COMPONENT PART NOTICE

THIS PAPER IS A COMPONENT PART OF THE FOLLOWING COMPILATION REPORT:

TITLE: 'Transactions of the Conference on Applied Mathematics and Computing (9th) held in Minneapolis, Minnesota on 18-21 June 1991.

TO ORDER THE COMPLETE COMPILATION REPORT, USE AD-A252 140.

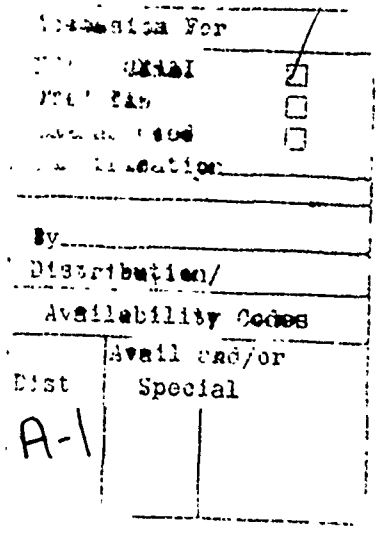
THE COMPONENT PART IS PROVIDED HERE TO ALLOW USERS ACCESS TO INDIVIDUALLY AUTHORED SECTIONS OF PROCEEDING, ANNALS, SYMPOSIA, ETC. HOWEVER, THE COMPONENT SHOULD BE CONSIDERED WITHIN THE CONTEXT OF THE OVERALL COMPILATION REPORT AND NOT AS A STAND-ALONE TECHNICAL REPORT.

THE FOLLOWING COMPONENT PART NUMBERS COMPRISE THE COMPILATION REPORT:

AD#: P006 590 thur [redacted] P006 637

AD#: AD#:

AD#: AD#:



An Accurate Algorithm for Minimal Partial Realizations

Adam W. Bojanczyk, Tong J. Lee, Franklin T. Luk
School of Electrical Engineering, Cornell University, Ithaca, New York 14853

ABSTRACT

We present a simple matrix representation of the Berlekamp-Massey algorithm for the minimum partial realizations problem, and show how pivoting can be added to the algorithm to improve numerical accuracy of the method.

1. Introduction

The problem of minimal realizatien of linear dynamical systems from input/output data has much practical importance. Many of realization procedures that have been developed rely on the solution of a Hankel system of linear equations. The Berlekamp-Massey (BM) algorithm [1], [6] is a fast Hankel linear system solver which originated in the field of coding theory. The algorithm is little known in the scientific computing community. One reason for its obscurity may be that the algorithm seems to lack a natural representation in matrix forms. Attempts to alleviate this situation can be found in Kung [4] and Jonckheere and Ma [3]. In this paper, we give a related but perhaps simpler, way to present the algorithm. We show how our presentation leads to a pivoting strategy that improves the numerical accuracy of the computation. What is more, unlike other pivoting schemes for Hankel and Toeplitz matrices, our new algorithm never requires more than $O(n^2)$ operations.

Throughout this paper, unless otherwise stated, all matrices are $n \times n$ and all vectors have n elements. Wherever convenient, we will use upper case Latin letters to denote matrices, lower case Latin letters to denote vectors, and lower case Greek letters to denote scalars. A Hankel matrix H has the form:

$$H = \begin{pmatrix} \eta_1 & \eta_2 & \cdots & \eta_n \\ \eta_2 & \eta_3 & \cdots & \eta_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \eta_n & \eta_{n+1} & \cdots & \eta_{2n-1} \end{pmatrix}, \tag{1.1}$$

and we are interested in solving the matrix equation:

$$Hx = b. \tag{1.2}$$

By rearranging its columns or rows, the Hankel system can be transformed into a Toeplitz system. For example, we can re-order the columns of H from the last to the first, and get the Toeplitz system of equations:

$$\begin{pmatrix} \eta_n & \eta_{n-1} & \cdots & \eta_1 \\ \eta_{n+1} & \eta_n & \cdots & \eta_2 \\ \vdots & \vdots & \ddots & \vdots \\ \eta_{2n-1} & \eta_{2n-2} & \cdots & \eta_n \end{pmatrix} \begin{pmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_1 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}. \tag{1.3}$$

Similarly, we define the Yule-Walker problem for the Hankel matrix to be:

$$\begin{pmatrix} \eta_1 & \eta_2 & \cdots & \eta_i \\ \eta_2 & \eta_3 & \cdots & \eta_{i+1} \\ \vdots & \vdots & \ddots & \vdots \\ \eta_i & \eta_{i+1} & \cdots & \eta_{2i-1} \end{pmatrix} y = - \begin{pmatrix} \eta_{i+1} \\ \eta_{i+2} \\ \vdots \\ \eta_{2i} \end{pmatrix}. \tag{1.4}$$

Many algorithms for solving (1.3) have been proposed, but most of them, e.g., Levinson [5], may fail to calculate an accurate solution if the Toeplitz matrix has ill-conditioned principal submatrices. Interestingly, our numerical experiments indicate that our new algorithm may still work very well under these circumstances. There is much recent interest to introduce pivoting to Toeplitz algorithms; see, e.g., [2].

This paper is organized as follows. Section 2 describes how one solves a Hankel matrix equation via the BM algorithm. Section 3 explains how the BM algorithm triangularizes a Hankel matrix that is strongly nonsingular. Section 4 presents our new numerical pivoting strategy. Section 5 considers the case of a general Hankel matrix. The last three sections contain examples that detail our numerical experience.

2. Solving a Hankel Matrix Equation

In Section 3, we will show how the Berlekamp-Massey algorithm constructs an upper triangular matrix R to reduce a Hankel matrix H to a lower triangular matrix L:

$$HR = L. \tag{2.1}$$

The triangular matrix R also has a unit diagonal. From this factorization, the Hankel system (1.2) can be easily solved. Now,

$$(HR)^T = L^T \quad \text{and} \quad H^T = H$$

imply that

$$R^T H = L^T.$$

Multiplying both sides of (1.2) by R^T, we get

$$L^T x = R^T b. \tag{2.2}$$

Hence we first apply R^T to b, and then solve the triangular system (2.2). So, if the factorization (2.1) is available, a total of n^2 multiplications is required to solve (1.2). It is worthwhile to point out here that the matrix R needs not be upper triangular. Even if R were a dense matrix, one could still solve (1.2) via (2.2), albeit at a cost of an additional $n^2/2$ multiplications. When we introduce pivoting in Section 4, we may destroy the triangularity of R.

3. Hankel Matrix Triangularization

For this section and the next, we assume that the Hankel matrix H is strongly nonsingular, i.e., that all its principal minors are nonzero. This assumption simplifies our presentation, and will be removed in Section 5. For convenience, we need a "shift-down" matrix:

$$Z = \begin{pmatrix} 0 & \cdots & 0 & 0 \\ & & & 0 \\ & I_{n-1} & & \vdots \\ & & & 0 \end{pmatrix},$$

where I_{n-1} is the identity matrix of order $(n-1)$. Thus,

$$Z \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ x_1 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{pmatrix}.$$

Note that

$$HZ = Z^T H + \begin{pmatrix} & & & -\eta_{n+1} \\ & 0_{n-1} & & \vdots \\ & & & -\eta_{2n-1} \\ \eta_{n+1} & \cdots & \eta_{2n-1} & 0 \end{pmatrix}. \tag{3.1}$$

We now show how the BM algorithm computes columns of the two matrices R and L of (2.1) recursively. We proceed by induction, and use the usual notation representing the columns of the three matrices:

$$H = (h_1, h_2, \cdots, h_n),$$

$$R = (r_1, r_2, \cdots, r_n),$$

$$L = (l_1, l_2, \cdots, l_n).$$

The first two columns of the matrices R and L are readily available:

$$r_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad , \quad r_2 = \begin{pmatrix} -\eta_2/\eta_1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{3.2}$$

and

$$l_1 = h_1 \quad , \quad l_2 = h_2 - (\eta_2/\eta_1)\, h_1. \tag{3.3}$$

Hence the top element of l_2 equals zero. Suppose now that the four columns r_j, r_{j+1}, l_j and l_{j+1} have already been computed, and that

$$H\,(r_j \quad r_{j+1}) = (l_j \quad l_{j+1})\,. \tag{3.4}$$

Let us denote the elements of r_{j+1} and l_{j+1} by

$$r_{j+1} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_j \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{and} \quad l_{j+1} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{n-j} \end{pmatrix}.$$

From the assumption of strong nonsingularity we are assuming that $\lambda_1 \neq 0$. Also, let

$$\hat{r}_{j+2} = Z r_{j+1}, \tag{3.5}$$

and

$$\hat{l}_{j+2} = H \hat{r}_{j+2}. \tag{3.6}$$

That is,

$$H\,(r_j \quad r_{j+1} \quad \hat{r}_{j+2}) = (l_j \quad l_{j+1} \quad \hat{l}_{j+2})\,.$$

The new vector $\hat{l}_{j+2}$ is easy to compute. From (3.1), it is seen that

$$\hat{l}_{j+2} = Z^T l_{j+1} + \xi e_n, \tag{3.7}$$

where

$$\xi = \eta_{n+1}\rho_1 + \eta_{n+2}\rho_2 + \cdots + \eta_{n+j}\rho_j + \eta_{n+j+1}, \tag{3.8}$$

and e_n denotes the last column of the $n \times n$ identity matrix. In words, the vector $\hat{l}_{j+2}$ is formed by "upshifting" each element of l_{j+1} by one slot and placing the scalar ξ in the n-th position. *A picture is worth a thousand words!* Hence we get

$$(l_j \quad l_{j+1} \quad \hat{l}_{j+2}) = \begin{pmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ \times & 0 & \lambda_1 \\ \times & \lambda_1 & \lambda_2 \\ \times & \lambda_2 & \lambda_3 \\ \vdots & \vdots & \vdots \\ \times & \lambda_{n-j-1} & \lambda_{n-j} \\ \times & \lambda_{n-j} & \xi \end{pmatrix}. \tag{3.9}$$

We now zero out the two leading nonzero elements of $\hat{l}_{j+2}$ by using appropriate multiples of the leading elements of l_j and l_{j+1}. That is, we post-multiply the $n \times 3$ matrix of (3.9) by the two 3×3 elimination matrices $E_1^{(0)}$ and $E_2^{(0)}$, where

$$E_1^{(0)} = \begin{pmatrix} 1 & 0 & m_1^{(0)} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad , \quad E_2^{(0)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & m_2^{(0)} \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.10}$$

and $m_1^{(0)}$, $m_2^{(0)}$ denote the corresponding multipliers. Finally,

$$(l_j \quad l_{j+1} \quad l_{j+2}) \leftarrow (l_j \quad l_{j+1} \quad \hat{l}_{j+2}) E_1^{(0)} E_2^{(0)}, \tag{3.11}$$

and

$$(r_j \quad r_{j+1} \quad r_{j+2}) \leftarrow (r_j \quad r_{j+1} \quad \hat{r}_{j+2}) E_1^{(0)} E_2^{(0)}. \tag{3.12}$$

Note that the $(j+2)$-nd component of r_{j+2} is nonzero. Hence from the strong nonsingularity assumption, the $(j+2)$-nd component of l_{j+2} is also nonzero. Thus the multipliers are well-defined.

Let us perform an operation count. The time-consuming steps include the calculation of the inner product ξ (j multiplications), and the multiplication of a scalar into the four vectors r_j, r_{j+1}, l_j and l_{j+1} ($2n$ multiplications). Hence a total of $5n^2/2$ multiplications is required to compute the decomposition (2.1).

4. Pivoting

One may have noted that the magnitude of two multipliers m_1 and m_2 of (3.10) can be arbitrarily large. In response, we propose a simple scheme of eliminating the leading nonzero element of either l_j or $\hat{l}_{j+2}$, using either $E_1^{(0)}$ or $E_1^{(1)}$, respectively, where

$$E_1^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m_1^{(1)} & 0 & 1 \end{pmatrix}. \tag{4.1}$$

The important point is that either $m_1^{(0)}$ or $m_1^{(1)}$ must be at most one in absolute value, in order to keep the overall process stable. We thus choose either $E_1^{(0)}$ or $E_1^{(1)}$ to achieve a better numerical accuracy. Our approach is somewhat similar to a pairwise pivoting scheme commonly used in systolic computing. Similarly, to eliminate the other nonzero element, we would choose among $E_2^{(0)}$,$E_2^{(1)}$, $E_2^{(2)}$ or $E_2^{(3)}$, where

$$E_2^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & m_2^{(1)} & 1 \end{pmatrix} \quad , \quad E_2^{(2)} = \begin{pmatrix} 1 & m_2^{(2)} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad , \quad E_2^{(3)} = \begin{pmatrix} 1 & 0 & 0 \\ m_2^{(3)} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.2}$$

The location (i,j) of the multiplier represents the non-zero leading entry of column j is to be eliminated by that of column i.

The column updating proceeds essentially as before:

$$(l_j \quad l_{j+1} \quad l_{j+2}) \leftarrow (l_j \quad l_{j+1} \quad \hat{l}_{j+2}) \bar{E}_1 \bar{E}_2, \tag{4.3}$$

and

$$(r_j \quad r_{j+1} \quad r_{j+2}) \leftarrow (r_j \quad r_{j+1} \quad \hat{r}_{j+2}) \bar{E}_1 \bar{E}_2, \tag{4.4}$$

where $\bar{E}_1$ equals $E_1^{(0)}$ or $E_1^{(1)}$, and $\bar{E}_2$ equals $E_2^{(0)}$, $E_2^{(1)}$, $E_2^{(2)}$ or $E_2^{(3)}$. Two important observations are as follows. First, the resultant matrix L stays lower triangular, but the previously upper triangular R may have gained two nonzero subdiagonals. Second, our pivoting scheme increases the number of multiplications by only $O(n)$, i.e , the total number of multiplications is still $5n^2/2 + O(n)$.

5. General case

Recall that under the strong nonsingularity assumption, we knew that the $(j+1)$-st element of l_{j+1} must be nonzero. We now remove the assumption that the matrix H is strongly nonsingular. During the elimination process, we may get additional leading zero elements in l_{j+1}. For our discussion in this section, let us assume that both $(j+1)$-st and $(j+2)$-nd elements are zero but that the $(j+3)$-rd element is nonzero. Hence the procedure described in Section 3 would not work because there is a gap in the nonzero structure in (3.7). Now, let

$$r_{j+1} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_j \\ \rho_{j+1} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{and} \quad l_{j+1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \lambda_3 \\ \vdots \\ \lambda_{n-j} \end{pmatrix}.$$

Define some new vectors by

$$\hat{r}_{j+2} = Zr_{j+1}, \quad \hat{r}_{j+3} = Z\hat{r}_{j+2}, \quad \hat{r}_{j+4} = Z\hat{r}_{j+3}, \tag{5.1}$$

and

$$\hat{l}_{j+2} = H\hat{r}_{j+2}, \quad \hat{l}_{j+3} = H\hat{r}_{j+3}, \quad \hat{l}_{j+4} = H\hat{r}_{j+4}. \tag{5.2}$$

That is,

$$H\begin{pmatrix} r_j & r_{j+1} & \hat{r}_{j+2} & \hat{r}_{j+3} & \hat{r}_{j+4} \end{pmatrix} = \begin{pmatrix} l_j & l_{j+1} & \hat{l}_{j+2} & \hat{l}_{j+3} & \hat{l}_{j+4} \end{pmatrix}.$$

From (3.1), the new vectors $\hat{l}_{j+2}$, $\hat{l}_{j+3}$ and $\hat{l}_{j+4}$ are calculated by

$$\begin{aligned} \hat{l}_{j+2} &= Z^T l_{j+1} + \xi_1 e_n, \\ \hat{l}_{j+3} &= Z^T \hat{l}_{j+2} + \xi_2 e_n, \\ \hat{l}_{j+4} &= Z^T \hat{l}_{j+3} + \xi_3 e_n, \end{aligned} \tag{5.3}$$

where

$$\begin{aligned} \xi_1 &= \eta_{n+1}\rho_1 + \eta_{n+2}\rho_2 + \cdots + \eta_{n+j+1}\rho_{j+1}, \\ \xi_2 &= \eta_{n+2}\rho_1 + \eta_{n+3}\rho_2 + \cdots + \eta_{n+j+2}\rho_{j+1}, \\ \xi_3 &= \eta_{n+3}\rho_1 + \eta_{n+4}\rho_2 + \cdots + \eta_{n+j+3}\rho_{j+1}. \end{aligned} \tag{5.4}$$

Indeed, the *million words* picture looks like:

$$\begin{pmatrix} l_j & l_{j+1} & \hat{l}_{j+2} & \hat{l}_{j+3} & \hat{l}_{j+4} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \\ \times & 0 & 0 & 0 & \lambda_3 \\ \times & 0 & 0 & \lambda_3 & \lambda_4 \\ \times & 0 & \lambda_3 & \lambda_4 & \lambda_5 \\ \times & \lambda_3 & \lambda_4 & \lambda_5 & \lambda_6 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \times & \lambda_{n-j-2} & \lambda_{n-j-1} & \lambda_{n-j} & \xi_1 \\ \times & \lambda_{n-j-1} & \lambda_{n-j} & \xi_1 & \xi_2 \\ \times & \lambda_{n-j} & \xi_1 & \xi_2 & \xi_3 \end{pmatrix}. \tag{5.5}$$

As described in Section 4, we would like to pivot and eliminate the above matrix so that each row contains an unique pivot element. The elimination matrices in this case are four 5×5 matrices, each with all 1's on the diagonal and a multiplier in the (i, j) location. Same as before, the jth leading non-zero entry is to be eliminated by the ith column, and all the multipliers in the elimination are less than one in absolute value.

In practice, we work with finite-precision arithmetic, so an exact zero would hardly happen. In order to tell if we are getting any additional zeros in the column of l_{j+1}, we need to choose a threshold, such that any number smaller (in absolute value) than the threshold is regarded as a zero. If this is the case, we will then apply the technique in this section to deal with the situation.

6. Numerical Examples

We consider the Hankel matrix equation (1.2) and the corresponding Toeplitz matrix equation (1.3). We compare three procedures: the BM algorithm for (1.2), our new pivoted BM algorithm for (1.2), and the Levinson algorithm for (1.3). We construct two sets of examples, the first where BM would fail and the second where Levinson would fail. Specifically, we tinker with the 2×2 leading submatrices of the Hankel and the corresponding Toeplitz matrices:

$$H^{(2)} = \begin{pmatrix} \eta_1 & \eta_2 \\ \eta_2 & \eta_3 \end{pmatrix} \quad \text{and} \quad T^{(2)} = \begin{pmatrix} \eta_n & \eta_{n-1} \\ \eta_{n+1} & \eta_n \end{pmatrix}. \tag{6.1}$$

In Example 1, the submatrix $H^{(2)}$ is ill-conditioned but the submatrix $T^{(2)}$ is not, while in Example 2, the situation is reversed. In Example 3 both submatrices are ill-conditioned. Whereas the BM algorithm fails in Examples 1 and 3, and the Levinson algorithm fails in Examples 2 and 3, our new algorithm works well on all three sets of equations.

For all examples in this paper, we choose the left hand vector b such that the solution vector $x = (1 \quad 1 \quad \cdots \quad 1)^T$. To compare the algorithms, we calculate $\| x - \hat{x} \|_2$, where $\hat{x}$ denotes the computed solution. We ran our examples using MATLAB on a Sun Sparc station. In this section we choose as a threshold, $\epsilon \| H \|_2$, where ϵ ($\approx 2.22 \cdot 10^{-16}$) denotes the machine precision. We use $\kappa(M)$ to denote the condition number with respect to the 2-norm of a matrix M.

Example 1. This example shows why pivoting is necessary for the Berlekamp-Massey algorithm. Let

$$H_1 = \begin{pmatrix} 1-\delta & 2 & 4 & 8 \\ 2 & 4 & 8 & 4 \\ 4 & 8 & 4 & 2 \\ 8 & 4 & 2 & 1-\delta \end{pmatrix} \quad \text{and} \quad H_1^{(2)} = \begin{pmatrix} 1-\delta & 2 \\ 2 & 4 \end{pmatrix}.$$

The submatrix of $H_1^{(2)}$ is ill-conditioned when δ is small, and is singular when δ is zero. As expected, the BM algorithm delivers worse accuracy as we decrease the size of δ. The matrix H_1 is well conditioned, with $\kappa(H_1) = 5.6$. However, the BM algorithm determines an L that is ill-conditioned, contributing to the loss in accuracy when one solves (1.2) via (2.2). On the other hand, our new algorithm computes a very well-conditioned L.

Table 1. Error Behavior for Example 1

	BM		Pivoted BM		Levinson
δ	$\kappa(L)$	$\| x - \hat{x} \|_2$	$\kappa(L)$	$\| x - \hat{x} \|_2$	$\| x - \hat{x} \|_2$
10^{-2}	$2.0 \cdot 10^{5}$	$9.98 \cdot 10^{-14}$	23	$3.24 \cdot 10^{-15}$	$1.11 \cdot 10^{-16}$
10^{-4}	$2.0 \cdot 10^{9}$	$4.96 \cdot 10^{-12}$	23	$3.24 \cdot 10^{-15}$	$0.00 \cdot 10^{-16}$
10^{-6}	$2.0 \cdot 10^{13}$	$4.96 \cdot 10^{-10}$	23	$3.24 \cdot 10^{-15}$	$4.00 \cdot 10^{-16}$
10^{-8}	$2.0 \cdot 10^{17}$	$1.98 \cdot 10^{-7}$	23	$3.24 \cdot 10^{-15}$	$1.11 \cdot 10^{-16}$

Example 2. This is an example where the Levinson algorithm fails because the submatrix $T_2^{(2)}$ is ill-conditioned:

$$H_2 = \begin{pmatrix} 0 & 2 & 1-\delta & 1 \\ 2 & 1-\delta & 1 & 1-\delta \\ 1-\delta & 1 & 1-\delta & 2 \\ 1 & 1-\delta & 2 & 0 \end{pmatrix} \quad \text{and} \quad T_2^{(2)} = \begin{pmatrix} 1 & 1-\delta \\ 1-\delta & 1 \end{pmatrix}.$$

This example also portrays a unique property of the BM algorithm, that it still works even though the (1,1) element of the matrix equals zero. However, the algorithm may deliver a poor solution if the (1,1) element is non-zero but small in size. Again, here the Hankel matrix is well-conditioned, with $\kappa(H_2) = 7.3$. Both our new algorithm and the BM algorithm calculate well conditioned L.

Table 2. Error Behavior for Example 2

	BM		Pivoted BM		Levinson
δ	$\kappa(L)$	$\| x - \hat{x} \|_2$	$\kappa(L)$	$\| x - \hat{x} \|_2$	$\| x - \hat{x} \|_2$
10^{-2}	36	$1.87 \cdot 10^{-15}$	5.3	$3.14 \cdot 10^{-16}$	$6.08 \cdot 10^{-15}$
10^{-4}	34	$1.85 \cdot 10^{-15}$	5.3	$6.28 \cdot 10^{-16}$	$3.07 \cdot 10^{-13}$
10^{-6}	34	$4.15 \cdot 10^{-16}$	5.3	$5.87 \cdot 10^{-16}$	$9.32 \cdot 10^{-11}$
10^{-8}	34	$2.24 \cdot 10^{-15}$	5.3	$1.11 \cdot 10^{-16}$	$1.10 \cdot 10^{-8}$

Example 3. This is an example where both BM and Levinson algorithms fail because the submatrices $H_3^{(2)}$ and $T_3^{(2)}$ are ill-conditioned:

$$H_3 = \begin{pmatrix} 1-\delta & 2 & 4 & 1-\delta & 1 \\ 2 & 4 & 1-\delta & 1 & 1-\delta \\ 4 & 1-\delta & 1 & 1-\delta & 4 \\ 1-\delta & 1 & 1-\delta & 4 & 2 \\ 1 & 1-\delta & 4 & 2 & 1-\delta \end{pmatrix}, \quad H_3^{(2)} = \begin{pmatrix} 1-\delta & 2 \\ 2 & 4 \end{pmatrix} \quad \text{and} \quad T_3^{(2)} = \begin{pmatrix} 1 & 1-\delta \\ 1-\delta & 1 \end{pmatrix}.$$

Again, here the Hankel matrix is well-conditioned, with $\kappa(H_2) \approx 31$. Our new algorithm calculates a "good" L, but the BM algorithm determines an L that is ill-conditioned.

Table 3. Error Behavior for Example 3

	BM		Pivoted BM		Levinson
δ	$\kappa(L)$	$\| x - \hat{x} \|_2$	$\kappa(L)$	$\| x - \hat{x} \|_2$	$\| x - \hat{x} \|_2$
10^{-2}	$7.0 \cdot 10^{4}$	$9.45 \cdot 10^{-11}$	279	$2.75 \cdot 10^{-14}$	$1.12 \cdot 10^{-13}$
10^{-4}	$7.0 \cdot 10^{8}$	$4.63 \cdot 10^{-15}$	288	$4.63 \cdot 10^{-15}$	$2.52 \cdot 10^{-11}$
10^{-6}	$7.0 \cdot 10^{12}$	$1.15 \cdot 10^{-14}$	288	$1.15 \cdot 10^{-14}$	$3.85 \cdot 10^{-9}$
10^{-8}	$3.8 \cdot 10^{16}$	$8.67 \cdot 10^{-14}$	288	$8.67 \cdot 10^{-14}$	$7.21 \cdot 10^{-8}$

The three examples have shown how our pivoting scheme works better than the other two conventional algorithms. However, when the size of the matrix H increases, roundoff errors may accumulate so that it becomes hard to define a numerical zero. If we choose a small threshold, such as the one we have used, then the condition number of any principal submatrix may become as large as the inverse of the threshold and a significant loss in accuracy may occur. From our experiments, we observed that the accuracy of the solution is proportional to the largest condition number of any principal submatrices.

On the other hand, if the threshold is large, we effectively work in a lower precision, so the factorization will have limited numerical accuracy. Therefore, in the next section, we experiment with a compromised threshould value. To compensate for the loss in accuracy due to this choice of a larger threshold, we adopt iterative refinement at the end.

7. Further Examples

In this section we show how our pivoting scheme works when the size of the matrix H increases. We observe that with an increase in dimensions there is a danger of underflow. Hence some form of normalization is required. Our choice is to normalize L to make the diagonal elements all ones, so that underflow can be avoided.

We construct our matrices from the Toeplitz examples in Sweet's paper [7], and select an iterative refinement scheme for improving the accuracy of the initial solution $x^{(0)}$:

1. Compute $r^{(i)} = Hx^{(i)} - b$.
2. Solve $L^T y = R^T r^{(i)}$.
3. Update $x^{(i+1)} = x^{(i)} - y$.

The criterion for ending the iterative refinement is when

$$\| r^{(i)} \|_2 < 10 \cdot \epsilon \cdot \| H \|_2 \cdot \| x^{(i)} \|_2 .$$

The threshold for the examples in this section is chosen as $10 \cdot \sqrt{\epsilon} \cdot \| H \|_2$.

Example 4. We pick an example to show how iterative refinement improves our solution, and how the number of refinements is affected by the conditioning of principal submatrices. The order-6 Hankel matrix is

$$H_4 = \begin{pmatrix} 3 & 2 & 6 & 1 & 195/14+\delta & 8 \\ 2 & 6 & 1 & 195/14+\delta & 8 & 4 \\ 6 & 1 & 195/14+\delta & 8 & 4 & -34 \\ 1 & 195/14+\delta & 8 & 4 & -34 & 5 \\ 195/14+\delta & 8 & 4 & -34 & 5 & 3 \\ 8 & 4 & -34 & 5 & 3 & 1 \end{pmatrix}.$$

For δ smaller in the magnitude than 0.5, the matrix is well conditioned with $\kappa(H_4)$ less than 100. The threshold is approximately equal to $6 \cdot 10^{-6}$. For $\delta = 0$ the order-3 principal submatrix is singular. Starting with $\delta = 10^{-2}$ and then decreasing it , we can make this submatrix progressively worse conditioned without significantly changing the condition number of H_4.

Table 4.1. Error Behavior for Example 4

	BM	Pivoted BM	BM	Pivoted BM
δ	10^{-2}	10^{-2}	10^{-4}	10^{-4}
$\sigma_{min}(H_4^{(3)})$	$1.36 \cdot 10^{-3}$	$1.36 \cdot 10^{-3}$	$1.36 \cdot 10^{-5}$	$1.36 \cdot 10^{-5}$
$\kappa(L)$	$2.29 \cdot 10^{7}$	197	$2.29 \cdot 10^{11}$	197
$\kappa(R)$	$2.29 \cdot 10^{7}$	182	$2.29 \cdot 10^{11}$	182
$\| x - \hat{x}^{(0)} \|_2$	$1.06 \cdot 10^{-8}$	$1.33 \cdot 10^{-13}$	$6.59 \cdot 10^{-5}$	$2.94 \cdot 10^{-11}$
$\| x - \hat{x}^{(1)} \|_2$	$7.61 \cdot 10^{-16}$		$7.84 \cdot 10^{-10}$	$2.88 \cdot 10^{-15}$
$\| x - \hat{x}^{(2)} \|_2$			$5.27 \cdot 10^{-15}$	
$\#Refine.$	1	0	2	1

Table 4.2. Error Behavior for Example 4 (Continued)

	BM	Pivoted BM	BM	Pivoted BM
δ	10^{-5}	10^{-5}	10^{-6}	10^{-6}
$\sigma_{min}(H_4^{(3)})$	$1.36 \cdot 10^{-6}$	$1.36 \cdot 10^{-6}$	$1.36 \cdot 10^{-7}$	$1.36 \cdot 10^{-7}$
$\kappa(L)$	$2.29 \cdot 10^{13}$	197	214	122
$\kappa(R)$	$2.29 \cdot 10^{13}$	182	202	53
$\| x - \hat{x}^{(0)} \|_2$	$1.23 \cdot 10^{-2}$	$1.12 \cdot 10^{-10}$	$1.47 \cdot 10^{-7}$	$2.11 \cdot 10^{-7}$
$\| x - \hat{x}^{(1)} \|_2$	$5.93 \cdot 10^{-6}$	$3.84 \cdot 10^{-15}$	$8.89 \cdot 10^{-14}$	$1.55 \cdot 10^{-13}$
$\| x - \hat{x}^{(2)} \|_2$	$2.71 \cdot 10^{-9}$		$1.37 \cdot 10^{-15}$	$4.29 \cdot 10^{-15}$
$\| x - \hat{x}^{(3)} \|_2$	$1.24 \cdot 10^{-12}$			
$\| x - \hat{x}^{(4)} \|_2$	$3.67 \cdot 10^{-15}$			
$\#Refine.$	4	1	2	2

Table 4.3. Error Behavior for Example 4 (Continued)

	BM	Pivoted BM	BM	Pivoted BM
δ	10^{-8}	10^{-8}	10^{-10}	10^{-10}
$\sigma_{min}(H_4^{(3)})$	$1.36 \cdot 10^{-9}$	$1.36 \cdot 10^{-9}$	$1.36 \cdot 10^{-11}$	$1.36 \cdot 10^{-11}$
$\kappa(L)$	214	122	214	122
$\kappa(R)$	202	53	202	53
$\| x - \hat{x}^{(0)} \|_2$	$1.47 \cdot 10^{-9}$	$2.11 \cdot 10^{-9}$	$1.47 \cdot 10^{-11}$	$2.11 \cdot 10^{-11}$
$\| x - \hat{x}^{(1)} \|_2$	$2.51 \cdot 10^{-15}$	$3.73 \cdot 10^{-15}$	$5.87 \cdot 10^{-16}$	$3.92 \cdot 10^{-15}$
$\#Refine.$	1	1	1	1

Note that the pivoted BM algorithm always produces factors R and L that are better conditioned than those produced by the BM algorithm without pivoting. As a consequence, the first approximation to the solution computed by the pivoted BM algorithm is more accurate than that computed by the BM algorithm. When the the smallest singular value of the order 3 principal submatrix becomes smaller than the threshold, both algorithms behave in a similar way.

Example 5. We construct a 13 x 13 Hankel matrix H_5 whose first row is given by

$$\eta_{1-13} = (-15, 10, 1, -7, -2, -5, -14.2766, -25.5087, -48.8789, -96.8384, -188.8878, -1, 5),$$

and the last column by

$$\eta_{13-25} = (5, 1, -3, 12.755, -19.656, 28.361, -7, -1, 2, 1, -6, 1, -0.5)^T,$$

the threshold is $5.35 \cdot 10^{-5}$.

The matrix is well conditioned in that $\kappa(H_5) = 89.0$, but it contains five consecutive ill-conditioned submatrices $H_5^{(4)}$ to $H_5^{(8)}$, i.e., orders 4 through 8. The smallest singular values of these five principal submatrices are $2.36 \cdot 10^{-5}$, $5.23 \cdot 10^{-5}$, $5.33 \cdot 10^{-5}$, $5.23 \cdot 10^{-5}$ and $2.36\ 10^{-5}$. The effect of encountering a sequence of ill-conditioned submatrices is felt later in the elimination process and is manifested by a severe loss in accuracy in the subsequent columns of R and L. Hence, some form of restoration is required for high accuracy. A way to bring back the lost information is to recompute the most recent columns of R by solving a Yule-Walker problem as in (1.4), utilizing the decomposition we already have at hand. In this example, columns 10 and 11 of R are recomputed, so that again the process restarts from a new and accurate point. The results are presented in Table 5. Notice that the two factor matrices produced by BM algorithm are nearly singular.

Table 5. Error Behavior for Example 5

	BM	Pivoted BM
$\kappa(L)$	$3.93 \cdot 10^{+13}$	$1.33 \cdot 10^{+3}$
$\kappa(R)$	$3.93 \cdot 10^{+13}$	$1.24 \cdot 10^{+3}$
$\| x - \hat{x}^{(0)} \|_2$	$6.58 \cdot 10^{-2}$	$3.29 \cdot 10^{-4}$
$\| x - \hat{x}^{(1)} \|_2$	$2.93 \cdot 10^{-5}$	$2.11 \cdot 10^{-9}$
$\| x - \hat{x}^{(2)} \|_2$	$8.82 \cdot 10^{-8}$	$4.18 \cdot 10^{-14}$
$\| x - \hat{x}^{(4)} \|_2$	$1.58 \cdot 10^{-13}$	
#Refine.	4	2

Example 6. We extend the previous example to size 50×50 by appending random numbers. For this example, the Hankel matrix is moderately ill-conditioned with $\kappa(H_6) = 1.97 \cdot 10^{+3}$. The results are shown in Table 6.

Table 6. Error Behavior for Example 6

	BM	Pivoted BM
$\kappa(L)$	$4.01 \cdot 10^{+13}$	$1.33 \cdot 10^{+3}$
$\kappa(R)$	$4.01 \cdot 10^{+13}$	$8.71 \cdot 10^{+4}$
$\| x - \hat{x}^{(0)} \|_2$	$5.86 \cdot 10^{-1}$	$2.44 \cdot 10^{-3}$
$\| x - \hat{x}^{(1)} \|_2$	$2.39 \cdot 10^{-3}$	$4.54 \cdot 10^{-8}$
$\| x - \hat{x}^{(2)} \|_2$	$7.46 \cdot 10^{-6}$	$9.86 \cdot 10^{-13}$
$\| x - \hat{x}^{(3)} \|_2$	$6.87 \cdot 10^{-8}$	$5.58 \cdot 10^{-14}$
$\| x - \hat{x}^{(5)} \|_2$	$1.17 \cdot 10^{-12}$	
#Refine.	5	3

8. Conclusion

We believe that the Berlekamp-Massey algorithm works well when the Hankel matrix is positive definite and well-conditioned, so that none of its principal submatrices is ill-conditioned, and no pivoting is necessary. In general, consider the Hankel matrix as a moments matrix with respect to certain weights. These weights are not necessarily all positive, and thus we may need to deal with Hankel matrices that do not have positive definite property. Strategies such as pivoting, normalization and gap-jumping are required in this case.

For the previous examples, we adopt a scheme that combines both pivoting and normalization. To generate a new column of the triangular factor, say column i, we combine it with columns $i-1$ and $i-2$. Since normalization is performed after each new column is generated, and column i is a shifted version of column $i-1$, so all the three columns have a 1 as the leading nonzero element. Therefore, no pivoting is performed in the first phase of column combination. After columns i and $i-2$ are combined, pivoting takes place in the second phase, in which columns i and $i-1$ are combined. Both steps are crucial to the stability of the procedure. Pivoting prevents multipliers from being too large, while normalization keeps the norm of the columns from underflowing.

Since we remove the constraint of positive-definiteness, a well-conditioned Hankel matrix may have several ill-conditioned submatrices. The choice of the threshold is a subtle issue, and from the previous examples, we see that a compromised threshould value, such as $10 \cdot \sqrt{\epsilon} \cdot \| H \|_2$, may be a good choice.

Whenever there is a sequence of ill-conditioned principal submatrices, i.e., a gap, we simply shift up the previous column of L until the next well-conditioned submatrix is encountered. Thus, we avoid the computation within the gap by "jumping over" it. Two columns of L are recomputed right after the gap, so that the errors in the factorization caused by the jumps are confined within the gap and do not propagate to the succeeding columns. Therefore, after a few steps of iterative refinement at the end, all the errors in the solution (not the decomposition) will be corrected and the solution will be accurate to machine precision.

Another possibility to deal with the gap is to perform a LU decomposition with partial pivoting instead of jumping over it. But the worst case for this approach requires $O(p^2n)$ operations to decompose the part of the matrix corresponding to the gap, in contrast to $O(pn)$ operations for the gap-jumping approach, where p denotes the size of the gap.

Acknowledgements

A. W. Bojanczyk was supported in part by the Army Research Office under contract DAAL03-90-G-0092. T. J. Lee and F. T. Luk were supported in part by the Army Research Office under contract DAAL03-90-G-0104, and by the Joint Services Electronics Program under contract F49620-90-C0039.

References

[1] E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York, NY, 1968.

[2] T. F. Chan and P. C. Hansen, "A stable Levinson algorithm for general Toeplitz systems," CAM Report 90-11, Computational and Applied Mathematics, University of California, Los Angeles, CA, 1990.

[3] E. Jonckheere and C. Ma, "A simple Hankel interpretation of the Berlekamp-Massey algorithm," *Linear Alg. Applics.*, vol. 125 (1989), pp. 65-76.

[4] S.-Y. Kung, "Multivariable and Multidimensional Systems: Analysis and Design," Ph.D. Dissertation, Department of Electrical Engineering, Stanford University, Stanford, CA, 1977.

[5] N. Levinson, "The Wiener RMS (root-mean-square) error criterion in filter design and prediction," *J. Math. Phys.*, vol. 25 (1946), pp. 261-278.

[6] J. L. Massey, "Shift register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. IT-15 (1967), pp. 122-127.

[7] D. R. Sweet, "Numerical methods for Toeplitz matrices," Ph.D. Dissertation, Department of Computing Science, University of Adelaide, Australia, 1982.