ELSEVIER Linear Algebra and its Applications 284 (1998) 41–52

LINEAR ALGEBRA AND ITS APPLICATIONS

A fast method to diagonalize a Hankel matrix [1]

Daniel L. Boley [a,*], Franklin T. Luk [b], David Vandevoorde [b]

[a] *Department of Computer Science, University of Minnesota, 200, Union Street, Minneapolis, MN 55455-0159 USA*

[b] *Rensselaer Polytechnic Institute, Troy, NY 12180 USA*

Received 6 June 1997; accepted 8 June 1998

Submitted by D.P. O'Leary

Abstract

We consider a Vandermonde factorization of a Hankel matrix, and propose a new approach to compute the full decomposition in $O(n^2)$ operations. The method is based on the use of a variant of the Lanczos method to compute a tridiagonal matrix whose eigenvalues are the modes generating the entries in the Hankel matrix. By adapting existing methods to solve for these eigenvalues and then for the coefficients, one arrives at a method to compute the entire decomposition in $O(n^2)$ operations. The method is illustrated with a simple numerical example.

Keywords: Hankel matrix; Vandermonde decomposition; Prony's method; Lanczos algorithm

1. Introduction

A signal or an impulse response generated by n distinct modes can be represented by an $n \times n$ nonsingular, complex, symmetric Hankel matrix

* Corresponding author. E-mail: boley@cs.umn.edu.

[1] This research was partially supported by NSF Grants CCR-9405380 and CCR-9628786.

0024-3795/98/$19.00
PII: S0024-3795(98)10101-5

$$H \triangleq \begin{pmatrix} h_1 & h_2 & h_3 & \cdots & h_n \\ h_2 & h_3 & h_4 & \cdots & h_{n+1} \\ h_3 & h_4 & h_5 & \cdots & h_{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_n & h_{n+1} & h_{n+2} & \cdots & h_{2n-1} \end{pmatrix}.$$

It is well established that H admits a Vandermonde factorization

$$H = VDV^{\mathrm{T}}, \tag{1}$$

where V is a Vandermonde matrix and D is diagonal; that is,

$$V \triangleq \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 & \cdots & \lambda_n \\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 & \cdots & \lambda_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda_1^{n-1} & \lambda_2^{n-1} & \lambda_3^{n-1} & \cdots & \lambda_n^{n-1} \end{pmatrix}$$

and

$$D \triangleq \begin{pmatrix} d_1 & 0 & 0 & \cdots & 0 \\ 0 & d_2 & 0 & \cdots & 0 \\ 0 & 0 & d_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & d_n \end{pmatrix}.$$

The general case where the modes $\{\lambda_i\}$ are not distinct was developed in [9] (and references therein); recent discussion can be found in [26] for finite Hankel matrices and in [8] for infinite Hankel matrices of finite rank. The decomposition (1) can be used to extract the important modes of a noisy signal, which is equivalent to computing the minimal realization of a system from its impulse response, i.e., the problem of system identification (see e.g. [21]). Many relations between Hankel matrices and partial realizations are found in [2,14,15,18]; these techniques are closely related to the theory of moments [1], which has applications in many areas, including computational geometry [12]. A famous algorithm for estimating a minimal realization was proposed by Kung [20]; however, his method is expensive because it uses the singular value decomposition and the generalized eigenvalue decomposition (both $O(n^3)$ algorithms).

In this note we present a new approach composed of known techniques to compute (1) in $O(n^2)$ operations. The paper is organized as follows. In Section 2 we describe the classical method of Prony to find $\{\lambda_i\}$. In Section 3

we give our Lanczos-based method to generate the tridiagonal matrix T with the desired modes. In Section 4 we discuss existing fast methods to obtain the modes from T and to solve for the coefficients $\{d_i\}$. A numerical example is given in Section 5, and concluding remarks in Section 6.

2. Prony's method

In 1795 Prony [25] proposed a two-step method to calculate the modes $\{\lambda_i\}$. First, consider the recurrence that generates the entries in H:

$$h_k = a_{n-1}h_{k-1} + a_{n-2}h_{k-2} + \cdots + a_0 h_{k-n} \quad \text{for } k = n+1, \ldots, 2n;$$

this relation can be re-written as

$$\begin{pmatrix} h_1 & h_2 & h_3 & \cdots & h_n \\ h_2 & h_3 & h_4 & \cdots & h_{n+1} \\ h_3 & h_4 & h_5 & \cdots & h_{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_n & h_{n+1} & h_{n+2} & \cdots & h_{2n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} h_{n+1} \\ h_{n+2} \\ h_{n+3} \\ \vdots \\ h_{2n} \end{pmatrix},$$

from which the unknown coefficients $\{a_i\}$ can be solved (cf. Yule–Walker problem in signal processing [13]). Second, the modes $\{\lambda_i\}$ are given by the zeros of the polynomial

$$p(\lambda) \triangleq \lambda^n - a_{n-1}\lambda^{n-1} - a_{n-2}\lambda^{n-2} - \cdots - a_0\lambda^0.$$

It is well known that the zeros of a polynomial can be extremely sensitive to the values of the coefficients, even for polynomials of a modest degree. A popular computational alternative is to instead solve for all the eigenvalues of the companion matrix C, defined by

$$C \triangleq \begin{pmatrix} 0 & 1 & & & & \\ & 0 & 1 & & & \\ & & 0 & \ddots & & \\ & & & \ddots & 1 & \\ & & & & 0 & 1 \\ a_0 & a_1 & a_2 & \cdots & a_{n-2} & a_{n-1} \end{pmatrix}.$$

In Section 3, we develop a method to avoid explicit computation of the coefficients $\{a_i\}$. Instead, we generate a tridiagonal matrix T which possesses the same eigenvalues as C. Note also that the entry h_{2n} may be a *free parameter*; a discussion on how one may choose h_{2n} advantageously is given in [26].

3. Lanczos tridiagonalization

Our Lanczos-based method to generate the tridiagonal matrix T with the desired modes is based on one principal observation, which is that the columns of H can be regarded as the Krylov sequence generated by a shift-up matrix. This idea was used in [5] to obtain two fast methods for computing an LU factorization of a Hankel matrix; the methods therein are equivalent to the Berlekamp–Massey and Phillips algorithms [17,24].

We use a two-sided Lanczos algorithm [28] to generate a symmetric tridiagonal matrix T. When applied to a general matrix A, this algorithm begins with two starting vectors $\mathbf{x}_0$ and $\mathbf{y}_0$ such that $\mathbf{y}_0^{\mathrm{T}}\mathbf{x}_0 \neq 0$ and then proceeds to generate two sequences of vectors:

$$\mathbf{X}_k \triangleq (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}) \quad \text{and} \quad \mathbf{Y}_k \triangleq (\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{k-1}),$$

such that for every k we get a so-called *bi-orthogonality* condition

$$\mathbf{Y}_k^{\mathrm{T}}\mathbf{X}_k = D_k, \tag{2}$$

where the matrix D_k is nonsingular and diagonal. The method is based on the recursive application of the identities

$$A\mathbf{X}_k = \mathbf{X}_k\hat{T}_k + \mathbf{x}_k(0, \ldots, 0, 1)\gamma_k \quad \text{and} \quad A^{\mathrm{T}}\mathbf{Y}_k = \mathbf{Y}_k\tilde{T}_k + \mathbf{y}_k(0, \ldots, 0, 1)\tau_k$$

for suitable scalars γ_k and τ_k. Here, $\hat{T}_k$ and $\tilde{T}_k$ denote the leading $k \times k$ principal submatrices of the $n \times n$ tridiagonal matrices $\hat{T}$ and $\tilde{T}$, respectively, that result at the end of the Lanczos process. The method is summarized in the following algorithm.

Algorithm I – Lanczos algorithm

0. Start with A, $\mathbf{x}_0$ and $\mathbf{y}_0$.
1. For $k = 0, 1, 2, \ldots, n-1$,
2. Set $\mathbf{x}_{\text{new}} = A\mathbf{x}_k$ and $\mathbf{y}_{\text{new}} = A^{\mathrm{T}}\mathbf{x}_k$.
3. Set $\hat{\mathbf{x}}_{k+1} = \mathbf{x}_{\text{new}} - \hat{t}_{kk}\mathbf{x}_k - \hat{t}_{k-1,k}\mathbf{x}_{k-1}$, where $\hat{t}_{kk}$ and $\hat{t}_{k-1,k}$ are computed to enforce the bi-orthogonality condition: $[\mathbf{y}_{k-1}, \mathbf{y}_k]^{\mathrm{T}}\hat{\mathbf{x}}_{k+1} = 0$, viz. Eq. (2).
4. Scale vector by setting $\mathbf{x}_{k+1} = \hat{t}_{k+1,k}\hat{\mathbf{x}}_{k+1}$.
5. Analogously, generate $\mathbf{y}_{k+1}$ from $\mathbf{y}_{\text{new}}$, $\mathbf{y}_k$ and $\mathbf{y}_{k-1}$ so that $[\mathbf{x}_{k-1}, \mathbf{x}_k]^{\mathrm{T}}\mathbf{y}_{k+1} = 0$.

When the algorithm terminates, we have the relation $A\mathbf{X} = \mathbf{X}\hat{T}$, or

$$A = \mathbf{X}\hat{T}\mathbf{X}^{-1},$$

so that the eigenvalues of A match those of $\hat{T}$. Note that the algorithm may terminate prematurely if a *breakdown* condition occurs. In our experiments, we avoid breakdowns because our signal contains a small amount of random noise.

Usually, the major computational cost in this algorithm is incurred in step 2. However, we have a special situation here. The key tool is the Krylov sequence generated by the companion matrix C, i.e.,

$$\mathbf{K}(C, \mathbf{x}_0, n) \triangleq \begin{pmatrix} \mathbf{x}_0 & C\mathbf{x}_0 & C^2\mathbf{x}_0 & \cdots & C^{n-1}\mathbf{x}_0 \end{pmatrix}. \tag{3}$$

Now we need to choose $\mathbf{x}_0$. Let $\mathbf{h}$ denote the first column of H:

$$\mathbf{h} \triangleq \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_n \end{pmatrix} = He_1.$$

From [10], p. 207 we find that the Hankel matrix H is a Krylov sequence generated by C starting with $\mathbf{h}$:

$$H = \mathbf{K}(C, \mathbf{h}, n).$$

We run the Lanczos algorithm using

$$A = C, \quad \mathbf{x}_0 = \mathbf{h} \quad \text{and} \quad \mathbf{y}_0 = \mathbf{e}_1. \tag{4}$$

With this choice of left starting vector, the left Krylov sequence is simply

$$\mathbf{K}(C^{\mathrm{T}}, \mathbf{y}_0, n) = I_{n\times n}$$

and hence has full rank n. Under the assumption that H is nonsingular, it follows that the right Krylov sequence (3) also has full rank n. Furthermore, if H is strongly nonsingular (all leading principal minors are nonzero), then the *breakdown* condition in the Lanczos process cannot occur.

One missing link remains, namely the matrix C. Computing the coefficients $\{a_i\}$ is exactly what we wish to avoid. We show here how it is not necessary to compute C. With the above choices for $\mathbf{x}_0$ and $\mathbf{y}_0$, it is seen that the left vectors $\mathbf{Y}_n$ resulting from Lanczos Algorithm will be upper triangular, because multiplying by C^{T} in step 2 amounts to a shift-down operation. Hence the bi-orthogonality condition (2) implies that $\mathbf{X}_n$ will be lower triangular. Furthermore, at each stage k, the enforcement of the bi-orthogonality condition in step 3 involves only the first k entries of both $\mathbf{x}_{\text{new}}$ and $\mathbf{y}_{\text{new}}$, since the remaining entries in the latter are all zero. Indeed, the effect on $\mathbf{x}_{\text{new}}$ of enforcing (2) is to set the first $k-1$ entries to zero by adding linear combinations of previous $\mathbf{x}$ vectors.

We next describe how the first n rows of $\mathbf{X}_n$ can be generated without computing C at all. Let

$$Z \triangleq \begin{pmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & 0 & \ddots & \\ & & & \ddots & 1 \\ & & & & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{k} \triangleq \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_{2n} \end{pmatrix}$$

denote the $2n \times 2n$ shift-up matrix and a $2n$-vector with the Hankel entries, respectively. Then

$$\mathbf{K}(Z, \mathbf{k}, n) = \begin{pmatrix} H \\ J \end{pmatrix},$$

where H is the given Hankel matrix and J is an upper anti-triangular Hankel matrix. We claim that instead of running the Lanczos algorithm with Eq. (4), we can obtain the same results by using

$$A = Z, \quad \mathbf{x}_0 = \mathbf{k} \quad \text{and} \quad \mathbf{y}_0 = \mathbf{e}_1^{(2n)}.$$

where $\mathbf{e}_1^{(2n)}$ denotes the first column of $I_{2n \times 2n}$. The claim follows from the fact that the first n rows of the Krylov sequence $\mathbf{K}(Z, \mathbf{k}, n)$ match exactly the first n rows of $\mathbf{K}(C, \mathbf{h}, n)$, and the fact that only the first n entries in each generated $\mathbf{x}_{\text{new}}$ and $\mathbf{y}_{\text{new}}$ play any role in enforcing the bi-orthogonality condition in step 3. This is analysed in detail in Section 4.1 of [5]. The net result is that step 2, which in general is the most costly part of the Lanczos algorithm, has been reduced to a simple shift operation. The resulting algorithm is intimately related to the Berlekamp–Massey algorithm. Details are given in [5], where it is also shown how to symmetrize $\hat{T}$ by a similarity transformation via a diagonal scaling matrix Δ

$$\Delta \hat{T} \Delta^{-1} = T.$$

The off-diagonal entries of T are related to those of $\hat{T}$ by

$$t_{i+1,i} = t_{i,i+1} = \sqrt{\hat{t}_{i+1,i} \hat{t}_{i,i+1}} \quad \text{for } i = 1, 2, \ldots, n-1,$$

where these may be complex even if those of $\hat{T}$ are real. The diagonal entries and eigenvalues of T match those of $\hat{T}$. We get

$$T = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}.$$

The adaptation of Algorithm I to generate the parameters $\{\alpha_i\}$ and $\{\beta_i\}$ directly from the $\{h_i\}$ is summarized in Algorithm II. We pick a row recurrence scheme, and work with an $n \times (2n-1)$ matrix F, whose first row is given by

$$f_{1,j} = h_j \quad \text{for } j = 1, 2, \ldots, 2n.$$

Algorithm II – Tridiagonalization algorithm

0. Start with $\alpha_1 = f_{1,2}/f_{1,1}$ and $\beta_1^2 = 0$,
1. For $i = 1, 2, \ldots, n-1$,
2. If $i > 1$, set $\alpha_i = f_{i,i+1}/f_{i,i} - f_{i-1,i}/f_{i-1,i-1}$ and $\beta_i^2 = f_{i,i}/f_{i-1,i-1}$.
3. For $j = i+1, i+2, \ldots, 2n-i$,
4. Set $f_{i+1,j} = f_{i,j+1} - \alpha_i f_{i,j} - \beta_i^2 f_{i-1,j}$.

We note that the above algorithm requires $O(n^2)$ operations.

4. Computational details

In this section, we discuss the $O(n^2)$ eigenvalue computation of a complex-symmetric (not Hermitian) tridiagonal matrix, and the $O(n^2)$ solution of a Vandermonde system of linear equations.

Once the tridiagonal matrix T has been generated, the task is to find its eigenvalues. The general QR-type algorithm is a standard algorithm for the general eigenproblem [13]. It is based on the iteration:

$$\begin{aligned} QR &= T^{\text{old}} - sI, \\ T^{\text{new}} &= RQ + sI, \end{aligned} \tag{5}$$

where Q and R are unitary and upper triangular, respectively, and s is a shift to accelerate convergence. Most implementations include many extras for efficiency and robustness which space does not permit us to discuss here. However, this standard algorithm does not preserve the tridiagonal structure present in T^{old}.

There are two variants of the QR-type method that can be applied to maintain the tridiagonal structure for non-Hermitian matrices. One possibility is the complex-symmetric QR algorithm proposed in [7] and a slightly different version in [23]. The resulting QR algorithm is a direct analog of the ordinary Hermitian QR method, but using complex-orthogonal rotations and complex-symmetric matrices instead of unitary rotations and Hermitian matrices, respectively [7,23]. The matrix Q in Eq. (5) is *complex-orthogonal*, meaning

$$Q^{\text{T}}Q = I,$$

as opposed to unitary, meaning $Q^{\text{H}}Q = I$. Since

$$T^{\text{new}} = Q^{\text{T}} T^{\text{old}} Q,$$

it follows that T^{new} preserves the symmetry and the tridiagonality properties enjoyed by T^{old}.

Another option is to use the LR algorithm [27], which is based on the LU factorization without pivoting to maintain the tridiagonal structure. The method does not preserve the complex-symmetry, so there is no need to symmetrize T beforehand. In this algorithm the Q in Eq. (5) is replaced by a lower triangular L computed using Gaussian elimination without pivoting. The LR algorithm can break down because of a zero pivot during the elimination process, but if a random shift is applied when this occurs, the process can still exhibit very rapid convergence. If T is real, an implicit double-shift LR algorithm can in principle be carried out in real arithmetic [27]. Both algorithms require $O(n)$ operations for each iteration in a manner very similar to the Hermitian analog, and the number of iterations is generally $O(n)$ in a manner very similar to the QR algorithm usually employed. So the total cost will be $O(n^2)$ for both methods. The relative merits between these alternative algorithms have not been studied in detail.

The remaining step of solving for D is well studied. From Eq. (1) we get

$$V^{-1}H\mathbf{e}_1 = DV^{\mathrm{T}}\mathbf{e}_1.$$

Let

$$\mathbf{d} \triangleq DV^{\mathrm{T}}\mathbf{e}_1 = (d_1 \quad d_2 \quad d_3 \quad \cdots \quad d_n)^{\mathrm{T}}.$$

Then we get

$$V\mathbf{d} = \mathbf{h}, \tag{6}$$

which can be solved with a fast Vandermonde solver [3], where Higham [16], p. 438 recommends ordering the eigenvalues with the Leja (or similar) ordering to achieve backward numerical stability in the algorithm in spite of the possible ill-conditioned nature of V. A simple derivation of the fast algorithm can be found in Section 4.6.2 of [13]. It is based on the fact that an implicit UL decomposition of the matrix V can be computed in $O(n^2)$ operations using divided differences [19], ch. 6. There has been much recent work on fast Vandermonde solvers, including variants exhibiting better backward stability properties for real data [11], and an $O(n \log^2 n)$ algorithm [22]. All these algorithms have been extended to the case of confluent Vandermonde matrices (arising when several λ_i's coincide).

5. Numerical example

We illustrate this method with a numerical example. We construct a 128×128 Hankel matrix (i.e. the signal $\mathbf{h}$ generating H has 256 entries, of which the last one is left over) from five "large" modes. In order to make

the matrix full rank, and indeed strongly nonsingular, we add to the signal a small amount of decaying white noise, decaying at a rate of 0.96, and with a signal to noise ratio of 10^{-6}. These parameters are a bit extreme for an engineering problem, but these suffice to illustrate some of the behavior of this algorithm. To verify that the cost is indeed $O(n^2)$, we applied this algorithm to the leading 32, 64, 128, and 256 entries of the signal. The flop counts obtained from a Matlab implementation are shown in Table 1. For this implementation, we used the modified Lanczos method as proposed in Section 3, an LR-based tridiagonal solver as discussed in Section 4, and a fast Vandermonde solver of [3] based on the Leja ordering of the computed modes.

We also measured the residual between the original matrix H and the reconstructed one obtained from the computed decomposition $V^{\mathrm{T}}DV$. This residual also has a Hankel structure, and we show its corresponding signal in Fig. 1. Among the noise modes in the generating tridiagonal matrix are some modes larger than unity in absolute value. These modes cause instability in the method, especially when generating the Vandermonde matrix or solving the Vandermonde system. Hence these modes were removed during the reconstruction. It is seen that the first half of the signal is reproduced almost precisely, since the entries of the diagonal matrix D are computed just to fit those parameters by solving Eq. (6). The remaining entries of the signal are reproduced only approximately, and an exploration of the behavior of this method beyond the first $n/2$ entries will be reserved for future work.

6. Conclusions, extensions, and future work

We have presented a brief description of a paradigm that is capable of obtaining the generic Vandermonde decomposition of a Hankel matrix in $O(n^2)$ operations. The method is based only on variants of "off-the-shelf" methods. Once it has been computed, the Vandermonde decomposition can be used to

Table 1
Flop count in megaflops from Matlab implementation, together with ratios between flop counts. If the cost were truly $O(n^2)$, the ratios would be exactly 4

Signal length	Flop count	Ratio
32	0.1073	
		3.5135
64	0.3770	
		3.8347
128	1.4457	
		3.7753
256	5.4579	

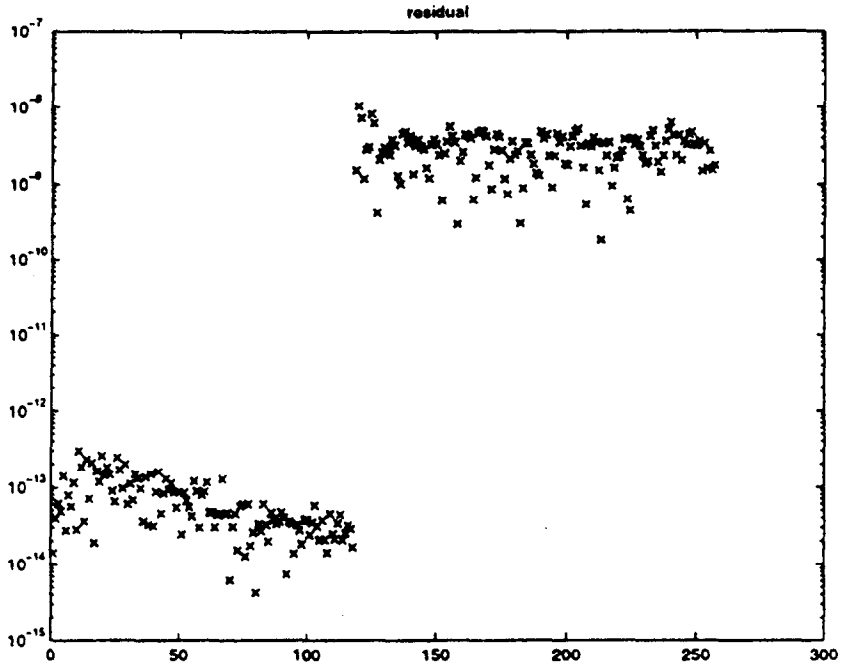

Fig. 1. Residual for signal of length 256.

produce a low rank approximation to the Hankel matrix. This is equivalent to extracting the principal modes which can approximately generate the original signal.

In this short note, we have discussed only the basic algorithm for the generic case. In particular, we have assumed that the Hankel matrix is strongly nonsingular, the same assumptions behind the LU factorization methods of [5,17,24]. If the rank r of an $n \times n$ Hankel matrix is less than n, and the leading $r \times r$ principal submatrix (call it H_r) is strongly nonsingular, then the methods of this paper can also be applied to H_r. The same is true for an infinite Hankel matrix of rank r.

If H_r is not strongly nonsingular, then one must use algorithms which are more complicated than the "off-the-shelf" methods used in this paper. Any zero leading principal minors among the first r minors will lead to breakdown in the Lanczos algorithm, necessitating the use of a look-ahead variant of the Lanczos method (see e.g. [4] and references therein). The resulting matrix T will no longer be strictly tridiagonal, but will have bulges corresponding to the look-ahead blocks. It is possible in principle to carry through the same basic steps as proposed here: (i) generate T, (ii) get the eigenvalues of T, and (iii) solve the Vandermonde system for D. But each step will require a more in-

volved algorithm at some increase in cost. A more thorough study of the most efficient algorithms available for each step in this more general case is beyond the scope of this short note.

We also have left a study of the method's stability properties to future work, because the effect of errors depends on the application involved. For example, in [6] it was demonstrated that it may be possible to reproduce the modes generating the signal, even in the presence of much more noise than used in the example of Section 5. At the noise levels of [6], the signal itself may be poorly reproduced even if the modes are well reproduced. This deserves more study with the specific application in mind.

References

[1] N.I. Akhiezer, M. Krein, Some questions in the theory of moments, AMS (1962).
[2] J. Ball, I. Gohberg, M. Kaashoek, Interpolation of Rational Matrix Functions, Birkhauser, Boston, 1990.
[3] Å. Björck, V. Pereyra, Solution of Vandermonde system of equation, Math. Comp. 24 (1970) 893–903.
[4] D.L. Boley, Krylov space methods on state-space control models, Circ. Syst. and Signal Proc. 13 (6) (1994) 733–758.
[5] D.L. Boley, T.J. Lee, F.T. Luk, The Lanczos algorithm and Hankel matrix factorization, Linear Algebra Appl. 172 (1992) 109–133.
[6] D. Boley, F. Luk, D. Vandevoorde, Fast identification of impulse response modes via Krylov space methods, in: 1997 Conference on Dec. Contr., San Diego, December 1997, pp. FM14–3.
[7] J.K. Cullum, R.A. Willoughby, A QL procedure for computing the eigenvalues of complex symmetric tridiagonal matrices, SIAM J. Matrix Anal. Appl. 17 (1) (1996) 83–109.
[8] R.L. Ellis, D.C. Lay, Factorization of finite rank Hankel and Toeplitz matrices, Linear Algebra Appl. 193 (1992) 19–38.
[9] M. Fiedler, Polynomials and Hankel matrices, Linear Algebra Appl. 66 (1985) 235–248.
[10] F.R. Gantmacher, Theory of Matrices, vol. 2, Chelsea, New York, 1959.
[11] I. Gohberg, V. Olshevsky, The fast generalized Parker–Traub algorithm for inversion of Vandermonde and related matrices, Journal of Complexity 13 (2) (1997) 208–234.
[12] G.H. Golub, P. Milanfar, J. Varah, A stable numerical method for inverting shape from moments, Technical Report SCCM-97-10, Stanford University, Sci. Comp. and Comp'l. Math. Pgm, 1997.
[13] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[14] W.B. Gragg, A. Lindquist, On the partial realization problem, Linear Algebra Appl. 50 (1983) 277–319.
[15] G. Heinig, U. Jungnickel, Hankel matrices generated by Markov parameters, Hankel matrix extension, partial realization, and Padé approximation, in: Operator Theory: Advances and Applications, vol. 19, Birkhauser, Boston, 1986, pp. 231–254.
[16] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, 1996.
[17] E. Jonckheere, C. Ma, A simple Hankel interpretation of the Berlekamp–Massey algorithm, Linear Algebra Appl. 125 (1989) 65–76.
[18] R.E. Kalman, On partial realizations, transfer functions and canonical forms, Acta Polytech. Scand. MA31 (1979) 9–32.

[19] D. Kincaid, W. Cheney, Numerical Analysis, 2nd ed., Brooks/Cole, 1996.
[20] S.Y. Kung, A new identification and model reduction algorithm via singular value decompositions, in: Proceedings of 12th Asilomar Conference Circ. Syst. Comp., November 1984, pp. 705–714.
[21] L. Ljung, System Identification: Theory for the User, Prentice-Hall, Englewood Cliffs, NJ, 1987.
[22] H. Lu, Fast solution of confluent Vandermonde linear systems, SIAM J. Matrix Anal. 15 (1994) 1277–1289.
[23] F.T. Luk, S. Qiao, Using complex-orthogonal transformations to diagonalize a complex symmetric matrix, in: Proceedings of SPIE, vol. 3162, Adv. Signal Processing Algorithms, Architectures and Implementations VII, 1997, pp. 418–425.
[24] J.L. Phillips, The triangular decomposition of Hankel matrices, Math. Comp. 25 (115) (1971) 599–602.
[25] R. Prony, Essai expérimental et analytique sur les lois de la dilatabilité et sur celles de la force expansive de la vapeur de l'eau et de la vapeur de l'alkool, à différentes températures, J. de l'École Polytechnique 1 (1795) 24-76.
[26] D. Vandevoorde, A Fast Exponential Decomposition Algorithm and Its Applications to Structured Matrices, Ph.D. Thesis, Computer Science Department, Rensselaer Polytechnic Institute, 1996.
[27] D. Watkins, L. Elsner, Chasing algorithms for the eigenvalue problem, SIAM J. Matrix Anal. 12 (1991) 374–384.
[28] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.