A look-ahead Schur Algorithm*

Kyle Gallivan Srikanth Thirumalai Paul Van Dooren

Abstract

The classical Schur algorithm computes the LDL^T factorization of a symmetric Toeplitz matrix in $O(n^2)$ operations, but requires that all the principal minors of the matrix be nonsingular. Look-ahead schemes have been proposed to deal with matrices that have exactly singular principal minors [9], [11]. Unfortunately, these algorithms cannot be extended to matrices that have ill-conditioned principal minors. Consequently, the relative errors obtained using the classical Schur algorithm on matrices having ill-conditioned principal minors is very poor. In this paper, we propose a look-ahead scheme for such matrices and present empirical results which demonstrate the improvement over the classical Schur algorithm for symmetric Toeplitz matrices with ill-conditioned principal minors.[1]

1 Introduction

Look-ahead techniques were originally proposed to improve the numerical robustness of the Lanczos algorithm applied to an indefinite matrix T in the presence of singular and nearly singular leading principal minors in T [10]. Most of the techniques related to these developments are based on the theory of orthogonal polynomials [7] or equivalently on that of T conjugate directions. This theory is in turn closely connected to that of Hankel matrices and the Padé algorithm [1] and of Toeplitz matrices and the Levinson algorithm [6]. In both cases one constructs the decomposition $L^{-1}TL^{-T} = D$ where T is the given Toeplitz matrix. The rows of L^{-1} are the conjugate directions or also contain the coefficients of the orthogonal polynomials. Look-ahead techniques have been proposed and yielded algorithms with satisfactory numerical behavior, [1], [2],[6], [10].

General look-ahead techniques have not been proposed for the Schur algorithm, which computes the decomposition $T = LDL^T$ of a given symmetric Toeplitz matrix T (only the exact singular case was treated in [5], [9]). The Schur approach for symmetric Toeplitz matrices has three basic advantages over the Levinson approach : (i) it directly constructs the matrix L rather than its inverse, (ii) it has a derivation solely based on standard matrix operations rather than on orthogonal polynomials, (iii) it is more amenable to parallel implementation (partly because of its matrix-based interpretation).

*This research was partially supported by ARPA under grant 60NANB2D1272 and the National Science Foundation under grants CCR 9209349 and CCR 9120105

[1]Appeared in Proceedings of Fifth SIAM Conference on Applied Linear Algebra, pp. 450–454.

In this paper we derive a look-ahead Schur algorithm which is entirely based on matrix operations. The pivoting techniques used do not require the polynomial machinery of the Levinson algorithm. This is an advantage in the sense that it is easily extended to symmetric block Toeplitz matrices, whereas we expect difficulties with the concept of orthogonal polynomial matrices in any look-ahead version of the block Levinson algorithm. We present here the basic ideas for "scalar" Toeplitz matrices only. The block case follows by assuming that the entries in our matrices are blocks and requires only minor modifications.

2 Algorithm

The Schur algorithm is one of the most popular techniques to compute the LDL^T decomposition of a general symmetric Toeplitz matrix T. In this section we outline one step of the look ahead technique required to improve the stability of this algorithm when the leading principal minors are ill-conditioned. Consider a Toeplitz matrix

$$T = \begin{bmatrix} t_0 & t_1 & \dots & t_n \\ t_1 & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ t_n & \dots & t_1 & t_0 \end{bmatrix}$$

of dimension $(n+1) \times (n+1)$. Define a shift matrix Z with $1's$ along the first subdiagonal. It follows that the matrix $T - ZTZ^T$ has rank 2, which is called the *displacement rank* of the Toeplitz matrix T. Matrices that have a displacement rank 2 are called *quasi-Toeplitz matrices* [8]. Fast algorithms [3], [4] to factor such matrices use the generator matrix G defined from the rank 2 factorization :

$$T - ZTZ^T = G_0^T \Sigma_0 G_0. \tag{1}$$

For an $(n+1) \times (n+1)$ Toeplitz or quasi-Toeplitz matrix T (with nonzero (1,1) element) the generator and signature matrices are of the form

$$G_0 = \begin{bmatrix} h_{00} & h_{01} & h_{02} & \cdots & h_{0n} \\ g_{00} & g_{01} & g_{02} & \cdots & g_{0n} \end{bmatrix}, \qquad \Sigma_0 = \begin{bmatrix} \sigma_{01} & 0 \\ 0 & \sigma_{02} \end{bmatrix}. \tag{2}$$

The algorithm then proceeds by applying a $\Sigma-unitary$ transformation U_0 ($U_0^T \Sigma_1 U_0 = \Sigma_0$) to G_0 such that

$$\tilde{G}_0 = U_0 G_0 = \begin{bmatrix} \tilde{h}_{00} & \tilde{h}_{01} & \tilde{h}_{02} & \cdots & \tilde{h}_{0n} \\ 0 & \tilde{g}_{01} & \tilde{g}_{02} & \cdots & \tilde{g}_{0n} \end{bmatrix}, \qquad \Sigma_1 = \begin{bmatrix} \sigma_{11} & 0 \\ 0 & \sigma_{12} \end{bmatrix}. \tag{3}$$

In the factorization $T = LDL^T$, the first row of $\tilde{G}_0$ is the first row of the triangular factor L^T and σ_{11} is the first element of D. The generator for the next step of the algorithm is obtained by shifting the first row of $\tilde{G}_0$ one place to the right :

$$\begin{bmatrix} 0 & \tilde{h}_{00} & \tilde{h}_{01} & \cdots & \tilde{h}_{0n-1} \\ 0 & \tilde{g}_{01} & \tilde{g}_{02} & \cdots & \tilde{g}_{0n} \end{bmatrix} \doteq \begin{bmatrix} 0 & h_{10} & h_{11} & \cdots & h_{1n-1} \\ 0 & g_{10} & g_{11} & \cdots & g_{1n-1} \end{bmatrix} \doteq \begin{bmatrix} 0 & \\ 0 & G_1 \end{bmatrix}. \tag{4}$$

Again, a $\Sigma - unitary$ transformation U_1 ($U_1^T \Sigma_2 U_1 = \Sigma_1$) is applied to G_1 to zero out the first entry in the second row. The first row of the resulting matrix is the second row of the upper triangular factor L^T and σ_{21} is the second diagonal element of D. Following this procedure recursively, the factorization is completed. We refer the reader to [3], [4] for a more detailed description and proof of the algorithm.

It is interesting to note that if $h_{00}^2 \sigma_{01} + g_{00}^2 \sigma_{02} = 0$ then the algorithm cannot proceed. This situation arises when the $(1,1)$ element of the Toeplitz or quasi-Toeplitz matrix T is zero. More generally, it could be that the first $k-1$ leading principal minors of T are exactly singular but the k^{th} leading principal minor not. In such cases the algorithm in [5] and [9] with a look-ahead step of k can be used. A weakness of this approach is that the algorithms in [5] and [9] cannot be used when the first $k-1$ leading principal minors of T are ill-conditioned rather than exactly singular. We propose here an alternate method to do a look-ahead step of size k when the minors are ill-conditioned. This look-ahead step groups the first k rows of T and the corresponding $k \times k$ diagonal block, and proceeds with one block elimination step.

The algorithm relies on an important property of Toeplitz and quasi-Toeplitz matrices. For such matrices, the displacement rank of the Schur complement of a nonsingular leading principal minor is also 2, i.e. the Schur complement is quasi-Toeplitz. This indicates that if we compute the Schur complement of the nonsingular leading principal minor, then two steps of the Bunch-Kaufman algorithm on the displacement of the Schur complement gives us the generators for the Schur complement. The Schur algorithm can then proceed as normal till another look-ahead step needs to be done. If we choose to use the immediate update form of the Bunch-Kaufman algorithm we would have to compute the entire Schur complement of the $k \times k$ leading principal minor. This would require $\mathrm{O}((n-k)^2)$ storage and an approximately equal amount of computation. On the other hand, if we use the delayed update version of the Bunch-Kaufman algorithm we would be computing the rows of the displacement of the Schur complement as we need them. This requires only $\mathrm{O}(n-k)$ storage and an approximately equal amount of computation.

Let us assume that the first $(k-1)$ leading principal minors of T are ill-conditioned and the k^{th} leading principal minor is well conditioned. If we were to use a look-ahead algorithm, we would have to compute the corresponding k rows of the upper triangular factor L^T and the $k \times k$ diagonal block of the diagonal matrix D. By construction it can be seen that the first k rows of T can be obtained as follows. Let

$$h = \begin{bmatrix} h_{00} & h_{01} & \cdots & h_{0n} \end{bmatrix} \qquad \text{and} \qquad g = \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0n} \end{bmatrix} \tag{5}$$

then using the colon notation of MATLAB we have

$$T(1:k,:) = \begin{bmatrix} (\sigma_{01}\, h_{00})\, h + (\sigma_{02}\, g_{00})\, g \\ (\sigma_{01}\, h_{01})\, h + (\sigma_{02}\, g_{01})\, g + T(1,:) * Z^T \\ (\sigma_{01}\, h_{02})\, h + (\sigma_{02}\, g_{02})\, g + T(2,:) * Z^T \\ \vdots \\ (\sigma_{01}\, h_{0k-1})\, h + (\sigma_{02}\, g_{0k-1})\, g + T(k-1,:) * Z^T \end{bmatrix} \tag{6}$$

where Z is the lower shift matrix. It can be seen from the above equation that we require approximately $3kn$ operations to compute the first k rows of T. We use

this to obtain the first k rows of the upper triangular factor, denoted as L_k^T, and the corresponding diagonal block, denoted as D_k. This computation requires $k^2 n$ operations. If k is small, then this computation has $O(n)$ complexity.

The next step in the look-ahead method involves obtaining the factorization of the displacement of the Schur complement of T with respect to its k^{th} leading principal minor D_k. As discussed earlier, the preferred method is the delayed update version of the Bunch-Kaufman algorithm. Since the displacement rank of the Schur complement is 2, we would need either one or two steps of the Bunch-Kaufman algorithm to obtain its generators. This requires the computation of only a few rows of the displacement of the Schur complement. The Schur complement $T_{sc}^{(k)}$ of the k^{th} leading principal minor is given as

$$T_{sc}^{(k)} = T - L_k D_k L_k^T. \tag{7}$$

Its displacement can be rewritten using (1) as :

$$\begin{aligned} T_{sc}^{(k)} - Z T_{sc}^{(k)} Z^T &= T - ZTZ^T - L_k D_k L_k^T + Z L_k D_k L_k^T Z^T \\ &= G_0^T \Sigma_0 G_0 - L_k D_k L_k^T + Z L_k D_k L_k^T Z^T. \end{aligned} \tag{8}$$

From (8) we now construct the p^{th} row of the displacement of the Schur complement. For notational simplicity, we omit the subscript "0" corresponding to the first step of the Schur algorithm.

$$\begin{aligned} p^{th}\text{row} &= (\sigma_1\ h_{p+k-1})\ h_{k+1:n+1} + (\sigma_2\ g_{p+k-1})\ g_{k+1:n+1} \\ &- L_{p+k,1:k} * T_{1:k,k+1:n+1} \\ &+ \left[\ T_{k,p+k-1},\ \ L_{1:k,p+k-1} * T_{1:k,k+1:n}\ \right] \end{aligned} \tag{9}$$

The first two terms in (9) correspond to the first term in (8). The third and fourth term in (9) correspond to the second and third term in (8).

From (9) it can be seen that this computation requires $(n-k)+2(n-k)+2k(n-k)+2k(n-k) = (4k+3)(n-k)$ operations. If k is small then the complexity again is O(n). This indicates that producing one row of the displacement of the Schur complement of the k^{th} leading principal minor requires O(n) computation. Each step of the Bunch-Kaufman algorithm produces one or two rows of the factorization. Since the rank of the matrix is two, we would need either one or two steps. This computation would require a maximum of 4 rows of the displacement of the Schur complement to be generated. Hence, the total number of operations required to obtain the generators of the Schur complement via the Bunch-Kaufman algorithm is $4(4k+3)(n-k)+4(n-k)$. The second term in this expression is the work done in the Bunch-Kaufman algorithm to produce the generators. The total work required to do one look-ahead step of k on a generator of length n using the method described above is approximately O($k^2 n$).

This gives us the generator G_k and the signature matrix Σ_k, and allows us to carry on with the Schur algorithm if the first column of G_k does not have a very small Σ_k norm. If it does, then we have to do another look-ahead step with a well-conditioned diagonal block. In our method, the decision to do a look-ahead step rather than the regular Schur algorithm step is based on the condition number of the diagonal block

D_k. If the condition number is larger than a predetermined threshold, we choose to go to a larger look-ahead step.

The above discussion indicates that the Bunch-Kaufman algorithm can be used to obtain the generator of the Schur complement following a look-ahead step. We present an alternate organization of the computation. We can rewrite (8) as :

$$\begin{aligned} T_{sc}^{(k)} - ZT_{sc}^{(k)}Z^T &= \begin{bmatrix} G_0^T & L_k & ZL_k \end{bmatrix} \begin{pmatrix} \Sigma_0 & 0 & 0 \\ 0 & -D_k & 0 \\ 0 & 0 & D_k \end{pmatrix} \begin{bmatrix} G_0 \\ L_k^T \\ L_k^T Z^T \end{bmatrix} \\ &= \hat{G}^T W \hat{G} \end{aligned} \tag{10}$$

This indicates that we can readily obtain a generator for the Schur complement. The problem we face is that the generator shown above has a rank of at most $2k+2$. We know that the minimal generator of the Schur complement of a Toeplitz matrix has rank at most 2. We, therefore, have to reduce the generator shown above so that a minimal generator of lowest rank is obtained. This is done using hyperbolic Householder matrices U satisfying the property

$$U^T \hat{W} U = W$$

such that the generator $\hat{G}$ is reduced to an upper triangular matrix. This scheme is very similar to a rank revealing QR factorization. The difference being that the norm used here is the hyperbolic norm. If we use column permutations to bring the column with the highest hyperbolic norm to the pivot column, we would essentially be doing an LDL^T factorization with symmetric pivoting. We know that such a scheme may not always exist. Alternately, we could use the Bunch-Kaufman algorithm and select either one pivot column or "two" pivot columns to carry out the skew eliminations. This would provide us with a rank revealing factorization of the displacement of the Schur complement.

3 Numerical Experiments

Consider the symmetric Toeplitz matrix whose first row is given by,

$$T = \begin{bmatrix} 1.0 & 0.999 & 0.8 & 0.9 & 0.98 & 1.165 & 0.5 & 0.6 & 0.1 & -0.1 \end{bmatrix} \tag{11}$$

The leading principal minors of size 2 and 6 have condition numbers $1.9990 * 10^3$ and $1.4415 * 10^4$ respectively. The matrix T itself has a condition number of $1.4472 * 10^3$. All the other minors are relatively well-conditioned. This indicates that a look-ahead step would have to be done at step 2 and 6. The step size was 2 in both cases since the next leading principal minors were well-conditioned. We compare this look-ahead factorization with the regular Schur algorithm by comparing the relative error

$$\text{relative error} = \frac{\|x_{computed} - x_{exact}\|}{\|x_{exact}\|}. \tag{12}$$

For the look-ahead algorithm, the relative error was $2.2566 * 10^{-13}$ whereas for the regular Schur algorithm, the error was $3.4322 * 10^{-8}$. Since the condition number of

the Toeplitz matrix was $1.4472 * 10^3$, we have obtained the best relative error one would expect in the look-ahead case. The regular Schur algorithm would require iterative refinement to obtain such a small relative error.

4 Concluding remarks

The development of the look-ahead scheme above only considered Toeplitz matrices with scalar entries. For the case of Toeplitz matrices with $m \times m$ block entries, all the above is readily extended. The shift operator Z is taken to be a block shift and the displacement rank is then bounded by $2m$, resulting in a $2m \times n$ generator G and a $2m \times 2m$ matrix Σ. The displacement of the Schur complement (8) will also have rank at most $2m$, which then becomes the maximum number of Bunch-Kaufman steps to be performed.

The main advantage of the Schur approach over the Levinson approach is its simplicity, which is inherited from its interpretation in terms of simple matrix operations, rather than polynomial operations for the Levinson approach. The Schur approach has also the important property of computing directly LDL^T rather than the matrices L^{-1} and D as in the Levinson approach. This has the following two consequences :

(i) for banded matrices (say with bandwidth $r << n$) the factor L will also have the same bandwidth, resulting in a $O(nr)$ rather than $O(n^2)$ Schur algorithm.

(ii) for semi-definite Toeplitz matrices of low rank (say with rank $r << n$), the Schur algorithm terminates after r steps, resulting again in a $O(nr)$ Schur algorithm, rather than $O(n^2)$. The obtained decomposition $T = L_r D_r L_r^T$ then provides the range space of T, which proves useful when solving, for example, least squares problems.

References

[1] S. Cabay and R. Meleshko, *A weakly stable algorithm for Padé approximants and the inversion of Hankel matrices*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 735–765.

[2] T. F. Chan and P. C. Hansen, *A look-ahead Levinson algorithm for indefinite Toeplitz systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 490–506.

[3] J. Chun, T. Kailath, and H. Lev-Ari, *Fast parallel algorithms for QR and triangular factorization*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 899–913.

[4] G. Cybenko and M. Berry, *Hyperbolic Householder algorithms for factoring structured matrices*, SIAM J. Matrix Anal. Appl, 11 (1990), pp. 499–520.

[5] P. Delsarte, Y. Genin, and Y. Kamp, *Pseudo-Carathéodory functions and Hermitian Toeplitz matrices*, Philips J. Res., 41 (1986), pp. 1–54.

[6] R. W. Freund and H. Zha, *Formally biorthogonal polynomials and a look-ahead Levinson algorithm for general Toeplitz systems*, Linear Algebra Appl., 188 (1993), p. 255.

[7] M. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, part ii*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 15–58.

[8] T. Kailath, S.-Y. Kung, and M. Morf, *Displacement ranks of matrices and linear equations*, Journal of Mathematical Analysis and Applications, 68 (1979), pp. 395–407.

[9] D. Pal and T. Kailath, *Fast triangular factorization and inversion of Hermitian, Toeplitz related matrices with arbitrary rank profile*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 1016–1042.

[10] B. Parlett, *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 567–593.

[11] C. J. Zarowski, *Schur algorithms for Hermitian Toeplitz and Hankel matrices with singular leading principal submatrices*, IEEE Trans. Signal Process., 39 (1991), pp. 2464–2480.