeplitz matrix given by

\[
\Phi(\lambda_i) = I_{n+1} - \lambda_i Z = \begin{bmatrix}
1 & & & \\
& -\lambda_i & 1 & \\
& & \ddots & \\
& & & -\lambda_i
\end{bmatrix}. \quad (A.2d)
\]

An efficient method of computing the \( a_i \)'s can be obtained because the matrices \( \Phi(\lambda_i) \) are sparse and Toeplitz. Let the first \( j \) terms in the product (A.2b) be denoted by

\[
T^{(j)} = \prod_{i=1}^{j} \Phi(\lambda_i) = T^{(j-1)} \Phi(\lambda_j) \quad (A.3a)
\]

\[
= T^{(j-1)} - \lambda_j T^{(j-1)} Z \quad (A.3b)
\]

where (A.2d) was used to expand \( \Phi(\lambda_j) \). Let the entries in the first column of \( T^{(j)} \) be \( a^{(j)}_i \). Equating individual entries on both sides of (A.3b) gives the recursion

\[
a^{(j)}_i = a^{(j-1)}_i - \lambda_j a^{(j-1)}_{i-1} \quad (A.4)
\]

for \( 1 \leq i \leq j \) and \( 1 \leq j \leq n \) with boundary conditions \( a^{(0)}_0 = 1 \) and \( a^{(j)}_0 = 0 \) for all other \( i, j \). Note that this recursion requires the use of complex arithmetic.

More efficient recursions can be found for constrained systems. For example, in polar coordinates, the equivalent relationship to (A.2a) is

\[
\sum_{i=0}^{n} a_i Z^i = \prod_{i=1}^{n/2} (I_{n+1} - 2\rho \cos(\omega_i) Z + \rho^2 Z^2). \quad (A.5)
\]

In this case, using the same approach as above, the following second-order recursion is obtained:

\[
a^{(j)}_i = a^{(j-1)}_i - 2\rho a^{(j-1)}_{i-1} \cos(\omega_j) + \rho^2 a^{(j-1)}_{i-2} \quad (A.6)
\]

for \( 1 \leq i \leq 2 \) and \( 1 \leq j \leq n/2 \) with boundary conditions \( a^{(0)}_0 = 1 \) and \( a^{(j)}_0 = 0 \) for all other \( i, j \). Calculation of the coefficients from the roots with this recursion requires approximately \( n^2 \) flops.

APPENDIX B

GRADIENT OF COEFFICIENTS WITH RESPECT TO ROOTS

This appendix presents a closed-form expression for the derivative of a polynomial’s coefficients with respect to
its roots. Define the extended vector \( \bar{a} \) of polynomial coefficients
\[
\bar{a} = [1 \ a^T]^T
\]  
(B.1)
corresponding to the first column of \( T \) in (A.2c). Suppose that the polynomial has \( p \) roots (complex or real) \( \lambda_1, \lambda_2, \ldots, \lambda_p \) with positive multiplicities \( \mu_1, \mu_2, \ldots, \mu_p \) such that
\[
n = \sum_{i=1}^{p} \mu_i.
\]  
(B.2)
From (A.2b) and (A.2c), the extended coefficient vector \( \bar{a} \) can be written in the form
\[
\bar{a} = \left[ \prod_{i=1}^{p} [\Phi(\lambda_i)]^{\mu_i} \right] e_1
\]  
(B.3a)
where
\[
e_1 = [1, 0, \ldots, 0]^T; \quad (n + 1) \times 1.
\]  
(B.3b)
Differentiating (B.3a) with respect to the root \( \lambda_i \) gives
\[
\frac{\partial \bar{a}}{\partial \lambda_i} = \frac{\partial}{\partial \lambda_i} \left( [\Phi(\lambda_i)]^{\mu_i} \right) \left[ \prod_{i=1}^{p} [\Phi(\lambda_i)]^{\mu_i} \right] e_1
\]  
(B.4a)
where, from (A.2d)
\[
\frac{\partial}{\partial \lambda_i} \left( [\Phi(\lambda_i)]^{\mu_i} \right) = -\mu_iZ[\Phi(\lambda_i)]^{\mu_i-1}.
\]  
(B.4b)
Substituting (B.4b) into (B.4a) and making use of definition (B.3a) leads to the result
\[
\frac{\partial \bar{a}}{\partial \lambda_i} = -\mu_iZ[\Phi(\lambda_i)]^{\mu_i-1} \bar{a}.
\]  
(B.5)
The inverse of \( \Phi(\lambda_i) \) is given by
\[
[\Phi(\lambda_i)]^{-1} = \begin{bmatrix}
1 & & \\
\lambda_i & \ddots & \\
& \ddots & \ddots \\
& & \lambda_i & \ddots & 1
\end{bmatrix}
\]  
(B.6)
Using this expression and rearranging terms leads to
\[
[\Phi(\lambda_i)]^{-1} \bar{a} = H\bar{v}(\lambda_i)
\]  
(B.7a)
where
\[
H = \begin{bmatrix}
1 & & \\
& \ddots & a_1 \\
& & \ddots \\
1 & a_1 & \cdots & a_p
\end{bmatrix}
\]  
(B.7b)
\[
\bar{v}(\lambda_i) = [\lambda_i^0, \ldots, \lambda_i, 1]^T.
\]  
(B.7c)
Using (B.7a), the derivative (B.5) can be written in the following compact form:
\[
\frac{\partial \bar{a}}{\partial \lambda_i} = -Z\bar{H}v(\lambda_i) \mu_i.
\]  
(B.8)
The desired derivative matrix is obtained from (B.8) by deleting the top row on both sides of the equation and by appending columns corresponding to all other roots giving
\[
\frac{\partial \bar{a}}{\partial \lambda_i} = -HVM; \quad n \times p
\]  
(B.9a)
where the Hankel matrix \( H \) and the Vandermonde matrix \( V \) are defined in Section III and \( M \) is a matrix of multiplicities
\[
M = \text{diag} \{ \mu_1, \ldots, \mu_p \}.
\]  
(B.9b)
For simple roots, the result (B.9a) is identical to the one derived in [16] and [17]. However, the expression in this paper is valid for polynomials with multiple roots and was obtained using a more direct approach.

Note that the gradient \( \partial a / \partial \lambda_i \) in (B.9a) is unchanged if the columns of \( H \) and the rows of \( V \) are reversed. Thus,
\[
\frac{\partial a}{\partial \lambda_i} = -A_i\Lambda M; \quad n \times p
\]  
(B.10a)
where \( A_i = HJ \) was defined in (4.4b), \( \Lambda = JV \), and \( J \) is the \( (n \times n) \) permutation matrix which consists of ones along the principal antidiagonal and zeros elsewhere.


**Appendix C**

**Indirect Pole Estimation by Coefficient Matching**

This appendix presents an alternative pole estimation algorithm based on coefficient matching. The method performs a two-step update with the reception of each data sample. First, the polynomial coefficients are updated using the standard recursive least squares (RLS) identification algorithm. Then, using the new coefficient estimates, a gradient method is used to update the pole estimates so as to minimize the following criterion function:
\[
V(t) = \epsilon^T(t) \epsilon(t)
\]  
(C.1)
where \( \epsilon(t) \) is an error vector defined by
\[
\epsilon(t) = \hat{a}(t) - \dot{a}(t).
\]  
(C.2)
In this case, \( \hat{a} \) is the estimate of polynomial coefficients obtained from the conventional RLS algorithm and \( \dot{a} \) is the coefficient vector estimate obtained from the pole estimate \( \theta(t) \) via (A.6).

Let \( \theta(t) \) be the \( i \)th iteration of the algorithm during sampling interval \( t \). If \( p \) iterations are performed during each sampling interval (typically \( p = 1 \)), then the estimates across sample boundaries are related by
\[
\theta(i+1) = \theta(i+1) \theta(t).
\]  
(C.3)
However, the following discussion concerns computations which all take place during the same sampling interval, so the time index $t$ can be dropped for notational simplicity.

The steepest descent algorithm used for the minimization of (C.1) is then
\[
\theta(t+1) = \theta(t) - \mu \frac{\partial V}{\partial \theta} |_{\theta = \theta(t)}, \tag{C.4}
\]
where $\mu \ll 1$ is the step size. Using the chain rule, the derivative above can be rewritten
\[
\frac{\partial V}{\partial \theta^2} = \frac{\partial V}{\partial \hat{\theta}} \frac{\partial \hat{\theta}}{\partial \theta} = \epsilon^T (-I) \frac{\partial \hat{\theta}}{\partial \theta}, \tag{C.5}
\]
Thus, substituting (C.5) into (C.4), the pole update is
\[
\theta(t+1) = \theta(t) + \mu \left( \frac{\partial \hat{\theta}}{\partial \theta(t)} \right) \epsilon \tag{C.6}
\]
where $\partial \hat{\theta}/\partial \theta$ is computed with the results of iteration $k$ using the recursive expressions derived in the body of the paper.

ACKNOWLEDGMENT
The authors thank Prof. B. Porat and Prof. P. Stoica for discussions concerning the subject matter of this paper.

REFERENCES