Reports and Technical Reports   All Technical Reports Collection

1993-09-15

An iterative extension of Prony's method for ARMA signal modeling

Therrien, Charles W.; Velasco, Carlos H.

Monterey, California. Naval Postgraduate School

https://hdl.handle.net/10945/28754

NAVAL POSTGRADUATE SCHOOL
Monterey, California

An Iterative Extension of Prony's Method
for ARMA Signal Modeling

by

Charles W. Therrien
Carlos H. Velasco

September 15, 1993

Undersea Warfare Center
ew London

Naval Postgraduate School
Monterey, California 93943-5000

Rear Admiral T. A. Mercer
Superintendent

H. Shull
Provost

Reproduction of all or part of this report is authorized.

This report was prepared by:

Computer Engineering

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 15, 1993	3. REPORT TYPE AND DATES COVERED Technical Report

4. TITLE AND SUBTITLE

An Iterative Extension of Prony's Method
for ARMA Signal Modeling

5. FUNDING NUMBERS

6. AUTHOR(S)

Charles W. Therrien and Carlos H. Velasco

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

NPSEC-93-025

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Undersea Warfare Center
Det. New London
New London, CT 06320-5594

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this report are those of the authors and do not reflect the
official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A new iterative version of the Prony method is presented and shown to be exceptionally effective in finding ARMA models for acoustic data in the signal domain. The method is based on a quadratic type of gradient algorithm where it is shown that the gradient and Hessian are easily computed from the data. The new algorithm is found experimentally to have excellent convergence behavior. The performance of the algorithm is demonstrated and compared to that of the basic Prony method and to that of the Steiglitz and McBride iterative prefiltering algorithm on some recorded acoustic data.

14. SUBJECT TERMS
Prony's method, ARMA modeling

15. NUMBER OF PAGES
25

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR

An Iterative Extension of Prony's Method for ARMA Signal Modeling*

Charles W. Therrien and Carlos H. Velasco
Department of Electrical and Computer Engineering

Abstract

A new iterative version of the Prony method is presented and shown to be exceptionally effective in finding ARMA models for acoustic data in the signal domain. The method is based on a quadratic type of gradient algorithm where it is shown that the gradient and Hessian are easily computed from the data. The new algorithm is found experimentally to have excellent convergence behavior. The performance of the algorithm is demonstrated and compared to that of the basic Prony method and to that of the Steiglitz and McBride iterative prefiltering algorithm on some recorded acoustic data.

1 Introduction

Linear pole-zero (ARMA) models are used for a variety of applications in signal processing. These applications include speech and image coding, spectrum estimation, and others. Both deterministic and stochastic methods for signal modeling have been studied extensively in the literature. Typically deterministic models attempt to represent the signal as the impulse response of a suitably chosen linear system while stochastic methods attempt to model the signal as the response of the linear system to white noise. However in many cases the procedures embedded in these methods (such as estimating a correlation matrix for the data) are similar so the philosophical distinction between

---

deterministic and stochastic models is of less significance. A review of various methods can be found in many places such as [1, 2, 3].

In our application we are interested in reproducing the time domain waveform as accurately as possible (as measured e.g., by the sum of squared errors). To do this frequently requires modeling procedures that can produce nonminimum-phase transfer functions [4, 5] and this eliminates most of the stochastic methods. Consequently, it is most appropriate here to think of the signal as being modeled as the impulse response of a linear system and seek to minimize the sum of squared errors between this impulse response and the given data.

A well known approach to the modeling problem is Prony's method, where the data is represented in the time domain by a sum of damped exponentials with appropriate weighting coeffients. Prony's method avoids the direct nonlinear problem of minimizing the sum of squared errors and solves a pair of linear problems. However in many cases involving real data this results in a suboptimal model as shown for the acoustic data (corresponding to the ringing of a wrench hit on the floor) in Fig. 1(a). In this paper we present a method for iteratively adjusting the parameters (poles and zeros) in Prony's method to values that further minimize the sum of squared errors. The method, which we call the *iterative Prony method*, produces a much improved model as shown in Fig. 1(b).

Iterative methods for ARMA modeling are not new [6, 7, 8, 9, 10]. Most stochastic methods involving maximum likelihood solutions involve iteration (see e.g., [6]) and a very effective method known as *iterative prefiltering* [7, 8] is capable of results similar to the iterative Prony method. The latter, like many other methods (e.g., [9]), uses the polynomial coefficients as the parameters and thus does not have a clear interpretation in terms of placement of poles and zeros. Iterative prefiltering can also lead to oscillatory behavior in the sum of squared errors while the iterative Prony method typically results in a monotonic decrease of the error norm and seems to have better behavior when the model order is not known.

The remainder of this paper is organized as follows. Section 2 states the basic form of the modern Prony method and establishes our notation. Section 3 develops the iterative Prony algorithm. Section 4 gives an example and comparison of performance on recorded acoustic data. Section 5 provides a summary and conclusions.
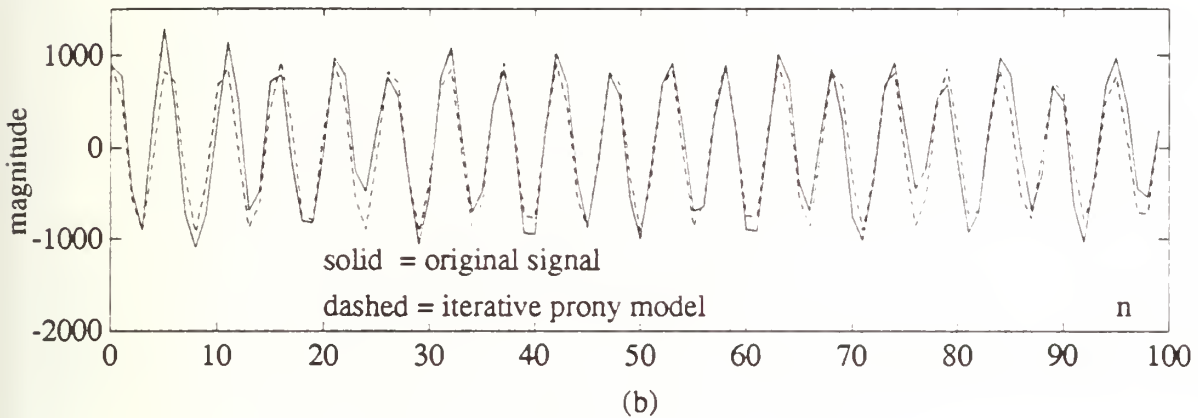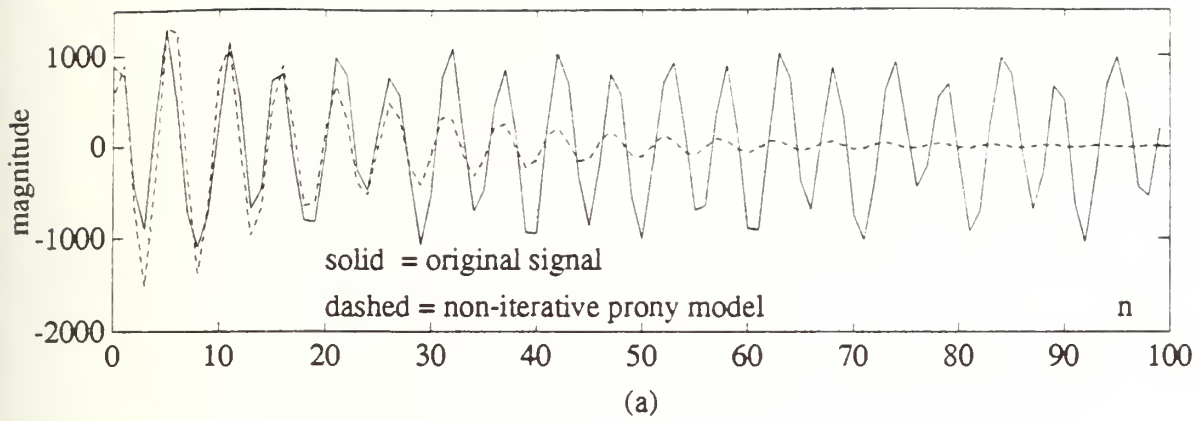
Figure 1: Approximately 10 ms segment of the sound corresponding to the ringing of a wrench hit on the floor. (a) Model produced by the (non-iterative) Prony method (4 poles and 3 zeros). (b) Model produced by the iterative Prony method (also 4 poles and 3 zeros).

2 Prony's Method of ARMA Modeling

Although there are many variations (see [11, Chapter 11]), the modern Prony method can be thought of as a method for approximating a data sequence $x[n]$ by a sequence $\check{x}[n]$ which is the impulse response of a rational linear system satisfying the difference equation

$$\check{x}[n] + a_1\check{x}[n-1] + \cdots + a_P\check{x}[n-P] = b_0\delta[n] + b_1\delta[n-1] + \cdots + b_Q\delta[n-Q]$$
(1)

If the requirement $\check{x}[n] = x[n]$; $\quad n = 0, 1, \ldots, N_s - 1$ is applied to (1), where N_s is the length of the data, and the difference equation is evaluated for $n = 0, 1, \ldots, N_s - 1$, the result is the matrix equation

$$\begin{bmatrix} x[0] & 0 & 0 & \cdots & 0 \\ x[1] & x[0] & 0 & \cdots & 0 \\ x[2] & x[1] & x[0] & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x[Q] & x[Q-1] & x[Q-2] & \cdots & x[Q-P] \\ x[Q+1] & x[Q] & x[Q-1] & \cdots & x[Q-P+1] \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x[N_s-1] & x[N_s-2] & x[N_s-3] & \cdots & x[N_s-P-1] \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_P \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_Q \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}$$
(2)

which can be written in partitioned form as

$$\begin{bmatrix} \mathbf{X}_B \\ \mathbf{X}_A \end{bmatrix} \mathbf{a} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}$$
(3)

Thus if $N_s \geq P + Q + 1$ a least squares solution for the a_i (AR) parameters can be found from the lower set of partitions, and the upper set of partitions can be used to find the b_j (MA) parameters. This is the fundamental idea underlying Prony's method.

Prony's method is frequently expressed directly in the signal domain. Instead of solving (3) for the vector $\mathbf{b}$, we can instead find the roots of the denomimator polynomial of the system in (1)

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_P z^{-P}$$
(4)

4

and express the impulse response $\check{x}[n]$ in terms of these roots. Specifically if $r_1, r_2, \ldots, r_P$ are the roots of (4) (assumed to be distinct) then $\check{x}[n]$ can be written as

$$\check{x}[n] = c_1 r_1^n + c_2 r_2^n + \cdots + c_P r_P^n \tag{5}$$

Again, if we require $\check{x}[n] = x[n]$; $\quad n = 0, 1, \ldots, N_s - 1$, then by evaluating (5) for $n = 0, 1, \ldots, N_s - 1$ we have the matrix equation

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ r_1 & r_2 & \cdots & r_P \\ r_1^2 & r_2^2 & \cdots & r_P^2 \\ \vdots & \vdots & \cdots & \vdots \\ r_1^{N_s-1} & r_2^{N_s-1} & \cdots & r_P^{N_s-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{bmatrix} = \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N_s - 1] \end{bmatrix} \tag{6}$$

which can be solved in a least squares sense to obtain the coefficients c_i. This implementation of Prony's method has an advantage over simply solving for **b** from (3) since all of the data is used in the computation. When (2) or (3) is used to find the b_j, only the data values from 0 to Q are used in the computation.

In the case of multiple roots at the same location a slight variation of the same procedure can be used. Suppose, for example, that r_1 is a double root. In this case, $\check{x}[n]$ has the form

$$\check{x}[n] = c_1 r_1^n + c_2 n r_1^n + \cdots + c_P r_P^n \tag{7}$$

and the matrix equation to solve for the coefficients becomes

$$\begin{bmatrix} 1 & 0 & 1 & \cdots & 1 \\ r_1 & r_1 & r_3 & \cdots & r_P \\ r_1^2 & 2r_1^2 & r_3^2 & \cdots & r_P^2 \\ \vdots & \vdots & & \cdots & \vdots \\ r_1^{N_s-1} & (N_s - 1)r_1^{N_s-1} & r_3^{N_s-1} & \cdots & r_P^{N_s-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{bmatrix} = \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N_s - 1] \end{bmatrix} \tag{8}$$

This situation is rare however, because computational errors and errors inherent to the modeling method itself contribute to produce roots that may be very close to each other but not exactly at the same location.

While Prony's method represents a clever scheme to separate the solution of the AR and MA parameters in the modeling problem, the separation is frequently achieved at the cost of a degradation in performance, as mentioned in the introduction. It is well known, for example, that Prony's method does *not* choose the AR parameters to minimize any

norm of the error $e[n] = x[n] - \tilde{x}[n]$ but rather chooses them to minimize the norm of a modified error of linear prediction resulting from processing the original data $x[n]$ through the filter represented by $A(z)$. As a result, there is motivation to modify the model obtained by Prony's method through an iterative scheme that seeks to directly minimize the norm of the error $e[n]$. This method retains the separability of the original Prony method and so is called the *iterative Prony method*. It differs from maximum likelihood in that there is no stochastic criterion and it does not attempt to solve for the AR and MA parameters simultaneously. The method is developed below.

3 An Iterative Prony Algorithm

3.1 Iterative Algorithms

A multidimensional function $\mathcal{Q}(r_1, r_2, \cdots, r_P)$ that is continuous and differentiable can be minimized using one of several very powerful optimization techniques known as *gradient* methods [12]. Some of those methods are derived on the basis of a quadratic model that can be obtained from a truncated Taylor series expansion of $\mathcal{Q}(\mathbf{r})$. Let $\mathbf{r}^{(k)}$ denote the value of $\mathbf{r}$ at the kth iteration. Then for any point $\mathbf{r} = \mathbf{r}^{(k)} + \delta$ when δ is small, the function can be approximated by

$$\mathcal{Q}(\mathbf{r}^{(k)} + \delta) \approx q^{(k)}(\delta) = \mathcal{Q}^{(k)} + \mathbf{g}^{(k)^T}\delta + \frac{1}{2}\delta^T \mathbf{G}^{(k)}\delta \qquad (9)$$

where $\mathbf{g}$ and $\mathbf{G}$ represent the vector of first derivatives (gradient) and the matrix of second derivatives (Hessian) of the function $\mathcal{Q}(\mathbf{r})$ and they should be available at every point. In *Newton's* method the iterate $\mathbf{r}^{(k+1)}$ is taken to be $\mathbf{r}^{(k)} + \delta^{(k)}$, where the correction $\delta^{(k)}$ minimizes $q^{(k)}(\delta)$. This method is only well defined when the Hessian matrix $\mathbf{G}$ is positive definite, in which case the kth iteration of Newton's method is given by the following procedure [13]:

$$\begin{array}{ll} \text{1. solve } \mathbf{G}^{(k)}\delta = -\mathbf{g}^{(k)} \quad \text{for} \quad \delta = \delta^{(k)} \\ \text{2. set } \mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \delta^{(k)} \end{array} \qquad (10)$$

The fact that $\mathbf{G}^{(k)}$ may not be positive definite when $\mathbf{x}^{(k)}$ is far from the solution, and that even when $\mathbf{G}^{(k)}$ is positive definite convergence may not occur, makes this method undesirable as a general formulation of a minimization algorithm.

Since Newton's method is defined only when the matrix $\mathbf{G}^{(k)}$ is positive definite, and this matrix is positive definite only when the error δ is "small", we can say that the

method is defined only in some neighborhood $\Omega^{(k)}$ of $\mathbf{r}^{(k)}$ in which $q^{(k)}(\delta)$ agrees with $\mathcal{Q}(\mathbf{r}^{(k)} + \delta)$ in some sense. In such cases, it is correct to choose $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \delta^{(k)}$, with the correction $\delta^{(k)}$ minimizing $q^{(k)}(\delta)$ for all $\mathbf{r}^{(k)} + \delta$ in $\Omega^{(k)}$. This method is referred to as the *restricted step method* because the step is restricted by the region of validity of the Taylor series [13]. The region of definition for the kth iteration can then be expressed as

$$\Omega^{(k)} = \left\{ \mathbf{r} : \| \mathbf{r} - \mathbf{r}^{(k)} \| \leq h^{(k)} \right\} \tag{11}$$

where $\| \cdot \|$ denotes the norm of the vector. In this case, the optimization problem can be stated as:

$$\text{minimize}_\delta \ q^{(k)}(\delta) \text{ subject to } \| \delta \| \leq h^{(k)} \tag{12}$$

As mentioned before, the least squares norm $\| \cdot \|_2$ is the one most commonly used in these type of problems so it is the one used in this paper. The problem that now becomes apparent is how to select the error margin $h^{(k)}$ of the neighborhood (11). This margin should be as large as possible because the iteration step is directly related to it. Various methods have been proposed to control the parameter $h^{(k)}$. One of these methods [13] attempts to insure that the Newton's minimization problem (10) is always defined by adding a multiple of the unit matrix $\mathbf{I}$ to $\mathbf{G}^{(k)}$ and computing the new problem

$$\left(\mathbf{G}^{(k)} + \nu \mathbf{I} \right) \delta^{(k)} = -\mathbf{g}^{(k)} \tag{13}$$

where the net effect is that increases in ν cause $\| \delta \|_2$ to decrease, and vice versa.

If we define

$$\rho^{(k)} = \frac{\Delta \mathcal{Q}^{(k)}}{\Delta q^{(k)}} = \frac{\mathcal{Q}^{(k)} - \mathcal{Q}(\mathbf{r}^{(k)} + \delta^{(k)})}{\mathcal{Q}^{(k)} - q^{(k)}(\delta^{(k)})} \tag{14}$$

then the ratio $\rho^{(k)}$ represents a measure of the accuracy to which $q^{(k)}(\delta^{(k)})$ approximates $\mathcal{Q}(\mathbf{r}^{(k)} + \delta^{(k)})$ on the kth step, and as the accuracy increases $\rho^{(k)}$ gets closer to unity. Using (14), Marquardt [14] suggested an algorithm that tries to adaptively maintain $h^{(k)}$ as large as possible while controlling the ratio $\rho^{(k)}$. The kth iteration of such an algorithm is stated as

1. given $\mathbf{r}^{(k)}$ and $\nu^{(k)}$, calculate $\mathbf{g}^{(k)}$ and $\mathbf{G}^{(k)}$;

2. factor $\mathbf{G}^{(k)} + \nu^{(k)}\mathbf{I}$; if not positive definite, reset $\nu^{(k)} = 4\nu^{(k)}$ and repeat;

3. solve (13) to find $\delta^{(k)}$;

4. evaluate $\mathcal{Q}(\mathbf{r}^{(k)} + \delta^{(k)})$ and hence $\rho^{(k)}$;

5. if $\rho^{(k)} < 0.25$ set $\nu^{(k+1)} = 4\nu^{(k)}$
   else if $\rho^{(k)} > 0.75$ set $\nu^{(k+1)} = \nu^{(k)}/2$
   else set $\nu^{(k+1)} = \nu^{(k)}$;

6. if $\rho^{(k)} \leq 0$ set $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)}$ else set $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \boldsymbol{\delta}^{(k)}$.

Here the parameters 0.25, 0.75, 4 and 2 are arbitrary and $\nu^{(1)} > 0$ is also chosen arbitrarily. Proofs of global and second order convergence for this algorithm are given in [13] for the cases when the first and second derivatives of the function $Q(\mathbf{r})$ exist, and the vector $\mathbf{r}^{(k)}$ belongs to a bounded n-dimensional space for all k. Although this method does have some disadvantages, it represents a good basis for the formulation of a general minimization algorithm.

3.2 Computation of Derivatives

Let us now return to the problem of representing a sequence $x[n]$ as a linear combination of complex exponentials. Equation 6 can be written as

$$\mathbf{Rc} = \mathbf{x} + \epsilon \tag{15}$$

where ϵ is the equation error, $\mathbf{x}$ represents the data, which may or may not be complex, $\mathbf{c}$ is the vector of complex coefficients, and $\mathbf{R}$ is the matrix of complex roots, which can be written in terms of real and imaginary parts as

$$\mathbf{R} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ r_{R_1} + jr_{I_1} & r_{R_2} + jr_{I_2} & \cdots & r_{R_P} + jr_{I_P} \\ (r_{R_1} + jr_{I_1})^2 & (r_{R_2} + jr_{I_2})^2 & \cdots & (r_{R_P} + jr_{I_P})^2 \\ \vdots & \vdots & \cdots & \vdots \\ (r_{R_1} + jr_{I_1})^{N_s-1} & (r_{R_2} + jr_{I_2})^{N_s-1} & \cdots & (r_{R_P} + jr_{I_P})^{N_s-1} \end{bmatrix} \tag{16}$$

By defining

$$Q(\mathbf{r}) = \|\epsilon\|_2 = \epsilon^{*T}\epsilon = (\mathbf{Rc} - \mathbf{x})^{*T}(\mathbf{Rc} - \mathbf{x}) \tag{17}$$

it is clear that the problem is to find the vector $\mathbf{r} = \mathbf{r}_R + j\mathbf{r}_I$ of P complex roots that minimizes the function $Q(\mathbf{r})$.

To make the following development less cumbersome let us define the gradient operator with respect to the real and imaginary parts of the roots as

$$\nabla_{\mathbf{r}} \overset{\text{def}}{=} \begin{bmatrix} \nabla_{\mathbf{r}_R} \\ \nabla_{\mathbf{r}_I} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial r_{R_1}} \\ \frac{\partial}{\partial r_{R_2}} \\ \vdots \\ \frac{\partial}{\partial r_{R_P}} \\ \frac{\partial}{\partial r_{I_1}} \\ \frac{\partial}{\partial r_{I_2}} \\ \vdots \\ \frac{\partial}{\partial r_{I_P}} \end{bmatrix} \tag{18}$$

Then the gradient vector $\mathbf{g}$ of first derivatives and the Hessian matrix $\mathbf{G}$ of second derivatives can be written compactly as

$$\mathbf{g} = \nabla_{\mathbf{r}} \mathcal{Q} = \begin{bmatrix} \nabla_{\mathbf{r}_R} \mathcal{Q} \\ \nabla_{\mathbf{r}_I} \mathcal{Q} \end{bmatrix} \tag{19}$$

and

$$\mathbf{G} = \nabla_{\mathbf{r}} \left[(\nabla_{\mathbf{r}} \mathcal{Q})^T \right] = \begin{bmatrix} \nabla_{\mathbf{r}_R} (\nabla_{\mathbf{r}_R} \mathcal{Q})^T & \nabla_{\mathbf{r}_R} (\nabla_{\mathbf{r}_I} \mathcal{Q})^T \\ \nabla_{\mathbf{r}_I} (\nabla_{\mathbf{r}_R} \mathcal{Q})^T & \nabla_{\mathbf{r}_I} (\nabla_{\mathbf{r}_I} \mathcal{Q})^T \end{bmatrix} \tag{20}$$

Note that it is essential to work with the derivatives involving the real and imaginary parts of the roots rather than the complex roots themselves because $\mathcal{Q}(\mathbf{r})$ in (17) is not an analytic function of the complex variables $r_1, r_2, \ldots, r_P$. It is shown in the appendix that the upper and lower partitions of $\mathbf{g}$ are given by

$$\nabla_{\mathbf{r}_R} \mathcal{Q} = 2 \operatorname{Re} \left\{ \mathbf{F}^{*T} \boldsymbol{\epsilon} \right\} \tag{21}$$

and

$$\nabla_{\mathbf{r}_I} \mathcal{Q} = 2 \operatorname{Im} \left\{ \mathbf{F}^{*T} \boldsymbol{\epsilon} \right\} \tag{22}$$

where $\mathbf{F}$ is a matrix with columns $\mathbf{f}_i$ defined by

$$\mathbf{f}_i \overset{\text{def}}{=} \begin{bmatrix} 0 \\ 1 \\ 2r_i \\ 3r_i^2 \\ \vdots \\ (N_s - 1)r_i^{N_s - 2} \end{bmatrix} c_i \tag{23}$$

9

Further, if $\mathbf{S}$ is defined as the matrix with columns

$$\mathbf{s}_i \stackrel{\text{def}}{=} \begin{bmatrix} 0 \\ 0 \\ 2 \\ 6r_i \\ 12r_i^2 \\ \vdots \\ (N_s - 1)(N_s - 2)r_i^{N_s - 3} \end{bmatrix} c_i \tag{24}$$

then it can be shown (see appendix) that the partitions of $\mathbf{G}$ are given by

$$\nabla_{\mathbf{r}_R}(\nabla_{\mathbf{r}_R}\mathcal{Q})^T = 2\mathrm{Re}\left\{\mathbf{F}^{*T}\mathbf{F} + \mathrm{diag}\left(\mathbf{S}^{*T}\epsilon\right)\right\} \tag{25}$$

$$\nabla_{\mathbf{r}_R}(\nabla_{\mathbf{r}_I}\mathcal{Q})^T = 2\mathrm{Im}\left\{\mathbf{F}^{*T}\mathbf{F} - \mathrm{diag}\left(\mathbf{S}^{*T}\epsilon\right)\right\} \tag{26}$$

$$\nabla_{\mathbf{r}_I}(\nabla_{\mathbf{r}_R}\mathcal{Q})^T = -2\mathrm{Im}\left\{\mathbf{F}^{*T}\mathbf{F} - \mathrm{diag}\left(\mathbf{S}^{*T}\epsilon\right)\right\} \tag{27}$$

$$\nabla_{\mathbf{r}_I}(\nabla_{\mathbf{r}_I}\mathcal{Q})^T = 2\mathrm{Re}\left\{\mathbf{F}^{*T}\mathbf{F} - \mathrm{diag}\left(\mathbf{S}^{*T}\epsilon\right)\right\} \tag{28}$$

where the notation "diag" applied to a vector represents the operation of forming a diagonal matrix whose components are the components of the vector. Thus the gradient $\mathbf{g}$ and Hessian $\mathbf{G}$ are convenient to compute.

3.3 Real Axis Poles and Zeros

Some discussion is necessary about how the iterative Prony algorithm handles poles and zeros on the real axis. When such poles and zeros occur, the imaginary parts and all of the associated derivatives are zero. Thus poles and zeros initially on the real axis remain on the real axis through all successive iterations. This can be a practical disadvantage because the original Prony algorithm, which is used to produce the starting configuration of poles and zeros, can sometimes place a pair of poles or zeros on the real axis, while in fact this pair really belongs off of the real axis in a conjugate symmetric configuration. In this situation the method so far described will never reach the true configuration. A modification is therefore introduced to deliberately displace those roots from the real axis. The algorithm places the roots in a conjugate symmetric position by calculating the mean of their values and adding to them a small arbitrary offset in the imaginary direction, and proceeds with the iterations. If the tendency of the roots is to "go back" to the real axis then they are returned to their initial position and the iterations continue.
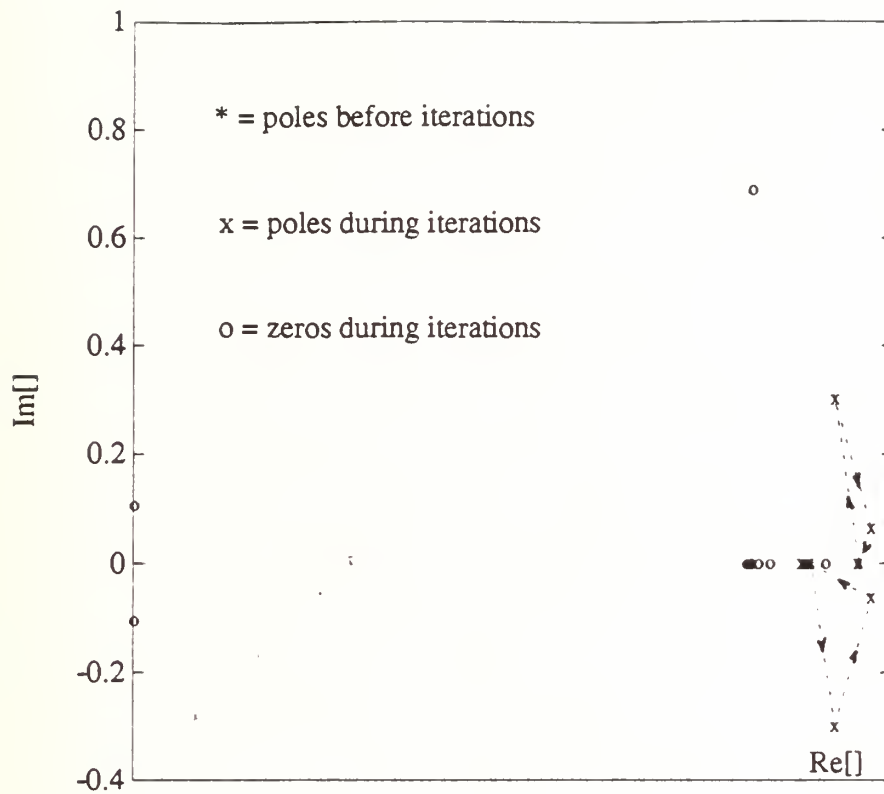
10

Figure 2: Displacement of the poles and zeros of an order (4,3) model; the modeled signal in this case actually has two poles on the real axis.

Otherwise the roots may continue to spread apart and move as a complex pair. Figure 2 is an example of the displacement of the poles and zeros of an order $(P,Q) = (4,3)$ model. In this case the modeled signal actually has two poles on the real axis and the initial model correctly placed two of the poles on the real axis. Those poles are displaced from the real axis by an amount of approximately 0.3 in the imaginary direction by the algorithm, but then after some iterations it is clear that the poles are tending to return to the real axis. At this point the poles are returned to the real axis by setting their imaginary parts to zero and the iterations continue until convergence is obtained. The opposite situation is shown in Figure 3. In this case the initial model also has two poles located on the real axis, but contrary to the case presented above, the roots, after being displaced from the real axis, (again by an amount of 0.3), continue to move away from the axis until they reach their final position in the first and fourth quadrants closer to the unit circle.
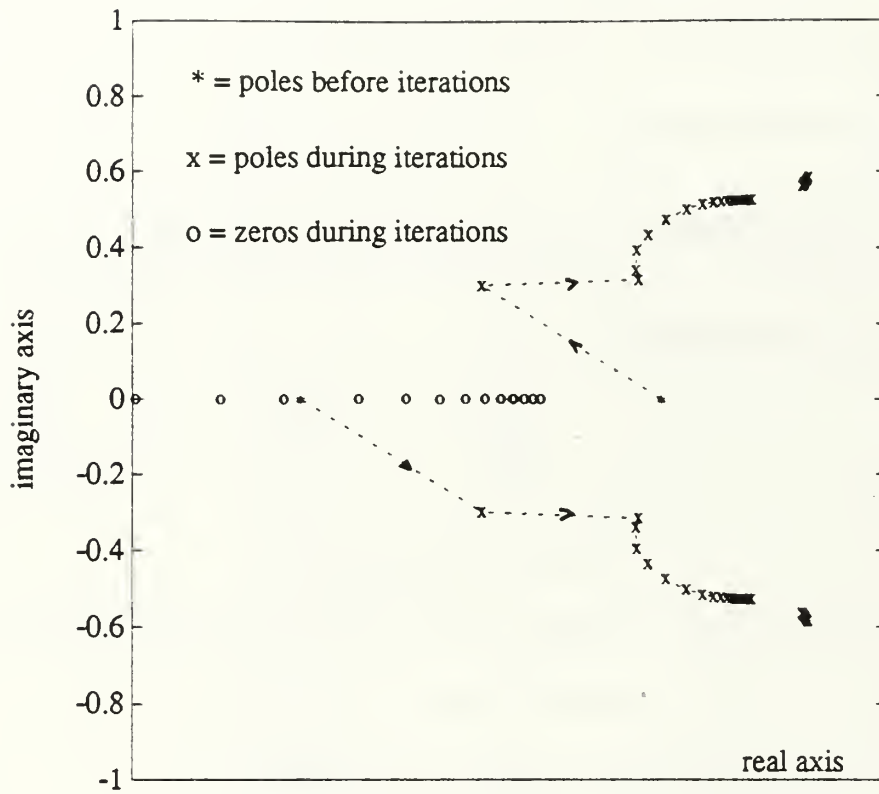
Figure 3: Displacement of the poles and zeros of an order (4,3) model. The initial model shown has poles on the real axis; the final model in this case has those poles moved to conjugate symmetric locations.

12

4 Performance Comparison

The iterative Prony method has been tested on a large variety of data including both simulated and recorded sonar and speech data. In testing, we compared the performance of the new iterative Prony algorithm to that of the iterative prefiltering algorithm, which previously had been found to be the algorithm most suitable for our work in waveform modeling of acoustic data in the signal domain [5]. In tests on simulated data, we found that when the correct order of the model is known and it is used to model the data, then both iterative algorithms tested produced very good models. However, when given an order higher than the true order of the signal, the iterative prefiltering algorithm tended to oscillate and took longer to reach convergence. On the other hand, in most cases the performance of the iterative Prony method was not affected by this change in the order of the model. The final model would simply have extra poles and zeros positioned to cancel each other.

On the real recorded data we found that in all cases the iterative Prony method provides much better models than the regular (non-iterative) Prony method. In many cases, the performance of the iterative prefiltering method was comparable to that of the iterative Prony method, but in a large number of cases the error produced by the iterative prefiltering algorithm was highly oscillatory. The same number of iterations was used for both the iterative Prony and the iterative prefiltering methods in all of the tests. In order to obtain the best possible results from the iterative prefiltering algorithm (especially in those cases when there was no convergence), the model selected was that corresponding to the iteration with the lowest error between the model and the original sequence.

Figure 4 shows the results of modeling a 100-point segment (approximately 8 ms) of underwater recorded data (ice cracking). The signal was modeled using the signal domain Prony method of section 2 and the two iterative algorithms mentioned above. A model of order (12,11) was used in all cases. It is clear that the models produced by the iterative methods (Fig. 4(b) and (c)) represent the original signal much more faithfully than the model produced by the non-iterative method (Fig 4(a)). It can also be seen that the model produced by the iterative Prony method provides a much better representation of the original signal than the model produced using the iterative prefiltering algorithm. Figure 5 shows the behavior of the sum of squared errors at each iteration between the original signal and the models produced by the two iterative methods compared here. The oscillating behavior of the error when using the iterative prefiltering algorithm was found to be one of the main disadvantages of using this method. Because the iterative Prony method is based in minimizing the error between the model and the original signal
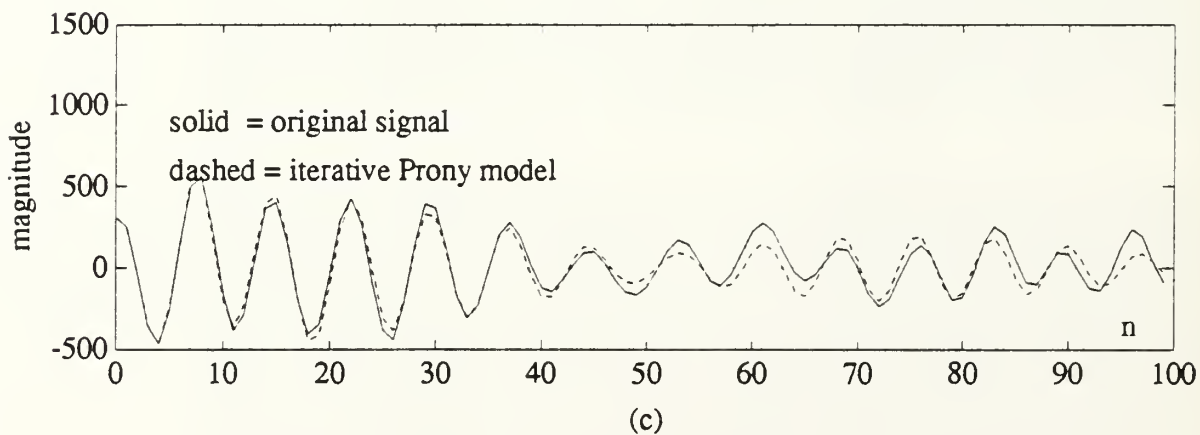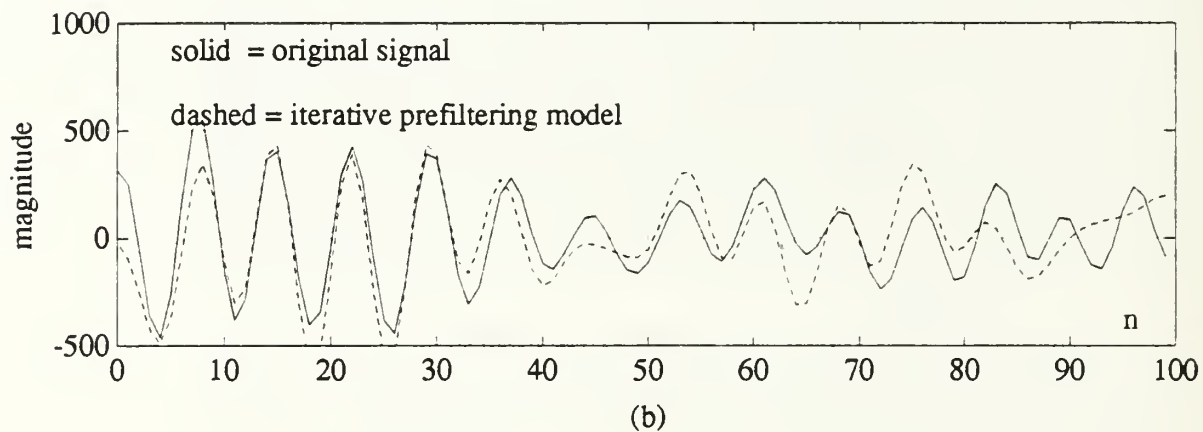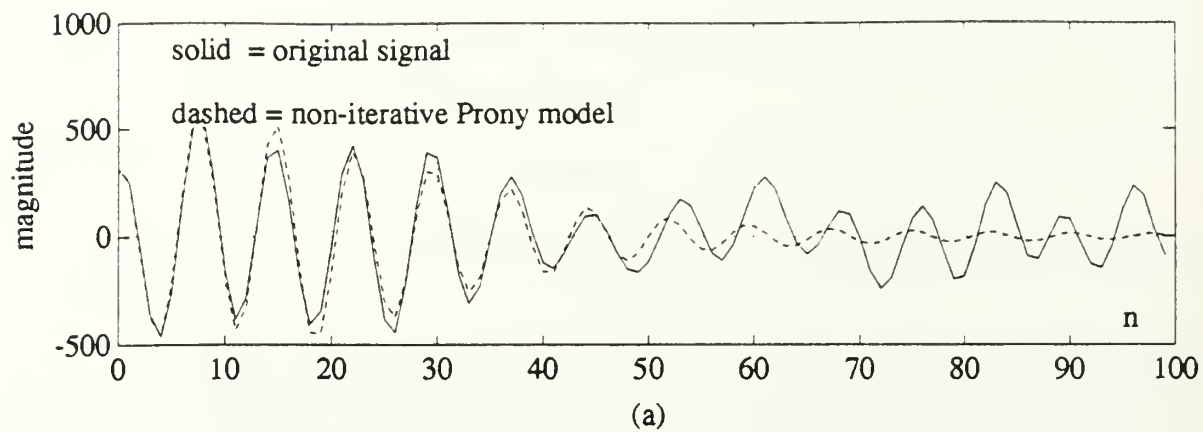
13

Figure 4: Segment of recorded underwater sound corresponding to ice cracking and three ARMA models. (a) Non-iterative Prony model. (b) Iterative prefiltering model. (c) Iterative Prony model.
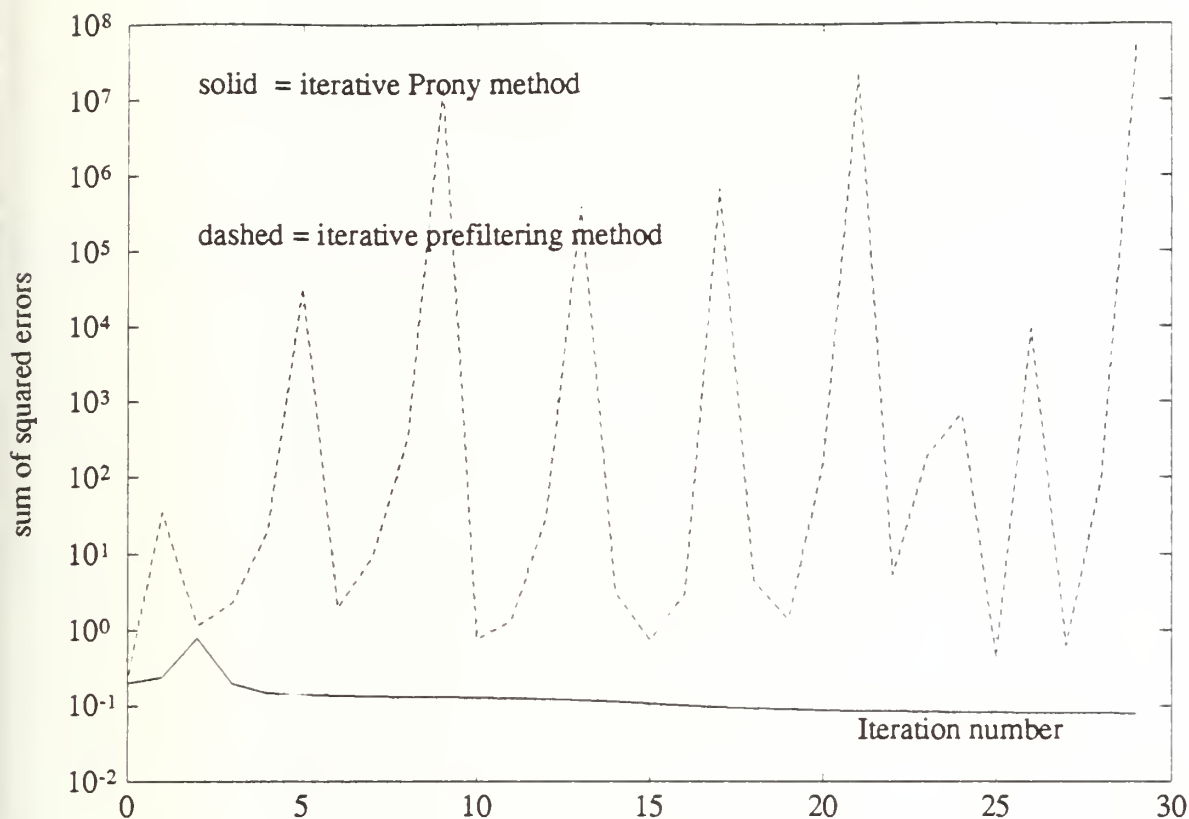
14

Figure 5: Sum of squared errors at each iteration between the original and the models produced by the iterative prefiltering and iterative Prony methods.

by directly adjusting critical parameters, the behavior of the error tends to be smoother and to decrease monotonically toward convergence with increasing number of iterations. In this case the error rises slightly at the third iteration but then decreases monotonically for the remainder of the test.

5   Conclusions

ARMA models can produce very accurate reproductions of short segments of acoustic data but the accuracy depends critically on the method used to find the model parameters. Frequently nonminimum-phase models are required to match the data in the signal domain. Prony's method is convenient to use for ARMA modeling because it finds the AR and MA parameters separately by solving *linear* equations. However the model provided by Prony's method is frequently suboptimal. The iterative Prony method described in this paper results in a significant performance improvement over the basic Prony method in practical problems while retaining the separability characterizing the original method. The new method is based on iteratively moving the poles of the model in directions that tend to decrease the error between the original and modeled data and has been found

experimentally to have excellent convergence properties.

The new method has also been compared with the iterative prefiltering algorithm of Steiglitz and McBride [7]. Neither method is guaranteed to have monotonic convergence of the error, but in many cases of testing on real and simulated data, where the true model order was not known, the iterative prefiltering algorithm had an oscillatory behavior while the iterative Prony method showed monotonic or near monotonic convergence. The computational requirements of the two algorithms were compared by testing and counting floating point operations within the MATLAB implementation. For a complex data set, the computational requirements for the iterative Prony method are approximately

$$672P^3 + (24N_s + 102)P^2 + (60N_s + 46)P + 198N_s,$$

where P is the number of poles and N_s is the length of the data, while those for the iterative prefiltering algorithm are

$$64(P + Q - 1)^3 + 8N_s(P + Q)^2 + 10(P + Q)N_s + 12N_s,$$

where Q is the number of zeros. In typical problems of models in the range of 8 to 16 poles and zeros the iterative Prony method requires about 25% more operations than the iterative prefiltering method. However the likelihood of the iterative Prony method to reach convergence in practical applications can result in running the algorithm for a fewer *total* number of iterations and thus an overall *decrease* in the total computation.

References

[1] Charles W. Therrien. *Discrete Random Signals and Statistical Signal Processing.* Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1992.

[2] James H. McClellan. Parametric signal modeling. In Jae S. Lim and Alan V. Oppenheim, editors, *Advanced Topics in Signal Processing.* pages 1–57, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1988.

[3] Leland B. Jackson. *Digital Filters and Signal Processing.* Kluwer, Boston. second edition, 1989.

[4] Gary L. May. *Pole-Zero Modeling of Transient Waveforms: A Comparison of Methods with Application to Acoustic Signals.* Master's Thesis. Naval Postgraduate School, Monterey, California. March 1991.

[5] Charles W. Therrien and Gary L. May. Comparison of ARMA Modeling Methods in the Time Domain. Proceedings 34th Midwest Symposium on Circuits and Systems. Monterey, California, May 14-17, 1991.

[6] H. Akaike. A new look at statistical model identification. *IEEE Transactions on Automatic Control,* AC-19:716–723. December 1974.

[7] K. Steiglitz and L. E. McBride. A technique for the identification of linear systems. *IEEE Transactions on Automatic Control.* AC-10:461–464. October 1965.

[8] K. Steiglitz. On the simultaneous estimation of poles and zeros in speech analysis. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* ASSP-25:229–234. June 1977.

[9] A. G. Evans and R. Fischl. Optimal Least-Squares Time-Domain Synthesis of Recursive Digital Filters. *IEEE Transactions on Audio and Electroacoustics,* AU-21:61–65, February 1973.

[10] R. N. McDonough and W. H. Huggins. Best Least-Squares Representation of Signals by Exponentials. *IEEE Transactions on Automatic Control.* AC-13:408–412, August 1968.

[11] S. Lawrence Marple. Jr. *Digital Spectral Analysis with Applications.* Prentice Hall, Inc., Englewood Cliffs. New Jersey. 1987.

[12] Byron S. Gottfried and Joel Weisman *Introduction to Optimization Theory.* Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1973.

[13] Roger Fletcher. *Practical Methods of Optimization.* Second Edition, John Wiley & Sons, New York, 1987.

[14] D. W. Marquardt. An algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal,* Volume 11, No. 2, pp. 431-441, June 1963.

A Appendix

A.1 Gradient vector g

From (15) it is easy to see that

$$\frac{\partial \epsilon}{\partial r_i} = \frac{\partial \mathbf{R}}{\partial r_i}\mathbf{c} \quad \text{and} \quad \frac{\partial \epsilon^{*T}}{\partial r_i} = \mathbf{c}^{*T}\frac{\partial \mathbf{R}^{*T}}{\partial r_i} \tag{29}$$

where r_i here represents the real or imaginary part of the ith root. Using (29) and the chain rule in (17) leads to

$$\frac{\partial Q}{\partial r_{R_i}} = \epsilon^{*T}\frac{\partial \mathbf{R}}{\partial r_{R_i}}\mathbf{c} + \mathbf{c}^{*T}\frac{\partial \mathbf{R}^{*T}}{\partial r_{R_i}}\epsilon = 2\mathrm{Re}\left[\epsilon^{*T}\frac{\partial \mathbf{R}}{\partial r_{R_i}}\mathbf{c}\right] \tag{30}$$

and explicit evaluation of the partial derivative of the matrix $\mathbf{R}$ results in

$$\frac{\partial \mathbf{R}}{\partial r_{R_i}}\mathbf{c} = \begin{bmatrix} 0 \\ 1 \\ 2\left(r_{R_i} + jr_{I_i}\right) \\ 3\left(r_{R_i} + jr_{I_i}\right)^2 \\ \vdots \\ \left(N_s - 1\right)\left(r_{R_i} + jr_{I_i}\right)^{N_s-2} \end{bmatrix} c_i \stackrel{\text{def}}{=} \mathbf{f}_i \tag{31}$$

where c_i is the ith component of the vector $\mathbf{c}$. Then, using (31), equation 30 can be written as

$$\frac{\partial Q}{\partial r_{R_i}} = 2\mathrm{Re}\left[\epsilon^{*T}\mathbf{f}_i\right] = 2\mathrm{Re}\left[\mathbf{f}_i^{*T}\epsilon\right] \tag{32}$$

Finally, using (32) for $i = 1\cdots P$, the upper partition of (19) becomes

$$\nabla_{\mathbf{r}_R}Q = 2\,\mathrm{Re}\left\{\begin{bmatrix} \mathbf{f}_1^{*T} \\ \mathbf{f}_2^{*T} \\ \vdots \\ \mathbf{f}_P^{*T} \end{bmatrix}\epsilon\right\} \tag{33}$$

which is the same as (21).

A similar procedure can be used to obtain the gradient of Q with respect to the imaginary part of the vector $\mathbf{r}$. Once more, from (29) and the chain rule applied to (17)

it follows that

$$\frac{\partial \mathbf{R}}{\partial r_{I_i}} \mathbf{c} = j\mathbf{f}_i \tag{34}$$

$$\mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{I_i}} = -j\mathbf{f}_i^{*T} \tag{35}$$

Thus

$$\frac{\partial \mathcal{Q}}{\partial r_{I_i}} = j\epsilon^{*T}\mathbf{f}_i - j\mathbf{f}_i^{*T}\epsilon = 2\mathrm{Im}\,[\mathbf{f}_i^{*T}\epsilon] \tag{36}$$

and (22) follows.

A.2   Hessian matrix G

An expression for the Hessian matrix **G** can be obtained as follows. Using a somewhat more convenient notation (20) can be rewritten as

$$\mathbf{G} = \left[\begin{array}{c} \frac{\partial}{\partial r_{R_k}} \left[\begin{array}{cc} \frac{\partial \mathcal{Q}}{\partial r_{R_i}} & \frac{\partial \mathcal{Q}}{\partial r_{I_i}} \end{array} \right] \\ \frac{\partial}{\partial r_{I_k}} \left[\begin{array}{cc} \frac{\partial \mathcal{Q}}{\partial r_{R_i}} & \frac{\partial \mathcal{Q}}{\partial r_{I_i}} \end{array} \right] \end{array} \right] \qquad \begin{array}{l} i = 1, \ldots, P \\ k = 1, \ldots, P \end{array} \tag{37}$$

then substituting (32) and (36) into (37) yields

$$\mathbf{G} = \left[\begin{array}{c} \frac{\partial}{\partial r_{R_k}} \left(\begin{array}{cc} [\epsilon^{*T}\mathbf{f}_i + \mathbf{f}_i^{*T}\epsilon] & j[\epsilon^{*T}\mathbf{f}_i - \mathbf{f}_i^{*T}\epsilon] \end{array} \right) \\ \frac{\partial}{\partial r_{I_k}} \left(\begin{array}{cc} [\epsilon^{*T}\mathbf{f}_i + \mathbf{f}_i^{*T}\epsilon] & j[\epsilon^{*T}\mathbf{f}_i - \mathbf{f}_i^{*T}\epsilon] \end{array} \right) \end{array} \right] \qquad \begin{array}{l} i = 1, \ldots, P \\ k = 1, \ldots, P. \end{array} \tag{38}$$

Now using the chain rule and both expressions in (29) leads to

$$\mathbf{G} = \left[\begin{array}{cc} \left[\epsilon^{*T} \frac{\partial \mathbf{f}_i}{\partial r_{R_k}} + \mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{R_k}} \mathbf{f}_i + \mathbf{f}_i^{*T} \frac{\partial \mathbf{R}}{\partial r_{R_k}} \mathbf{c} + \frac{\partial \mathbf{f}_i^{*T}}{\partial r_{R_k}} \epsilon \right] & j\left[\epsilon^{*T} \frac{\partial \mathbf{f}_i}{\partial r_{R_k}} + \mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{R_k}} \mathbf{f}_i - \mathbf{f}_i^{*T} \frac{\partial \mathbf{R}}{\partial r_{R_k}} \mathbf{c} - \frac{\partial \mathbf{f}_i^{*T}}{\partial r_{R_k}} \epsilon \right] \\ \left[\epsilon^{*T} \frac{\partial \mathbf{f}_i}{\partial r_{I_k}} + \mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{I_k}} \mathbf{f}_i + \mathbf{f}_i^{*T} \frac{\partial \mathbf{R}}{\partial r_{I_k}} \mathbf{c} + \frac{\partial \mathbf{f}_i^{*T}}{\partial r_{I_k}} \epsilon \right] & j\left[\epsilon^{*T} \frac{\partial \mathbf{f}_i}{\partial r_{I_k}} + \mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{I_k}} \mathbf{f}_i - \mathbf{f}_i^{*T} \frac{\partial \mathbf{R}}{\partial r_{I_k}} \mathbf{c} - \frac{\partial \mathbf{f}_i^{*T}}{\partial r_{I_k}} \epsilon \right] \end{array} \right]$$

$$i = 1, \ldots, P; \quad k = 1, \ldots, P \tag{39}$$

and from the definition of $\mathbf{f}_i$ in (31) it is seen that

$$\frac{\partial \mathbf{f}_i}{\partial r_{R_k}} = \begin{cases} \begin{bmatrix} 0 \\ 0 \\ 2 \\ 6\left(r_{R_i} + jr_{I_i}\right) \\ 12\left(r_{R_i} + jr_{I_i}\right)^2 \\ \vdots \\ (N_s - 1)(N_s - 2)\left(r_{R_i} + jr_{I_i}\right)^{N_s-3} \end{bmatrix} c_i & \text{if i=k} \\ \\ 0 & \text{otherwise} \end{cases} \tag{40}$$

Thus if $\mathbf{s}_i$ is defined by (24) it is seen that

$$\frac{\partial \mathbf{f}_i}{\partial r_{R_k}} = \mathbf{s}_i \delta_{ik} \tag{41}$$

where δ_{ik} is the dirac delta. In the same way it can be shown that

$$\frac{\partial \mathbf{f}_i}{\partial r_{I_k}} = j\mathbf{s}_i \delta_{ik} \tag{42}$$

These last two expressions together with (31),(34). and (35) can now be substituted in (39) to obtain

$$\mathbf{G} = \begin{bmatrix} \left[\epsilon^{*T}\mathbf{s}_i\delta_{ik} + \mathbf{f}_k^{*T}\mathbf{f}_i + \mathbf{f}_i^{*T}\mathbf{f}_k + \mathbf{s}_i^{*T}\epsilon\delta_{ik}\right] & j\left[\epsilon^{*T}\mathbf{s}_i\delta_{ik} + \mathbf{f}_k^{*T}\mathbf{f}_i - \mathbf{f}_i^{*T}\mathbf{f}_k - \mathbf{s}_i^{*T}\epsilon\delta_{ik}\right] \\ j\left[\epsilon^{*T}\mathbf{s}_i\delta_{ik} - \mathbf{f}_k^{*T}\mathbf{f}_i + \mathbf{f}_i^{*T}\mathbf{f}_k - \mathbf{s}_i^{*T}\epsilon\delta_{ik}\right] & -\left[\epsilon^{*T}\mathbf{s}_i\delta_{ik} - \mathbf{f}_k^{*T}\mathbf{f}_i - \mathbf{f}_i^{*T}\mathbf{f}_k + \mathbf{s}_i^{*T}\epsilon\delta_{ik}\right] \end{bmatrix}$$

$$i = 1,\ldots,P; \quad k = 1,\ldots.P. \tag{43}$$

Since the elements of the matrix $\mathbf{G}$ are formed by additions and subtractions of complex scalars with their respective complex conjugates, an alternate expression for $\mathbf{G}$ is

$$\mathbf{G} = \begin{bmatrix} 2\text{Re}\left[\mathbf{f}_i^{*T}\mathbf{f}_k\right] + 2\text{Re}\left[\mathbf{s}_i^{*T}\epsilon\delta_{ik}\right] & 2\text{Im}\left[\mathbf{f}_i^{*T}\mathbf{f}_k\right] - 2\text{Im}\left[\mathbf{s}_i^{*T}\epsilon\delta_{ik}\right] \\ -2\text{Im}\left[\mathbf{f}_i^{*T}\mathbf{f}_k\right] + 2\text{Im}\left[\mathbf{s}_i^{*T}\epsilon\delta_{ik}\right] & 2\text{Re}\left[\mathbf{f}_i^{*T}\mathbf{f}_k\right] - 2\text{Re}\left[\mathbf{s}_i^{*T}\epsilon\delta_{ik}\right] \end{bmatrix}$$

$$i = 1,\ldots,P; \quad k = 1,\ldots,P. \tag{44}$$

Finally, since the notation in (44) specifies the ik^{th} element of each partition, we notice that it is equivalent to (25) through (28).

INITIAL DISTRIBUTION LIST

9. Professor Roberto Cristi, Code EC/Cx   1
    Department of Electrical and Computer Engineering
    Naval Postgraduate School
    Monterey, CA 93943-5121

10. Professor Ralph Hippenstiel, Code EC/Hi   1
    Department of Electrical and Computer Engineering
    Naval Postgraduate School
    Monterey, CA 93943-5121

11. C. C. Carlos H. Velasco   1
    Escuela Naval de Cadetes Almircante Padilla
    Manzanillo, Casa No. 62
    Cartagena, Columbia, South America