Stable Factorization of Hankel and Hankel-like Matrices

Vadim Olshevsky and Michael Stewart

July 16, 1998

ABSTRACT

This paper gives displacement structure algorithms for the factorization positive definite and indefinite Hankel and Hankel-like matrices. The positive definite algorithm uses orthogonal symplectic transformations in place of the Σ-orthogonal transformations used in Toeplitz algorithms. The indefinite algorithm uses a look-ahead step and is based on the observation that displacement structure algorithms for Hankel factorization have a natural and simple block generalization. Both algorithms can be applied to Hankel-like matrices of arbitrary displacement rank.

1 Introduction

For h_i with $0 \leq i \leq 2n-2$ we define the Hankel matrix

$$H = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{n-1} \\ h_1 & \iddots & & \iddots & h_n \\ h_2 & & \iddots & \iddots & \vdots \\ \vdots & \iddots & \iddots & & \vdots \\ h_{n-1} & h_n & \cdots & \cdots & h_{2n-2} \end{bmatrix}. \tag{1}$$

For real h_i, such matrices arise often in system theory. Applications in which the elements h_i are taken from a finite field arise in coding theory. The fast $O(n^2)$ Berlekamp-Massey algorithm for solving Hankel systems of equations was developed in the latter context, [1]. It generalizes to the case in which the h_i are real and it is one of the best known fast algorithms for solving Hankel systems of equations.

Several other authors have developed algorithms for the factorization or inversion of Hankel matrices, [13, 14, 15]. None of these algorithms incorporate pivoting or look-ahead. In exact arithmetic they run to completion only if H has non-singular leading principal submatrices. In floating point arithmetic, they are unstable when applied to anything other than a positive definite matrix. Further, some of the algorithms explicitly compute triangular factors of H^{-1}. This suggests that they might not be backward stable, even for positive definite H. An possible exception which computes triangular factors of H is given in [13]; the stability properties of this algorithm are not known.

Two $O(n \log^2(n))$ algorithms based on Padé approximation were given in [4]. One of them applies to completely general indefinite Hankel and nonsymmetric Toeplitz matrices. The numerical properties of this method have not been investigated, though it seems that

there is potential for numerical instability. Other authors have also developed algorithms based on the connection of structured matrices with Padé approximation, [8].

In §3, we propose a new algorithm for the Cholesky factorization of a positive definite Hankel matrix. The algorithm is analogous to the Schur algorithm for the factorization of a Toeplitz-like matrix. Instead of using hyperbolic rotations or Σ-orthogonal transformations, it uses orthogonal symplectic matrices to manipulate generators for a Hankel-like displacement. It is proven to be numerically stable in §4. We generalize the algorithm to matrices of higher displacement rank in §6. In §6, we also give a condensed error analysis of the general algorithm using results from §4.

To some extent, investigating the stability properties of a fast solver for positive definite Hankel matrices is a theoretical concern. It was shown in [16] that

$$\kappa_2(H) = \|H\|_2 \|H^{-1}\|_2 \geq 3 \cdot 2^{n-6}$$

if H is positive definite. There are even more discouraging estimates which suggest that for positive definite Hankel matrices the condition number grows asymptotically as 4^n. Thus we do not expect to find accurate solutions even for relatively small positive definite Hankel systems of equations. Nevertheless, the stability properties of the algorithm are theoretically interesting and provide insight into the stability of methods for more general positive definite and indefinite Hankel-like matrices.

For the indefinite case, look-ahead algorithms have been proposed to improve the stabilty in the presence of ill-conditioned leading submatrices, [3, 12]. The algorithm of [8] is also a look-ahead algorithm. Except for [3], most of the applicable algorithms have been developed using polynomials rather than matrices. A displacement structure approach leads to a surprisingly simple block factorization step which can be derived in matrix notation in a few lines. We describe the basic block step in §3 and look-ahead refinements in §5.

When it is convenient, we use MATLAB notation to indicate submatrices. Thus the matrix $H(i:j, k:l)$ is the $(j-i+1) \times (l-k+1)$ block of H formed by taking only elements that are in rows i through j and columns k through l. The notation $H(:, j)$ indicates column j of H.

2 The Displacement

The Hankel matrix H is determined by $2n-1$ parameters. None of the results in this paper are strictly limited to Hankel matrices; everything will apply to the more general class of Hankel-like matrices. To define this class, we borrow and adapt an idea used in the study of Toeplitz matrices by introducing the Hankel displacement rank of a matrix. A symmetric matrix H is *Hankel-like* with Hankel displacement rank 2 whenever the displacement

$$\Delta_Z(H) \;:=\; ZH - HZ^{\mathrm{T}}$$

is rank 2. The matrix Z is the downshift matrix defined by $(Z)_{ij} = 1$ when $i - j = 1$ and $(Z)_{ij} = 0$ otherwise. The notion of a Hankel-like matrix can be generalized to larger ranks. We will consider such generalizations in §6. For the moment, we deal exclusively with displacement rank 2 Hankel-like matrices.

For any matrix, the displacement is skew-symmetric. For a Hankel matrix

$$\Delta_Z(H) := \begin{bmatrix} 0 & -h_0 & -h_1 & \cdots & -h_{n-2} \\ h_0 & & & & \\ h_1 & & & & \\ \vdots & & & & \\ h_{n-2} & & & & \end{bmatrix}.$$

We will represent a Hankel-like matrix by means of its *generators*, the columns of a matrix A such that

$$\Delta_Z(H) = AJA^{\mathrm{T}} \tag{2}$$

where

$$J := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

A real skew-symmetric matrix of rank 2 can always be decomposed as (2). Given $\Delta_Z(H)$, the decomposition can be computed in a stable manner using a single 2×2 pivot step of a skew-symmetric elimination procedure, [5]. Further, a real skew-symmetric always has even rank. Thus if the displacement is nonzero then A has full rank.

The generators are not unique. For a Hankel matrix it is possible to choose generators with the simple form

$$A^{\mathrm{T}} = \begin{bmatrix} \sqrt{h_0} & 0 & 0 & \cdots & 0 \\ 0 & \sqrt{h_0} & h_1/\sqrt{h_0} & \cdots & h_{n-2}/\sqrt{h_0} \end{bmatrix}. \tag{3}$$

A generator matrix A for which the $(1,2)$ element equals zero is said to be in *proper form.*

The displacement operator, $\Delta_Z(\cdot)$, is different from the better known Toeplitz-like displacements in three algorithmically significant respects.

1. The Hankel displacement is skew-symmetric. This fact determines the classes of transformations that we can use in implementing a Schur-type factorization algorithm.

2. The proper form generators of A are not unique, even ignoring sign changes or scaling. For a Toeplitz matrix, the proper form generators are unique up to sign changes. This difference has implications for the numerical stability of factorizations: for a given Hankel-like matrix, there exists a generator matrix in proper form with arbitrarily large norm. Hence to get a stable algorithm, we must be very careful in the choice of generators. An algorithm that is forced forced to work with generators that are much larger than the matrix to be factored is likely to be unstable.

3. The linear displacement operator $\Delta_Z(\cdot)$ has a non-trivial null space. This means that it is not possible to reconstruct H given only $\Delta_Z(H)$. A factorization algorithm cannot work solely with the generators of H; it must incorporate some additional information. We will show that H can be reconstructed from $\Delta_Z(H)$ and from the last column of H.

We will elaborate on each of these points in turn.

A matrix S that satisfies $SJS^{\mathrm{T}} = J$ is a 2×2 *symplectic* matrix. General $r \times r$ symplectic matrices are defined by the relation

$$S \begin{bmatrix} 0 & -I_{r/2} \\ I_{r/2} & 0 \end{bmatrix} S^{\mathrm{T}} = \begin{bmatrix} 0 & -I_{r/2} \\ I_{r/2} & 0 \end{bmatrix}.$$

We will make use of these more general symplectic matrices in §6. We will use 2×2 symplectic matrices in the factorization of displacement rank 2 Hankel-like matrices. If S is symplectic then

$$ASJS^{\mathrm{T}}A^{\mathrm{T}} = AJA^{\mathrm{T}}$$

and AS is an alternate set of generators for $\Delta_Z(H) = AJA^{\mathrm{T}}$. In devising a factorization algorithm we are free to apply any symplectic transformation to the generator matrix to compute an alternate set of generators for the same displacement.

Note that

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix} = \begin{bmatrix} 0 & -(ad-bc) \\ (ad-bc) & 0 \end{bmatrix}.$$

In the 2×2 case, if $\det(S) = 1$ then S is symplectic. Symplectic transformations that will later be of particular algorithmic interest are the plane rotations

$$Q = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

with $c^2 + s^2 = 1$ and the diagonal scaling scaling matrices

$$D = \begin{bmatrix} d & 0 \\ 0 & \frac{1}{d} \end{bmatrix}$$

for $d \neq 0$.

If $A \neq 0$ so that the displacement rank is exactly 2 and A must have full rank then

$$AJA^{\mathrm{T}} = BJB^{\mathrm{T}}$$

implies that

$$A^{\mathrm{T}} = \left(J^{\mathrm{T}}A^{\dagger}BJ\right) B^{\mathrm{T}}.$$

and

$$J = (J^{\mathrm{T}}A^{\dagger}BJ)J(J^{\mathrm{T}}A^{\dagger}BJ)^{\mathrm{T}}$$

so that $J^{\mathrm{T}}A^{\dagger}BJ) = 1$ is symplectic. It follows that for a particular fixed choice of generator matrix A the set of equivalent generator matrices representing $\Delta_Z(H)$ can be characterized by $B = AS$ where S is symplectic.

Consider the proper form generators

$$A = \begin{bmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{bmatrix}$$

where a_{11} is a scalar and a_{21} and a_{22} are vectors. In general, if A is in proper form and $a_{11} \neq 0$, then the set of equivalent proper form generators representing the displacement has the form

$$B = \begin{bmatrix} b_{11} & 0 \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} d & 0 \\ l & \frac{1}{d} \end{bmatrix} \tag{4}$$

for any l and any $d \neq 0$. The assumption that $a_{11} \neq 0$ can be justified in the positive definite case by the fact that if $a_{11} = 0$ then the $(1,1)$ element of H must be zero. This is not possible for a positive definite matrix.

To see that (4) captures all proper form generators note that any equivalent generator matrix can be represented as $B = AS$ for $\det(S) = 1$. Since B is assumed to be in proper form

$$\begin{bmatrix} b_{11} & 0 \end{bmatrix} = \begin{bmatrix} a_{11} & 0 \end{bmatrix} S.$$

Since $a_{11} \neq 0$, $S(1,1) = b_{11}/a_{11} = d$ and $S(1,2) = 0$. The complete form of S follows from the fact that to get $\det(S) = 1$ for a triangular matrix, we must have

$$S(2,2) = 1/S(1,1) = 1/d.$$

If $d = 1$, $1/d$ or l are very large, then $\|B\|$ will be much larger than $\|A\|$. Putting the generators in proper form does not directly guarantee a bound on their size. This is in contrast to the generators of a positive definite Toeplitz matrix for which the proper form generators are unique up to sign changes and can be bounded in a way that is useful for a numerical stability analysis, [2].

We now consider the null space of $\Delta_Z(\cdot)$. Define the permutation, P, so that its action on a vector is to reverse the order of the elements. Thus $P(i,j) = 1$ if $i + j = n + 1$ and $P(i,j) = 0$ otherwise. Then

$$PZ^{\mathrm{T}} = ZP$$

so that if

$$N = \sum_{i=0}^{n-1} n_i Z^i P$$

where $Z^0 = I$ then

$$\Delta_Z(N) = \sum_{i=0}^{n-1} n_i (ZZ^i P - Z^i P Z^{\mathrm{T}}) = \sum_{i=0}^{n-1} n_i (ZZ^i P - Z^i ZP) = 0.$$

Considering the form of N, we see that any Hankel matrix of the form (1) for which

$$h_0 = h_1 = \cdots h_{n-2} = 0$$

is in the null space of $\Delta_Z(\cdot)$.

To see that this is a complete characterization of the null space, note that the relation $ZN - NZ^{\mathrm{T}} = 0$ implies

$$N(i, j-1) = N(i-1, j)$$

for $2 \le i, j \le n$. Given the first row and last column of N, this recurrence gives the other elements. If $ZN - NZ^{\mathrm{T}} = 0$ then the first row of N must be zero except possibly for its last element. Thus the recurrence implies that if $\Delta_Z(N) = 0$ then

$$N = \sum_{i=0}^{n-1} n_i Z^i P = \begin{bmatrix} 0 & 0 & 0 & \cdots & n_1 \\ 0 & \iddots & & \iddots & n_2 \\ 0 & & \iddots & \iddots & \vdots \\ \vdots & \iddots & \iddots & & \vdots \\ n_1 & n_2 & \cdots & \cdots & n_n \end{bmatrix}.$$

Consequently the null space of $\Delta_Z(\cdot)$ is the set of Hankel matrices that are lower triangular with respect to the cross-diagonal.

Although H cannot be recovered from $\Delta_Z(H)$, it is determined uniquely by $\Delta_Z(H)$ together with $H(:, n)$. Since the first row of ZH is zero

$$H(1, 1 : n-1) = -\left[\Delta_Z(H)\right](1, 2 : n). \tag{5}$$

Thus we have the first row and the last column of H. By symmetry we also have the first column and last row. Given these elements, it is easy to see that the other elements can be obtained uniquely from the relation

$$[\Delta_Z(H)]_{i,j} = H(i-1, j) - H(i, j-1), \qquad 2 \le i, j \le n. \tag{6}$$

This is simply an indexed form of the definition $\Delta_Z(H) = ZH - HZ^{\mathrm{T}}$.

Finally, we note that (5) is algorithmically important. It provides a natural way to obtain most of the first row of H. This is precisely what is needed to obtain a row of the Cholesky factor in a factorization algorithm. If the generators are in proper form then we get the particularly simple formula

$$H(1, 1 : n-1) = a_{11} a_{22}^{\mathrm{T}}. \tag{7}$$

If C is the Cholesky factor of H then

$$C(1, 1 : n-1) = \sqrt{\frac{a_{11}}{a_{22}(1)}}\, a_{22}^{\mathrm{T}}. \tag{8}$$

3 Schur Type Algorithms

As with the Toeplitz displacement, the Schur complement of an arbitrary matrix has a displacement rank with respect to $\Delta_Z(\cdot)$ that is no larger than the displacement rank of the original matrix. Thus Hankel-like structure is preserved under the process of Schur complementation. This observation forms the basis of a class of fast Schur-type algorithms. At each stage of the triangular factorization the algorithms work with the generators of the current Schur complement together with the last column instead of with the full matrix.

Since any symplectic S can be used to transform the generators, many distinct algorithms are possible. In this section we will explain the basic ideas behind algorithms for both the positive definite and the indefinite cases. The algorithm for the positive definite case is chosen to be provably stable. More efficient variations are possible and it seems that some of these algorithms are likely to be stable in practice. Nevertheless, some possible algorithms are clearly unstable; we will illustrate the problem with one such algorithm in this section.

Suppose that H is real, symmetric and Hankel-like with

$$\Delta_Z(H) = AJA^{\mathrm{T}}.$$

Partition H as

$$H = \begin{bmatrix} H_{11} & H_{21}^{\mathrm{T}} \\ H_{21} & H_{22} \end{bmatrix} \tag{9}$$

A as

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \tag{10}$$

and Z as

$$Z = \begin{bmatrix} Z_{11} & 0 \\ Z_{21} & Z_{22} \end{bmatrix}.$$

We consider the Schur complement

$$H_S = H_{22} - H_{21}H_{11}^{-1}H_{21}^{\mathrm{T}}$$

and the elimination matrix

$$L = \begin{bmatrix} I & 0 \\ -H_{21}H_{11}^{-1} & I \end{bmatrix}.$$

Clearly

$$LZL^{-1}(LHL^{\mathrm{T}}) - (LHL^{\mathrm{T}})L^{-\mathrm{T}}Z^{\mathrm{T}}L^{\mathrm{T}} = LAJA^{\mathrm{T}}L^{\mathrm{T}} \tag{11}$$

and

$$LHL^{\mathrm{T}} = \begin{bmatrix} H_{11} & 0 \\ 0 & H_S \end{bmatrix}.$$

The matrix LZL^{-1} is lower triangular and its $(2,2)$ block is LZL^{-1} is just Z_{22}. It follows that

$$Z_{22}H_S - H_S Z_{22}^{\mathrm{T}} = (A_2 - H_{21}H_{11}^{-1}A_1)J(A_2 - H_{21}H_{11}^{-1}A_1)^{\mathrm{T}}. \tag{12}$$

Since Z_{22} is a shift matrix, this shows that H_S has the same sort of displacement structure as H and

$$A_S = A_2 - H_{21}H_{11}^{-1}A_1. \tag{13}$$

The formula (13) will be the starting point in deriving a stable algorithm for positive definite matrices. It also forms a direct basis for a look-ahead algorithm for indefinite matrices. The look-ahead algorithm is recursive and, neglecting details about the computation of the look-ahead step size that we will cover in §5, it can be described as follows.

Algorithm 1 (Block Schur Algorithm) Let $n_S = n$ and

$$\Delta_Z(H) = AJA^{\mathrm{T}}.$$

Let r be the last column of H. Start with $L = I$ and $D = 0$.

1. While $n_S > 0$:
2. Let H_S be the current $n_S \times n_S$ Schur complement with generators A and last column r. Find an appropriate look-ahead step size, m, and let H_S and A be partitioned as (9) and (10) where H_{11} is $m \times m$.
3. Compute H_{11} and H_{21} from A and r using (6).
4. Let

   $$L(n - n_S + m : n, n - n_S + 1 : n - n_S + m) \leftarrow H_{21}H_{11}^{-1}$$

   and

   $$D(n - n_S + 1 : n - n_S + m, n - n_S + 1 : n - n_S + m) \leftarrow H_{11}.$$

5. Update A and r by

   $$A \leftarrow A_2 - H_{21}H_{11}^{-1}A_1, \qquad r \leftarrow r(m+1 : n_S) - H_{21}H_{11}^{-1}r(1 : m)$$

   and let

   $$n_S \leftarrow n_S - m. \blacksquare$$

Neglecting numerical errors, the algorithm computes lower triangular L and block diagonal D such that $H = LDL^{\mathrm{T}}$. The sizes of the diagonal blocks in D are the look-ahead step sizes m chosen in each iteration of the algorithm. Given (12), it is trivial to prove that the algorithm correctly computes the decomposition. Consider the first step of the algorithm for

which $H_S = H$. Step 4 is a block elemination step with an $m \times m$ pivot. Step 5 uses (12) a direct implementation of the formula for a Schur complement to get the generators and last column of the Schur complement of H. By returning to step 1, the algorithm recursively computes L_S and D_S such that

$$H_{22} - H_{21}H_{11}^{-1}H_{21}^{\mathrm{T}} = L_S D_S L_S^{\mathrm{T}}.$$

Thus

$$LDL^{\mathrm{T}} = \begin{bmatrix} I & 0 \\ H_{21}H_{11}^{-1} & L_S \end{bmatrix} \begin{bmatrix} H_{11} & 0 \\ 0 & D_S \end{bmatrix} \begin{bmatrix} I & H_{11}^{-1}H_{21}^{\mathrm{T}} \\ 0 & L_S^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{21}^{\mathrm{T}} \\ H_{21} & L_S D_S L_S^{\mathrm{T}} + H_{21}H_{11}^{\mathrm{T}}H_{21}^{\mathrm{T}} \end{bmatrix} = H. \tag{14}$$

This is a familiar and natural recursive description of block Gaussian elimination. It is easy to check from the indices in step 4 that the algorithm fills in $H_{21}H_{11}^{-1}$ and H_{11} into the blocks of L and D in a manner consistent with (14).

For the indefinite case, we will choose a look-ahead step size with the intent of keeping $\|H_{21}H_{11}^{-1}\|$ small to prevent any significant growth in the size of the generators when applying (12) directly. For reasons of stability, it is necessary to keep this quantity from becoming too large in the application of any type of block elimination algorithm, [10]. While we do not give a proof, experimental results suggest that this criterion is also sufficient to give a stable algorithm. We give further details on the implementation of the algorithm and its performance in §5.

For the case in which H is positive definite and we wish to compute its Cholesky factor we consider the paritioning

$$H = \begin{bmatrix} h_{11} & h_{21}^{\mathrm{T}} \\ h_{21} & H_{22} \end{bmatrix} = \begin{bmatrix} h_0 & h_{21}^{\mathrm{T}} \\ h_{21} & H_{22} \end{bmatrix} \tag{15}$$

where $h_{11} = h_0$ is a scalar. This corresponds to the case $m = 1$ and Algorithm 1 will use the simple update

$$\hat{A}_2 = A_2 - \frac{h_{21}}{h_0} A_1 \tag{16}$$

where A_1 is a row vector. The scalar h_0 and the vector h_{21} can be obtained by using the relation

$$\begin{bmatrix} h_0 \\ h_{21}(1:n-2) \end{bmatrix} = A_2 J A_1^{\mathrm{T}}$$

or by using (7) if the generator matrix is in proper form. Thus step 3 of Algorithm 1 is trivial and we get a simple $O(n^2)$ algorithm to generate the rows of the Cholesky factor of H in sequence.

However, if $\|h_{21}/h_0\|$ is large then (16) could result in significant generator growth. The following example shows that $\|h_{21}/h_0\|$ can be large for a positive definite Hankel-like matrix and that this can negatively impact the stability of the 1×1 pivot version of Algorithm 1.

Example 1 We construct a 5×5 Hankel matrix

$$H = K^{\mathrm{T}} K$$

from a Krylov matrix

$$K = \begin{bmatrix} b & Bb & B^2 b & B^3 b & B^4 b \end{bmatrix}$$

formed from $B = 4 \cdot diag(1, 2, 3, 4, 5)$ and

$$b^{\mathrm{T}} = 1 \times 10^{-5} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

It is easily verified that this Hankel matrix has generators

$$A^{\mathrm{T}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 5 \times 10^{-10} & 4.5 \times 10^{-9} & 4.95 \times 10^{-8} & 6.075 \times 10^{-7} \end{bmatrix}.$$

This matrix was formed from the elements of H by multiplying the generator matrix A^{T} shown in (3) from the left by

$$\begin{bmatrix} \frac{1}{\sqrt{h_0}} & 0 \\ 0 & \sqrt{h_0} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

We would hope that a stable algorithm would prevent generator growth and introduce errors not much larger than the size of the generators. Since $\|H\|^{1/2} \approx .55$ is not too different from $\|A\|$ this would give a small relative backward error.

However, factoring this matrix using a direct implementation of the scalar version of Algorithm 1 results in substantial generator growth. We get a computed Cholesky factor for which

$$\frac{\|C^{\mathrm{T}} C - H\|}{\|H\|} = 3.9 \times 10^{-5}$$

Although the example might seem somewhat contrived, it is worth noting that even starting with (3), the generators of the Schur complements would not have any obvious special structure when computed by (16). We have not found a Hankel matrix for which initial generators of the form (3) lead to instability in (16). ■

To deal with this problem we present a more sophisticated algorithm for the positive definite case. It will make use of transformations of the generators AS with S satisfying $SJS^{\mathrm{T}} = J$. As with Algorithm 1, it can be viewed as a recursive process in which we obtain the first row of the Cholesky factor of H from the generators and from the stored last column of H and then compute the generators and the last column of the Schur complement of H. The main difference is that rather than applying (16) directly, we will use transformations of the generators of the form AS to get a proper form in which the row of the Cholesky factor can be retrieved from (8) and in which (16) is guaranteed not to produce any significant generator growth.

The result of these modifications is the following algorithm.

Algorithm 2 (Positive Definite Schur Algorithm) Let $n_S = n$ and let

$$\Delta_Z(H) = AJA^{\mathrm{T}}.$$

Let r be the last column of H. Start with $C = 0$.

1. While $n_S > 0$:
2. Partition the current $n_S \times 2$ generator matrix A as

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{17}$$

   where a_{11} and a_{12} are scalars.

3. Scale

$$A \leftarrow A \begin{bmatrix} d & 0 \\ 0 & \frac{1}{d} \end{bmatrix}$$

   so that $\|A(:,1)\|_2 = \|A(:,2)\|_2$.

4. Put the generators into proper form using an orthogonal transformation

$$A \leftarrow A \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

   so that after the transformation $a_{12} = 0$ and $a_{11} > 0$.

5. Let

$$C(n - n_S + 1, n - n_S + 1 : n - 1) \leftarrow \sqrt{\frac{a_{11}}{a_{22}(1)}} a_{22}^{\mathrm{T}}$$

   and

$$C(n - n_S + 1, n) \leftarrow \frac{r(1)}{\sqrt{a_{11} a_{22}(1)}}.$$

6. Update A and r by

$$A \leftarrow \begin{bmatrix} a_{21} - a_{11} \begin{bmatrix} a_{22}(2 : n_S - 1)/a_{22}(1) \\ r(1)/(a_{22}(1) a_{11}) \end{bmatrix} & a_{22} \end{bmatrix},$$

$$r \leftarrow r(2 : n_S) - r(1) \begin{bmatrix} a_{22}(2 : n_S - 1)/a_{22}(1) \\ r(1)/(a_{22}(1) a_{11}) \end{bmatrix}$$

   and let

$$n_S \leftarrow n_S - 1. \blacksquare$$

The fact that the algorithm computes C such that $H = C^{\mathrm{T}}C$ is easy to establish. Steps 3 and 4 apply transformations AS such that $\det(S) = 1$ and $SJS^{\mathrm{T}} = J$. After step 4, we have proper form generators for the current Schur complement. Thus (8) holds. Step 5 uses this relation and the first element of r in an obvious way to get the complete first row of the Cholesky factor of the current Schur complement. The update for A in step 6 is just (16) applied to generators in proper form. The update for r is a direct application of the expression for the last column of the Schur complement of H. The process proceeds recursively on the Schur complement of H in the usual manner.

In practice, the complexity of the algorithm can be reduced by replacing the plane rotation with

$$\begin{bmatrix} 1 & 0 \\ l & 1 \end{bmatrix}, \qquad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ l & 1 \end{bmatrix}$$

where the pivoting is done as needed to ensure that $|l| < 1$. The analysis of §4 suggests the potential for an imbalance in the scaling of the columns of the generators, thus making the normalization in step 3 necessary to ensure the stability of the plane rotation. However, in practice this does not seem to occur very often if at all. The scaling does not seem to be necessary for stability in most cases.

As presented, Algorithm 2 requires about $8.5n^2$ flops. Neglecting the norm computation of Step 3, it requires approximately $6.5n^2$ flops. If $\|\cdot\|_\infty$ is used rather than $\|\cdot\|_2$ for the normalization, then the algorithm really can be implemented in $6.5n^2$ flops, although searches are required to find the largest element in each column of A. Replacing the plane rotation with a pivoted lower triangular transformation reduces the computation further to $4.5n^2$ flops. Eliminating step 3 altogether gives an algorithm that runs in $3.5n^2$ flops. While all of these variations seem to have robust stability properties, none of them suggest a means for deriving the simple and natural backward error bounds that can be found for Algorithm 2.

4 Error Analysis

In this section we will give a proof of the numerical stability of Algorithm 2. We will show that if C is the computed Cholesky factor of H then

$$\|C^{\mathrm{T}}C - H\| \leq \epsilon f(n)\|H\|$$

where ϵ is the machine precision and $f(n)$ is a polynomial in n. Rather than specifying $f(n)$ exactly, we will simply verify the existence of such a bound.

We work with partitionings of the form

$$A = \begin{bmatrix} a_1 & a_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{18}$$

where a_{11} and a_{12} are scalars. More generally let $A^{(0)} = A$ be the generator matrix for $H^{(0)} = H$ and let $A^{(k)}$ be the generator matrix for the Schur complement of H corresponding

to the partitioning (9) where H_{11} is $k \times k$. We assume that this Schur complement is a computed quantity determined by the computed generators and last column produced by Algorithm 2. We also assume that $A^{(k)}$ is the generator matrix at the start of the iteration prior to the application of the scaling in step 3. Thus $A^{(0)}$ is the initial unscaled generator matrix with which the algorithm starts the factorization.

We will use superscripts to denote subvectors of $A^{(k)}$ according to (18) so that

$$A^{(k)} = \begin{bmatrix} a_1^{(k)} & a_2^{(k)} \end{bmatrix} = \begin{bmatrix} a_{11}^{(k)} & a_{12}^{(k)} \\ a_{21}^{(k)} & a_{22}^{(k)} \end{bmatrix}. \tag{19}$$

We also partition the last column of the Schur complement as

$$r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}, \qquad r^{(k)} = \begin{bmatrix} r_1^{(k)} \\ r_2^{(k)} \end{bmatrix}.$$

The greater part of the work is in the analysis of a single generic iteration of the factorization. The errors produced during this iteration can be bounded independent of k. Consequently, through much of this section we will be able to drop the superscript (k).

The final bounds will be in terms of the norm

$$\|H\|_m = \max_{i,j} |H(i,j)|.$$

Unless otherwise noted, variables will refer to computed quantities. Exact versions of the generators, a row of the Cholesky factor and the last column of the current Schur complement will be written as $\breve{A}$, $\breve{c}$ and $\breve{r}$. We will freely ignore $O(\epsilon^2)$ terms without noting the fact in each equation. Subscripted variables ϵ_i always satisfy $|\epsilon_i| \leq \epsilon$ where ϵ is the machine precision. A subscripted matrix D_i will represent a diagonal matrix with elements satisfying

$$|D_i(j,j)| \leq 1.$$

Let $H_Z(A,r)$ be the operator taking A and r to the Hankel-like matrix defined uniquely by the generator matrix A and the last column r. Thus

$$ZH_Z(A,r) - H_Z(A,r)Z^{\mathrm{T}} = AJA^{\mathrm{T}}$$

and $[H_Z(A,r)](1:n,n) = r$. The relation (6) which suffices to compute H from the displacement, last column and first row shows that $H_Z(A,r)$ is uniquely defined.

An outline of the analysis is as follows.

1. We use an inductive argument to show that it is sufficient to analyze the stability of a single step of the algorithm.

2. We give a suitable bound on the generators produced by each iteration of the algorithm. If we are not too worried about achieving the tightest possible backward error bound, we can write the generator bound in a way that does not depend on k. This makes it possible to give bounds on a single step of elimination that are independent of k. To simplify the notation by dropping k as an index and consider a generic, unspecified iteration of the algorithm.

3. We show that small errors in the displacment and the last column correspond to small errors in the Hankel-like matrices.

4. We show that the transformations in steps 3 and 4 produce proper form generators corresponding to a Hankel-like matrix that is close to the Hankel-like matrix with which the current iteration started.

5. We show that the computed generators of the Schur complement of H and the computed row of the Cholesky factor are part of a stable factorization step. This completes the induction, proving the stability of the algorithm.

4.1 Induction

Let c^{T} be the computed first row of the Cholesky factor of $H = H_Z(A, r)$ and let A_S and r_S be the computed generators and last column of the Schur complement of H. For the moment we assume without proof that a single step of Algorithm 2 on a positive definite Hankel-like matrix of size n gives

$$cc^{\mathrm{T}} + \begin{bmatrix} 0 & 0 \\ 0 & H_Z(A_S, r_S) \end{bmatrix} - H = E \tag{20}$$

where $\|E\|_m \leq \epsilon f_1(n)\|H\|_m$ for some polynomial $f_1(n)$ and where A_S and r_S are the computed generators and last column of the Schur complement of H. If Algorithm 2 continues recursively to compute C_S such that

$$C_S^{\mathrm{T}} C_S - H_Z(A_S, r_S) = E_S$$

then

$$cc^{\mathrm{T}} + \begin{bmatrix} 0 & 0 \\ 0 & C_S^{\mathrm{T}} C_S \end{bmatrix} - H = E + \begin{bmatrix} 0 & 0 \\ 0 & E_S \end{bmatrix}.$$

Since

$$C = \begin{bmatrix} c(1) & c(2:n)^{\mathrm{T}} \\ 0 & C_S \end{bmatrix}$$

we have

$$C^{\mathrm{T}} C - H = E + \begin{bmatrix} 0 & 0 \\ 0 & E_S \end{bmatrix}.$$

Define

$$f(n) = f(n-1) + f_1(n), \qquad f(1) = 2 \tag{21}$$

so that

$$f(n) = 2 + \sum_{j=2}^{n} f_1(j)$$

and $f(n)$ is a polynomial in n. Note that if H is 1×1 then $r(1) = H$ and $c = (1+\epsilon_1)\sqrt{H}$. Thus for the 1×1 case

$$\|C^{\mathrm{T}}C - H\|_m = |c^2 - H| = 2\epsilon_1|H|$$

or

$$\|C^{\mathrm{T}}C - H\|_m \leq \epsilon f(1)\|H\|_m.$$

Inductively, if

$$\|E_S\|_m \leq \epsilon f(n-1)\|H_Z(A_S, r_S)\|_m \leq \epsilon f(n-1)\|H\|_m$$

then

$$\|C^{\mathrm{T}}C - H\|_m \leq \|E\|_m + \left\| \begin{bmatrix} 0 & 0 \\ 0 & E_S \end{bmatrix} \right\|_m \leq \epsilon f_1(n)\|H\|_m + \epsilon f(n-1)\|H\|_m = \epsilon f(n)\|H\|_m.$$

Thus to show that Algorithm 2 is stable, it suffices to show that (20) holds for some polynomial $f_1(n)$.

4.2 Bounding the Generators

We have already seen with Example 1 that generator growth has the potential to cause instability in a Schur-type algorithm. Neither the scaling in step 3 nor the plane rotation in step 4 can cause a problem. There is potential for growth only in step 6. However, we will show that if it does occur it will be harmless. While there is no obvious and direct way to bound $\|A\|$ produced by step 6, it is easy to show that if we do get a large value for $\|A\|$, it will be purely a result of poor scaling. The rescaling in step 3 will always reduce $\|A\|$ to a managable size in the next iteration without harming the stability of the factorization.

To make this more precise, we will show that none of the transformations in Algorithm 2 can significantly increase

$$\|a_1\|_2 \cdot \|a_2\|_2.$$

If this quantity is not very large, then the scaling in step 3 will reduce $\|A\|$ to a reasonable magnitude.

The following lemma refers to exact rather than computed quantities.

Lemma 1 *For $A^{(k)}$ at the beginning of an iteration in Algorithm 2*

$$\|a_1^{(k)}\|_2 \cdot \|a_2^{(k)}\|_2 \leq \|a_1^{(0)}\|_2 \cdot \|a_2^{(0)}\|_2 + \|C(1:k,:)\|_F^2$$

where $C^{\mathrm{T}}C = H$ is the Cholesky factorization of H.

Proof: The proof is by induction. Trivially, we see that the inequality holds for $k = 0$ at the start of the first iteration of the algorithm. Assume that

$$\|a_1^{(k-1)}\|_2 \cdot \|a_2^{(k-1)}\|_2 \leq \|a_1^{(0)}\|_2 \cdot \|a_2^{(0)}\|_2 + \|C(1:k-1,:)\|_F^2.$$

The scaling in step 3 does not change this quantity. After the scaling

$$\|a_1^{(k-1)}\|_2 = \|a_2^{(k-1)}\|_2.$$

If $\hat{a}_1^{(k-1)}$ and $\hat{a}_2^{(k-1)}$ denote the generator vectors after the application of the rotation in step 4 then

$$\begin{aligned}
\|\hat{a}_1^{(k-1)}\|_2^2 \cdot \|\hat{a}_2^{(k-1)}\|_2^2 &= \left(ca_1^{(k-1)} + sa_2^{(k-1)}\right)^{\mathrm{T}} \left(ca_1^{(k-1)} + sa_2^{(k-1)}\right) \cdot \\
&\quad \left(-sa_1^{(k-1)} + ca_2^{(k-1)}\right)^{\mathrm{T}} \left(-sa_1^{(k-1)} + ca_2^{(k-1)}\right) \\
&= \left(\|a_1^{(k-1)}\|_2^2 + 2cs\left(a_1^{(k-1)}\right)^{\mathrm{T}} a_2^{(k-1)}\right) \cdot \\
&\quad \left(\|a_1^{(k-1)}\|_2^2 - 2cs\left(a_1^{(k-1)}\right)^{\mathrm{T}} a_2^{(k-1)}\right) \\
&= \|a_1^{(k-1)}\|_2^4 - 4c^2s^2\left(\left(a_1^{(k-1)}\right)^{\mathrm{T}} a_2^{(k-1)}\right)^2 \\
&\leq \|a_1^{(k-1)}\|_2^2 \cdot \|a_2^{(k-1)}\|_2^2.
\end{aligned}$$

Finally, we consider the update to compute $A^{(k)}$. We have

$$a_2^{(k)} = \hat{a}_{22}^{(k-1)}$$

and

$$a_1^{(k)} = \hat{a}_{21}^{(k-1)} - \hat{a}_{11}^{(k-1)} \begin{bmatrix} \hat{a}_{22}^{(k-1)}(2:n-k)/\hat{a}_{22}^{(k-1)}(1) \\ r_1^{(k-1)}/(\hat{a}_{11}^{(k-1)}\hat{a}_{22}^{(k-1)}(1)) \end{bmatrix}.$$

Thus

$$\|a_1^{(k)}\|_2 \cdot \|a_2^{(k)}\|_2 \leq \|\hat{a}_{21}^{(k-1)}\|_2 \cdot \|\hat{a}_{22}^{(k-1)}\|_2 + \frac{\hat{a}_{11}^{(k-1)}}{\hat{a}_{22}^{(k-1)}(1)} \left\| \begin{bmatrix} \hat{a}_{22}^{(k-1)}(2:n-k) \\ r_1^{(k-1)}/\hat{a}_{11}^{(k-1)} \end{bmatrix} \right\|_2^2.$$

However since the plane rotation puts the generators in proper form

$$H^{(k-1)}(1,1) = \hat{a}_{11}^{(k-1)}\hat{a}_{22}^{(k-1)}(1)$$

and

$$C(k, k+1:n) = \sqrt{\frac{1}{\hat{a}_{11}^{(k-1)}\hat{a}_{22}^{(k-1)}(1)}} \begin{bmatrix} \hat{a}_{11}^{(k-1)}\hat{a}_{22}^{(k-1)}(2:n-k) & r_1^{(k-1)} \end{bmatrix}$$

so that

$$\begin{aligned}
\|a_1^{(k)}\|_2 \cdot \|a_2^{(k)}\|_2 &\leq \|\hat{a}_{21}^{(k-1)}\|_2 \cdot \|\hat{a}_{22}^{(k-1)}\|_2 + \|C(k, k+1:n)\|_2^2 \\
&\leq \|a_{21}^{(k-1)}\|_2 \cdot \|a_{22}^{(k-1)}\|_2 + \|C(k, k+1:n)\|_2^2 \\
&\leq \|a_1^{(0)}\|_2 \cdot \|a_2^{(0)}\|_2 + \|C(1:k-1,:)\|_F^2 + \|C(k, k+1:n)\|_2^2 \\
&= \|a_1^{(0)}\|_2 \cdot \|a_2^{(0)}\|_2 + \|C(1:k,:)\|_F^2. \ \blacksquare
\end{aligned}$$

As we have noted, strictly speaking these bounds hold only in the absence of numerical errors. However, since our interest in bounding the generators is to bound terms that are $O(\epsilon\|A\|)$, the difference caused by neglecting numerical errors in a bound on $\|A\|$ is $O(\epsilon^2)$ in the final backward error bound. We will apply the bound to the computed generators without further comment on these second order errors.

4.3 Errors on the Generators

We start with a simple lemma relating errors in the displacement $\Delta_Z(H)$ and the last column r to errors in H.

Lemma 2 *If*

$$BJB^{\mathrm{T}} - AJA^{\mathrm{T}} = E, \qquad s - r = f$$

then

$$\|H_Z(B,s) - H_Z(A,r)\|_m \leq \frac{n}{2}\|E\|_m + \|f\|_m.$$

Proof: We have already noted that $H_Z(A,r)$ is well defined and gives a unique Hankel-like matrix. The first row and last column of the Hankel-like matrix H are determined uniquely by its generators A and its last column r. Let

$$G := H_Z(B,s), \qquad H = H_Z(A,r).$$

If

$$\Delta := AJA^{\mathrm{T}} = \Delta_Z(H), \qquad \Gamma := BJB^{\mathrm{T}} = \Delta_Z(G)$$

then

$$H_{i,j-1} = H_{i-1,j} - \Delta_{ij} \tag{22}$$

implies that H is determined by A and r and $\breve{H}$ is determined by $\breve{A}$ and $\breve{r}$. We have $|\Delta_{ij} - \Gamma_{ij}| \leq \|E\|_m$ for each i and j and $|s_i - r_i| \leq \|f\|_m$ for each i. From (22), we have

$$G_{i,j-1} - H_{i,j-1} = G_{i-1,j} - H_{i-1,j} - \Gamma_{ij} + \Delta_{ij}$$

so that

$$|G_{i,j-1} - H_{i,j-1}| \leq |G_{i-1,j} - H_{i-1,j}| + \|E\|_m$$

and

$$|G_{i-1,j} - H_{i-1,j}| \leq |G_{i,j-1} - H_{i,j-1}| + \|E\|_m$$

These relations together with

$$\|G(1,1:n-1) - H(1,1:n-1)\|_m = \|G(1:n-1,1) - H(1:n-1,1)\|_m \leq \|E\|_m$$

and

$$\|G(1:n,n) - H(1:n,n)\|_m = \|G(n,1:n) - H(n,1:n)\|_m \leq \|f\|_m.$$

show that

$$\|G - H\|_m \leq \frac{n}{2}\|E\|_m + \|f\|_m.$$

The proof of this is simply a matter of using the bounds on the last row and column and first row and column of $G - H$ as boundary conditions and applying the worst case of the above inequality recurrences by assuming that they turn out to be equalities. The factor $n/2$ comes from using the recurrence that gives the lowest bound for a given element of $G - H$. The element with the largest bound appears roughly in the center of the matrix (depending on whether n is even or odd). ■

4.4 Backward Error Analysis of the Transformations

For any iteration k of Algorithm 2

$$\|a_1^{(k)}\|_2 \cdot \|a_2^{(k)}\|_2 \leq \|a_1^{(0)}\|_2 \cdot \|a_2^{(0)}\|_2 + \|C\|_F^2. \tag{23}$$

This bound is independent of k. Since our goal is to prove a bound of the form (20) without worrying too much about constants, we analyze a generic iteration of the algorithm for some k using (23). We drop the superscripts by letting generators $A = A^{(k)}$ and $r = r^{(k)}$. We let $H = H_Z(A, r)$.

In particular, in this section, we will analyze the stability of steps 3 and 4. We will show that these steps result in generators $\hat{A}$ (after step 3) and $\tilde{A}$ (after step 4) for which $\|H_Z(\hat{A}, r) - H\|_m$ and $\|H_Z(\tilde{A}, r) - H\|_m$ are small.

For step 3

$$\hat{A} = \text{fl}\left(\begin{bmatrix} da_1 & \frac{1}{d}a_2 \end{bmatrix}\right) = \begin{bmatrix} (I + \epsilon D_1)a_1 d & (I + \epsilon D_2)\frac{1}{d}a_2 \end{bmatrix}$$

where d is the computed quantity. Thus

$$\hat{A}J\hat{A}^{\mathrm{T}} = AJA^{\mathrm{T}} - \epsilon D_1 a_1 a_2^{\mathrm{T}} + \epsilon a_2 a_1^{\mathrm{T}} D_1 - \epsilon a_1 a_2^{\mathrm{T}} D_2 + \epsilon D_2 a_2 a_1^{\mathrm{T}}.$$

Using Lemma 1

$$\left\|\hat{A}J\hat{A}^{\mathrm{T}} - AJA^{\mathrm{T}}\right\|_F \leq 4\epsilon\left(\|a_1^{(0)}\|_2\|a_2^{(0)}\|_2 + \|C\|_F^2\right). \tag{24}$$

Lemma 2 implies that

$$\begin{aligned} \|H_Z(\hat{A}, r) - H\|_m &\leq \frac{n}{2}\left\|\hat{A}J\hat{A}^{\mathrm{T}} - AJA^{\mathrm{T}}\right\|_m \\ &\leq \frac{n}{2}\left\|\hat{A}J\hat{A}^{\mathrm{T}} - AJA^{\mathrm{T}}\right\|_F \\ &\leq 2n\epsilon\left(\|a_1^{(0)}\|_2\|a_2^{(0)}\|_2 + \|C\|_F^2\right). \end{aligned}$$

¿From basic results on the application of plane rotations, [17], we have

$$\begin{bmatrix} \tilde{a}_{11} & 0 \\ \tilde{a}_{21} & \tilde{a}_{22} \end{bmatrix} = \begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} \\ \hat{a}_{21} & \hat{a}_{22} \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} + E \tag{25}$$

for some c and s satisfying $c^2 + s^2 = 1$ and with E satisfying

$$\|E\|_F \leq 6\epsilon \|\hat{A}\|_F.$$

Consequently

$$\tilde{A}J\tilde{A}^{\mathrm{T}} = \hat{A}J\hat{A}^{\mathrm{T}} + EJ\tilde{A}^{\mathrm{T}} + \tilde{A}JE^{\mathrm{T}} + O(\epsilon^2)$$

and

$$\left\| \tilde{A}J\tilde{A}^{\mathrm{T}} - \hat{A}J\hat{A}^{\mathrm{T}} \right\|_F \leq 12\epsilon \|\hat{A}\|_F^2.$$

Thus

$$\begin{aligned} \|H_Z(\hat{A}, r) - H_Z(\tilde{A}, r)\|_m &\leq \frac{n}{2} \left\| \hat{A}J\hat{A}^{\mathrm{T}} - \tilde{A}J\tilde{A}^{\mathrm{T}} \right\|_m \\ &\leq \frac{n}{2} \left\| \hat{A}J\hat{A}^{\mathrm{T}} - AJA^{\mathrm{T}} \right\|_F \\ &\leq 6n\epsilon \|\hat{A}\|_F^2 \\ &\leq 12n\epsilon \left(\|a_1^{(0)}\|_2 \|a_2^{(0)}\|_2 + \|C\|_F^2 \right). \end{aligned}$$

The last inequality follows from the fact that after step 3

$$\|\hat{a}_1\|_2 = \|\hat{a}_2\| + O(\epsilon).$$

Combining these inequalities we get

$$\|H_Z(\tilde{A}, r) - H\|_m \leq 14n\epsilon \left(\|a_1^{(0)}\|_2 \|a_2^{(0)}\|_2 + \|C\|_F^2 \right). \tag{26}$$

4.5 Computing the Schur Complement

After step 4 we have proper form generators

$$\tilde{A} = \begin{bmatrix} \tilde{a}_{11} & 0 \\ \tilde{a}_{21} & \tilde{a}_{22} \end{bmatrix}$$

where $\tilde{a}_{11}$ is a scalar. Consider the exact first row $\breve{c}$ of the Cholesky factor of $H_Z(\tilde{A}, r)$ and the corresponding computed quantity c. We have

$$c = \begin{bmatrix} (I + 3\epsilon D_3)\tilde{a}_{22} \sqrt{\frac{\tilde{a}_{11}}{\tilde{a}_{22}(1)}} \\ (1 + 3\epsilon_1) \frac{r_1}{\sqrt{\tilde{a}_{11}\tilde{a}_{22}(1)}} \end{bmatrix} = \breve{c} + 3\epsilon D_4 c.$$

Thus

$$H_Z(\tilde{A},r) - cc^{\mathrm{T}} = \begin{bmatrix} 0 & 0 \\ 0 & \breve{H}_S \end{bmatrix} + E_1 \tag{27}$$

where $\|E_1\|_F \leq 6\epsilon\|c\|_2^2$ and where $\breve{H}_S$ is the exact Schur complement of $H_Z(\tilde{A},r)$.

Given computed A_S and r_S defined by

$$\begin{aligned} A_S &= \mathrm{fl}\left(\left[\tilde{a}_{21} - \tilde{a}_{11}\begin{bmatrix}\tilde{a}_{22}(2:n-1)/\tilde{a}_{22}(1) \\ r_1/(\tilde{a}_{22}(1)\tilde{a}_{11})\end{bmatrix} \quad \tilde{a}_{22}\right]\right) \\ &= \left[(I+\epsilon D_5)\left(\tilde{a}_{21} - \tilde{a}_{11}(I+2\epsilon D_6)\begin{bmatrix}\tilde{a}_{22}(2:n-1)/\tilde{a}_{22}(1) \\ r_1/(\tilde{a}_{22}(1)\tilde{a}_{11})\end{bmatrix}\right) \quad \tilde{a}_{22}\right] \end{aligned}$$

and

$$\begin{aligned} r_S &= \mathrm{fl}\left(r_2 - r_1\begin{bmatrix}\tilde{a}_{22}(2:n-1)/\tilde{a}_{22}(1) \\ r_1/(\tilde{a}_{22}(1)\tilde{a}_{11})\end{bmatrix}\right) \\ &= (I+\epsilon D_7)\left(r_2 - r_1(I+2\epsilon D_8)\begin{bmatrix}\tilde{a}_{22}(2:n-1)/\tilde{a}_{22}(1) \\ r_1/(\tilde{a}_{22}(1)\tilde{a}_{11})\end{bmatrix}\right) \end{aligned}$$

we seek a bound on $\|\breve{H}_S - H_Z(A_S,r_S)\|$. If $\breve{A}_S$ and $\breve{r}_S$ are the exact quantities computed from $\tilde{A}$ and r then

$$\begin{aligned} A_S J A_S^{\mathrm{T}} &= \breve{A}_S J \breve{A}_S^{\mathrm{T}} - \epsilon D_5 A_S(:,1)\tilde{a}_{22}^{\mathrm{T}} + \epsilon\tilde{a}_{22}A_S(:,1)^{\mathrm{T}}D_5 - \\ &\quad 2\epsilon D_6 c(2:n)c(1:n-1)^{\mathrm{T}} + 2\epsilon c(1:n-1)c(2:n)^{\mathrm{T}}D_6. \end{aligned}$$

Consequently using Lemma 1

$$\|A_S J A_S^{\mathrm{T}} - \breve{A}_S J \breve{A}_S^{\mathrm{T}}\|_F \leq 6\epsilon\left(\|a_1^{(0)}\|_2 \cdot \|a_2^{(0)}\|_2 + \|C\|_F^2\right). \tag{28}$$

Turning to the computation of r_S we have

$$\begin{aligned} r_S &= \breve{r}_S + \epsilon D_7 r_S - 2r_1\epsilon D_8 c(2:n)/c(1) \\ &= \breve{r}_S + \epsilon D_7 r_S + 2\epsilon D_8(r_S - r_2) \end{aligned}$$

so that

$$\|r_S - \breve{r}_S\|_2 \leq 3\epsilon\|r_S\|_2 + 2\epsilon\|r\| \leq 5\epsilon\|H\|_2 \leq 5\epsilon\|C\|_F^2. \tag{29}$$

Thus from Lemma 2 we get

$$\begin{aligned} \|\breve{H}_S - H_Z(A_S,r_S)\|_m &\leq \frac{n}{2}\|\breve{A}_S J \breve{A}_S^{\mathrm{T}} - A_S J A_S^{\mathrm{T}}\|_m + \|\breve{r}_S - r_S\|_m \\ &\leq 3n\epsilon\|a_1^{(0)}\|_2 \cdot \|a_2^{(0)}\|_2 + (3n+5)\epsilon\|C\|_F^2. \end{aligned} \tag{30}$$

We can write

$$\begin{aligned} \left\| cc^{\mathrm{T}} + \begin{bmatrix} 0 & 0 \\ 0 & H_Z(A_S,r_S)\end{bmatrix} - H\right\|_m &\leq \|H_Z(\tilde{A},\tilde{r}) - H\|_m + \|\breve{H}_S - H_Z(A_S,r_S)\|_m + \\ &\quad \left\| cc^{\mathrm{T}} + \begin{bmatrix} 0 & 0 \\ 0 & \breve{H}_S\end{bmatrix} - H_Z(\tilde{A},\tilde{r})\right\|_m \end{aligned}$$

to get a bound for (20) in terms of quantities that we have already bounded. Using (26), (27) and (30) we get

$$\begin{aligned}\left\| cc^{\mathrm{T}} + \begin{bmatrix} 0 & 0 \\ 0 & H_Z(A_S, r_S) \end{bmatrix} - H \right\| &\leq (17n+8)\epsilon \|C\|_F^2 + 17n\epsilon \|a_1^{(0)}\|_2 \|a_2^{(0)}\|_2 \\ &\leq (17n+8)n^{3/2}\epsilon \|H\|_m + 17n\epsilon \|a_1^{(0)}\|_2 \|a_2^{(0)}\|_2 \\ &\leq \left(17n^{5/2} + 8n^{3/2} + 17n\right)\epsilon \|H\|_m\end{aligned}$$

where the last inequality follows by assuming that $a_1^{(0)}$ and $a_2^{(0)}$ are chosen according to (3). Given the previous discussion of the induction step this is sufficient to prove the stability of the algorithm. By defining

$$f_1(n) = \left(17n^{5/2} + 8n^{3/2} + 17n\right)$$

and

$$f(n) = 2 + \sum_{j=2}^{n} f_1(j) \leq 2 + \sum_{j=2}^{n} 17j^3 + 8j^2 + 17j = \frac{17}{4}n^4 + \frac{67}{6}n^3 + \frac{67}{4}n - 40$$

we immediately get the following theorem.

Theorem 1 *Let C be computed by Algorithm 2 from initial generators of the form (3). Then*

$$\|C^{\mathrm{T}}C - H\|_m \leq \left(\frac{17}{4}n^4 + \frac{67}{6}n^3 + \frac{67}{4}n - 40\right)\epsilon \|H\|_m.$$

5 More on the Look-Ahead Algorithm

When implemented with a stable method for inverting H_{11}, Algorithm 1 forms the basis of a look-ahead algorithm. However, it is not complete. Further refinements are necessary for efficient implementation. To see why let H represent a Schur complement at some generic iteration of Algorithm 1 and consider the block partitioning (9). To determine an appropriate look-ahead step we try to choose H_{11} to be $m \times m$ with $\|H_{11}^{-1}H_{21}^{\mathrm{T}}\|$ of moderate size. Estimating this norm for each possible size of H_{11} until an appropriate look-ahead step size is found can be costly. If the search were implemented in a naive manner the algorithm would have to invert H_{11} for each tested step size. If the algorithm required look-ahead step sizes that were some significant fraction of n, the resulting algorithm could require $O(n^4)$ flops.

Fortunately, it is possible to guarantee that even in the worst case the number of required computations is $O(n^3)$ with $O(n^2)$ being more typical if the look-ahead step size can be bounded independent of n. The method uses the fact that for a displacement rank 2 Hankel-like matrix, $H_{11}^{-1}H_{21}^{\mathrm{T}}$ is Toeplitz-like with displacement rank 3. To demonstrate and exploit this fact, we use a general theorem describing a single step of a Schur algorithm for a rectangular matrix containing both Toeplitz-like and Hankel-like blocks.

Theorem 2 *For*

$$K = \begin{bmatrix} H \\ T \end{bmatrix}$$

consider the mixed Toeplitz-Hankel displacement equation

$$\begin{bmatrix} Z_n & 0 \\ 0 & I_m \end{bmatrix} K - \begin{bmatrix} I_n & 0 \\ 0 & Z_m \end{bmatrix} K Z_n^{\mathrm{T}} = \begin{bmatrix} B \\ D \end{bmatrix} J_{HT} \hat{G}^{\mathrm{T}}$$

where all the matrices are real, H is $n \times n$ and symmetric, T is $m \times n$, Z_n and Z_m are $n \times n$ and $m \times m$ shift matrices and J_{HT} is a general matrix that may include both a symmetric and a skew-symmetric component. Assume that $H(1,1) \neq 0$. Let

$$H = \begin{bmatrix} h_{11} & h_{21}^{\mathrm{T}} \\ h_{21} & H_{22} \end{bmatrix},$$

$$T = \begin{bmatrix} t_1 & T_2 \end{bmatrix},$$

$$B = \begin{bmatrix} b_1^{\mathrm{T}} \\ B_2 \end{bmatrix}$$

and

$$G = \begin{bmatrix} g_1^{\mathrm{T}} \\ G_2 \end{bmatrix}$$

where b_1^{T} and d_1^{T} are row vectors. Also let

$$K_S = \begin{bmatrix} H_{22} \\ T_2 \end{bmatrix} - \frac{1}{h_{11}} \begin{bmatrix} h_{21} \\ t_1 \end{bmatrix} h_{21}^{\mathrm{T}}$$

so that K_S is the Schur complement of K. If

$$B_S = B_2 - \frac{1}{h_{11}} h_{21} b_1^{\mathrm{T}}, \tag{31}$$

$$D_S = D - \frac{1}{h_{11}} Z_m t_1 b_1^{\mathrm{T}} \tag{32}$$

and

$$G_S = G_2 - \frac{1}{h_{11}} h_{21} g_1^{\mathrm{T}}, \tag{33}$$

then

$$\begin{bmatrix} Z_{n-1} & 0 \\ 0 & I \end{bmatrix} K_S - \begin{bmatrix} I_{n-1} & 0 \\ 0 & Z_m \end{bmatrix} K_S Z_{n-1}^{\mathrm{T}} = \begin{bmatrix} B_S \\ D_S \end{bmatrix} J_{HT} G_S^{\mathrm{T}}$$

where Z_{n-1} is a $(n-1) \times (n-1)$ shift matrix.

Proof: The proof is by direct verifcation that the displacement of K_S has the required factorization. Note that H is Hankel-like. If we start with the factorization $Z_nH - HZ_n = BJ_{HT}G^{\mathrm{T}}$ instead of $Z_nH - HZ_n = AJA^{\mathrm{T}}$ then as before (11) shows that

$$Z_{n-1}\left(H_{22} - \frac{1}{h_{11}}h_{21}h_{21}^{\mathrm{T}}\right) - \left(H_{22} - \frac{1}{h_{11}}h_{21}h_{21}^{\mathrm{T}}\right)Z_{n-1} = B_S J_{HT} G_S^{\mathrm{T}}$$

whatever the choice of J_{HT}. Thus if we define

$$T_S = T_2 - \frac{1}{h_{11}}t_1 h_{21}^{\mathrm{T}},$$

it is sufficient to verify that

$$T_S - Z_m T_S Z_{n-1}^{\mathrm{T}} = D_S J_{HT} G_S^{\mathrm{T}}.$$

Since

$$DJ_{HT}G^{\mathrm{T}} = \begin{bmatrix} t_1 & T_2 \end{bmatrix} - Z_m \begin{bmatrix} t_1 & T_2 \end{bmatrix} \begin{bmatrix} 0 & e_1^{\mathrm{T}} \\ 0 & Z_{n-1}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} t_1 & T_2 - Z_m T_2 Z_{n-1}^{\mathrm{T}} - Z_m t_1 e_1^{\mathrm{T}} \end{bmatrix}$$

we have

$$T_2 - Z_m T_2 Z_{n-1}^{\mathrm{T}} = DJ_{HT}G_2^{\mathrm{T}} + Z_m t_1 e_1^{\mathrm{T}}.$$

Consequently

$$\begin{aligned} T_S - Z_m T_S Z_{n-1}^{\mathrm{T}} &= DJ_{HT}G_2^{\mathrm{T}} + Z_m t_1 e_1^{\mathrm{T}} + \frac{1}{h_{11}} Z_m t_1 h_{21}^{\mathrm{T}} Z_{n-1}^{\mathrm{T}} - \frac{1}{h_{11}} t_1 h_{21}^{\mathrm{T}} \\ &= DJ_{HT}G_2^{\mathrm{T}} + \frac{1}{h_{11}}\left(Z_m t_1 \begin{bmatrix} h_{11} & h_{21}(1:n-2)^{\mathrm{T}} \end{bmatrix}\right) - \frac{1}{h_{11}} t_1 h_{21}^{\mathrm{T}}. \end{aligned}$$

¿From the displacement equation $t_1 = DJ_{HT}g_1$. Since the first row of Z_nH is zero the displacement also gives

$$b_1^{\mathrm{T}} J_{HT} G_2^{\mathrm{T}} = -\begin{bmatrix} h_{11} & h_{21}(1:n-2)^{\mathrm{T}} \end{bmatrix}.$$

Thus

$$\begin{aligned} T_S - Z_m T_S Z_{n-1}^{\mathrm{T}} &= DJ_{HT}G_2^{\mathrm{T}} - \frac{1}{h_{11}} Z_m t_1 b_1^{\mathrm{T}} J_{HT} G_2^{\mathrm{T}} - \frac{1}{h_{11}} DJ_{HT} g_1 h_{21}^{\mathrm{T}} \\ &= \left(D - \frac{1}{h_{11}} Z_m t_1 b_1^{\mathrm{T}}\right) J_{HT} \left(G_2^{\mathrm{T}} - \frac{1}{h_{11}} g_1 h_{21}^{\mathrm{T}}\right) \end{aligned}$$

where the last equality follows because the skew-symmetry of $Z_nH - HZ_n^{\mathrm{T}}$ implies that $b_1^{\mathrm{T}} J_{HT} g_1 = 0$. ■

Given a Hankel-like matrix H obtained as a Schur complement at some iteration in Algorithm 1 and partitioned as in (9), Theorem 2 gives a Schur algorithm for the matrix

$$K := \left[\begin{array}{cc} H_{11} & H_{21}^{\mathrm{T}} \\ H_{21} & H_{22} \\ \hline I_m & 0 \end{array}\right] = \left[\frac{H}{T}\right].$$

If H_{11} is $m \times m$ then after m steps of elimination on this matrix we have a Schur complement

$$K_S = \begin{bmatrix} H_{22} - H_{21}H_{11}^{-1}H_{21}^{\mathrm{T}} \\ -H_{11}^{-1}H_{21}^{\mathrm{T}} \end{bmatrix}. \tag{34}$$

Suppose that

$$Z_n H - H Z_n^{\mathrm{T}} = AJA^{\mathrm{T}}$$

for

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

where A_1 has m rows. Then the matrix K satisfies

$$\begin{bmatrix} Z_n & 0 \\ 0 & I_m \end{bmatrix} K - \begin{bmatrix} I_n & 0 \\ 0 & Z_m \end{bmatrix} K Z_n^{\mathrm{T}} = \left[\begin{array}{cc} A_1 J A_1^{\mathrm{T}} & A_1 J A_2^{\mathrm{T}} \\ A_2 J A_1^{\mathrm{T}} & A_2 J A_2^{\mathrm{T}} \\ \hline e_1 e_1^{\mathrm{T}} & 0 \end{array}\right] = \left[\frac{B}{D}\right] J_{HT} G^{\mathrm{T}}$$

where

$$B = \begin{bmatrix} A & 0_{n,1} \end{bmatrix}, \qquad D = \begin{bmatrix} 0_{m,2} & e_1 \end{bmatrix}, \qquad G = \begin{bmatrix} A & e_1 \end{bmatrix}$$

and

$$J_{HT} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Although the K_S shown in (34) is not the same as the one in Theorem 2 in which H_{11} is assumed to be 1×1, we can obtain generators for (34) by recursive application of (31), (32) and (33). This gives generators B_S, D_S and G_S for the displacement

$$\begin{aligned} \begin{bmatrix} B_S \\ D_S \end{bmatrix} J_{HT} G_S^{\mathrm{T}} \;=\; & \begin{bmatrix} Z_{n-m} & 0 \\ 0 & I_m \end{bmatrix} \begin{bmatrix} H_{22} - H_{21}H_{11}^{-1}H_{21}^{\mathrm{T}} \\ -H_{11}^{-1}H_{21}^{\mathrm{T}} \end{bmatrix} - \\ & \begin{bmatrix} I_{n-m} & 0 \\ 0 & Z_m \end{bmatrix} \begin{bmatrix} H_{22} - H_{21}H_{11}^{-1}H_{21}^{\mathrm{T}} \\ -H_{11}^{-1}H_{21}^{\mathrm{T}} \end{bmatrix} Z_{n-m}^{\mathrm{T}} \end{aligned}$$

Thus

$$H_{11}^{-1}H_{21}^{\mathrm{T}} - Z_m H_{11}^{-1}H_{21}^{\mathrm{T}} Z_{n-m}^{\mathrm{T}} = -D_S J_{HT} G_S^{\mathrm{T}} \tag{35}$$

and clearly $H_{11}^{-1}H_{21}^{\mathrm{T}}$ is a Toeplitz like matrix with displacement rank 3. The displacement operator is invertible with the inverse given by the formula

$$H_{11}^{-1}H_{21}^{\mathrm{T}} = \sum_{j=0}^{\min(m-1,n-m-1)} -(Z_m^j) D_S J_{HT} G_S^{\mathrm{T}} (Z_{n-m}^j)^{\mathrm{T}}. \tag{36}$$

The following algorithm computes generators for $H_{11}^{-1}H_{21}$.

Algorithm 3 For $n \times n$ H satisfying $Z_n H - H Z_n^{\mathrm{T}} = A J A^{\mathrm{T}}$, start with

$$B = \begin{bmatrix} A & 0_{n,1} \end{bmatrix}, \qquad D = \begin{bmatrix} 0_{m,2} & e_1 \end{bmatrix}, \qquad G = \begin{bmatrix} A & e_1 \end{bmatrix}$$

and let r be the last column of H. For $j = 1, 2, \ldots, m$:

1. For B, D and G partitioned as in Theorem 2, compute h_{11} and h_{21} from

$$h_{11} = -b_1^{\mathrm{T}} J_{HT} G_2(2,:)^{\mathrm{T}}$$

and

$$h_{21}(1 : n-j-1) = -b_1^{\mathrm{T}} J_{HT} G_2(3 : n-j+1,:)^{\mathrm{T}}, \qquad h_{21}(n-j) = r(1).$$

Thus h_{21} is a vector of length $n - j$. Compute t_1 from

$$t_1 = D_S J_{HT} g_1.$$

2. Update B, D and G using (31), (32) and (33).

3. Update r using

$$r \leftarrow r(2 : n-j+1) - \frac{1}{h_{11}} h_{21} r(1).$$

4. Let $j \leftarrow j + 1$ and go to 1.

At the end of this process we have D and G satisfying (35). The matrix $H_{11}^{-1} H_{21}^{\mathrm{T}}$ can be obtained from (36). ■

To estimate a step size for which $\|H_{11}^{-1} H_{21}^{\mathrm{T}}\|$ is suitably small, we can apply Algorithm 3 until a suitable step size is found. Clearly each element of $\Delta := D J_{HT} G^{\mathrm{T}}$ can be computed in a number of flops that depends on the displacement rank and not on n or m. At each iteration of Algorithm 3, the matrix $T := H_{11}^{-1} H_{21}^{\mathrm{T}}$ can be computed in $O(mn)$ flops using the relation

$$T_{i,j} = \Delta_{i,j} + T_{i,j}$$

which is satisfied by any matrix T such that

$$T - Z_m T Z_{n-m}^{\mathrm{T}} = \Delta.$$

Thus, even if the final look-ahead step size is $m = n - 1$, the extra computation involved in computing T for each iteration of Algorithm 3 is $O(n^3)$.

6 General Hankel-like Matrices

Consider the case in which the displacement $\Delta_Z(H) = ZH - HZ^{\mathrm{T}}$ has higher rank. In §2, we verified that if $\Delta_Z(H)$ has rank equal to 2 then it has a decompositon of the form (2). The following theorem shows that a similar decomposition is possible in the more general case.

Theorem 3 *Let Δ be a real, skew-symmetric $n \times n$ matrix with $r = rank(\Delta) > 0$. Then Δ has the following factorization.*

$$\Delta = \begin{bmatrix} B_1 & B_2 & \cdots & B_{r/2} \end{bmatrix} \begin{bmatrix} J & & & \\ & J & & \\ & & \ddots & \\ & & & J \end{bmatrix} \begin{bmatrix} B_1^{\mathrm{T}} \\ B_2^{\mathrm{T}} \\ \vdots \\ B_{r/2}^{\mathrm{T}} \end{bmatrix} \tag{37}$$

where the B_i are real $n \times 2$ matrices.

Proof: The decomposition can be constructed using a skew-symmetric Gaussian elimination procedure with 2×2 pivots of the form

$$\begin{bmatrix} 0 & -\delta_1 \\ \delta_1 & 0 \end{bmatrix}.$$

Further details may be found in [5]. ■

If the displacement $\Delta_Z(H)$ is available, it is possible to use $r/2$ steps of the skew-symmetric elimination procedure of [5] to get a low rank factorization of this form. However, because of potential sensitivity in the Schur complement to be truncated, [9], it is important to use a complete pivoting strategy.

It is easy to show that there exists a permutation P such that

$$P \begin{bmatrix} J & & & \\ & J & & \\ & & \ddots & \\ & & & J \end{bmatrix} P^{\mathrm{T}} = \begin{bmatrix} 0 & -I_{r/2} \\ I_{r/2} & 0 \end{bmatrix} = J_r$$

Thus Theorem 3 implies that for any real skew-symmetric Δ of rank r there exists a matrix

$$A := \begin{bmatrix} A_1 & A_2 \end{bmatrix}$$

where A_1 and A_2 are $n \times \frac{r}{2}$ matrices satisfying

$$\Delta_Z(H) = A J_r A^{\mathrm{T}}.$$

Note that this notation is different from the notation of (10).

6.1 Factorization Algorithms

We partition H as (9) and let

$$\begin{bmatrix} A_1 & A_2 \end{bmatrix\!\!\begin{bmatrix} \\ \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix\!\!\begin{bmatrix} \\ \end{bmatrix}.$$

If $H_S = H_{22} - H_{21}H_{11}^{-1}H_{21}^{\mathrm{T}}$ then the arguments from §3 show that

$$\Delta_Z(H_S) = A_S J_r A_S^{\mathrm{T}}$$

where

$$A_S = \begin{bmatrix} A_{21} - H_{21}H_{11}^{-1}A_{11} & A_{22} - H_{21}H_{11}^{-1}A_{11} \end{bmatrix}. \tag{38}$$

It is possible to compute H_{11} and H_{21} from A using (6). Thus Algorithm 1 can be adapted to an arbitrary displacement rank by using (38) in step 5 instead of (13). As before, this is potentially unstable; a small perturbation to the matrix from Example 1 can be chosen to give a displacement rank 4 Hankel-like matrix for which the modified version of Algorithm 1 fails. All of the results of §5 apply to the larger displacement rank case by adding extra J blocks to J_{HT}. Consequently a completely general look-ahead algorithm is not significantly different from the displacement rank 2 version.

As before, it is possible to get a provably stable algorithm for the positive definite case. In extending Algorithm 2, we note that any transformation S for which $SJ_rS^{\mathrm{T}} = J_r$ may be applied to A to get an equivalent set of generators $\hat{A} = AS$ such that $\Delta_Z(H) = \hat{A}J_r\hat{A}^{\mathrm{T}}$. The set of matrices satisfying $SJ_rS^{\mathrm{T}} = J_r$ are known as *symplectic* matrices. In order to prevent any possibility of generator growth, we will make use of the group of real orthogonal symplectic transformations

$$\left\{ Q \in \mathbb{R}^{r\times r} \,|\, Q = \begin{bmatrix} Q_{11} & Q_{12} \\ -Q_{12} & Q_{11} \end{bmatrix}, Q^{\mathrm{T}}Q = I_r, Q_{11}, Q_{12} \in \mathbb{R}^{\frac{r}{2}\times\frac{r}{2}} \right\}.$$

It is trivial to verify that any matrix of the specified form is both orthogonal and symplectic. Such matrices have applications in the study of Hamiltonian eigenvalue problems [11, 6].

Following [11], we introduce two types of elementary orthogonal symplectic matrices. The first is the *Householder symplectic* matrix of the form

$$R_H = \begin{bmatrix} R_1 & 0 \\ 0 & R_1 \end{bmatrix}$$

where

$$R_1 = I_r - \frac{2uu^{\mathrm{T}}}{u^{\mathrm{T}}u}.$$

The second is the *Jacobi symplectic* rotation of the form

$$R_J = \begin{bmatrix} C_1 & S_1 \\ -S_1 & C_1 \end{bmatrix}$$

where C_1 and S_1 are $\frac{r}{2} \times \frac{r}{2}$ diagonal matrices of the form

$$C_1 = \mathrm{diag}(\underbrace{1, \ldots, 1}_{k-1}, c, 1, \ldots, 1), \qquad S_1 = \mathrm{diag}(\underbrace{0, \ldots, 0}_{k-1}, s, 0 \ldots 0)$$

with real c and s satisfying $c^2 + s^2 = 1$. These elementary transformations may be computed to zero elements in a vector. In particular

$$\begin{bmatrix} x^{\mathrm{T}} & y^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} R_1 & 0 \\ 0 & R_1 \end{bmatrix} \begin{bmatrix} C_1 & S_1 \\ -S_1 & C_1 \end{bmatrix} = \begin{bmatrix} \hat{x}^{\mathrm{T}} & 0 \end{bmatrix}$$

if $y^{\mathrm{T}} R_1 = \|y\| e_1^{\mathrm{T}}$ and

$$\begin{bmatrix} x^{\mathrm{T}} R_1(:, r/2) & \|y\| \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} \sqrt{(x^{\mathrm{T}} R_1(:, r/2))^2 + \|y\|^2} & 0 \end{bmatrix}.$$

If $\hat{x}^{\mathrm{T}} R_2 = \|\hat{x}\| e_1^{\mathrm{T}}$ then

$$\sqrt{\|x\|^2 + \|y\|^2} \begin{bmatrix} e_1^{\mathrm{T}} & 0 \end{bmatrix} = \begin{bmatrix} x^{\mathrm{T}} & y^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} R_1 & 0 \\ 0 & R_1 \end{bmatrix} \begin{bmatrix} C_1 & S_1 \\ -S_1 & C_1 \end{bmatrix} \begin{bmatrix} R_2 & 0 \\ 0 & R_2 \end{bmatrix}.$$

The usual methods for computing plane rotations and Householder transformations, [7], ensure that these relations hold and that the transformations are numerically stable in a sense that is a direct generalization of (25). We will shortly have more to say on the stability of the transformations.

By substituting this combination of transformations for the single rotation of Algorithm 2, we get the following algorithm.

Algorithm 4 (Positive definite Hankel-like Schur Algorithm) Start with A_1 and A_2 such that

$$\Delta_Z(H) = \begin{bmatrix} A_1 & A_2 \end{bmatrix} J_r \begin{bmatrix} A_1^{\mathrm{T}} \\ A_2^{\mathrm{T}} \end{bmatrix}.$$

Let r be the last column of H. Let $n_S = n$ and $C = 0$.

1. While $n_S > 0$:

2. Let the current $n_S \times r$ generator matrix A be partitioned as

$$\begin{bmatrix} A_1 \mid A_2 \end{bmatrix} = \left[\begin{array}{cc|cc} a_{11} & a_{12}^{\mathrm{T}} & a_{13} & a_{14}^{\mathrm{T}} \\ a_{21} & A_{22} & a_{23} & A_{24} \end{array}\right]$$

   where a_{11} and a_{13} are scalars and a_{12}^{T} and a_{14}^{T} are both length $r/2 - 1$ row vectors.

3. Scale

$$A_1(:, 1) \leftarrow d A_1(:, 1), \qquad A_2(:, 1) \leftarrow \frac{1}{d} A_2(:, 1)$$

   where d is chosen so that $\|A_1(:, 1)\|_2 = \|A_2(:, 1)\|_2$.

4. Put the generators into proper form using a symplectic orthogonal transformation of the form

$$A \leftarrow A \begin{bmatrix} P_1 & 0 \\ 0 & P_1 \end{bmatrix} \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{bmatrix} P_2 & 0 \\ 0 & P_2 \end{bmatrix}.$$

so that after the transformation

$$A = \left[\begin{array}{cc|cc} a_{11} & 0 & 0 & 0 \\ a_{21} & A_{22} & A_{23} & A_{24} \end{array}\right]$$

with $a_{11} > 0$.

5. Let

$$C(n - n_S + 1, n - n_S + 1 : n - 1) \leftarrow \sqrt{\frac{a_{11}}{a_{23}(1)}} a_{23}^{\mathrm{T}}$$

and

$$C(n - n_S + 1, n) \leftarrow \frac{r(1)}{\sqrt{a_{11} a_{23}(1)}}.$$

6. Update A and r by

$$A \leftarrow \left[\begin{array}{cc|cc} a_{21} - a_{11} \begin{bmatrix} a_{23}(2 : n_S - 1)/a_{23}(1) \\ r(1)/(a_{23}(1)a_{11}) \end{bmatrix} & A_{22} & a_{23} & A_{24} \end{array}\right],$$

$$r \leftarrow r(2 : n_S) - r(1) \begin{bmatrix} a_{23}(2 : n_S - 1)/a_{23}(1) \\ r(1)/(a_{23}(1)a_{11}) \end{bmatrix}$$

and let

$$n_S \leftarrow n_S - 1. \blacksquare$$

Verifying that Algorithm 4 computes C such that $C^{\mathrm{T}}C = H$ is not substantially different from the verification of Algorithm 2. Steps 3 and 4 apply orthogonal symplectic transformations AS such that $SJ_rS^{\mathrm{T}} = J_r$. After step 4, the generators are in proper form. Using the obvious generalization of (7)

$$H(1, 1 : n - 1) = -a_{11} a_{23}^{\mathrm{T}}$$

and the first element of r we get the expressions for the first row of the Cholesky factor of the current Schur complement. The update for A in step 6 is just (38) applied to generators in proper form. The update for r is the same as in Algorithm 2. The process then proceeds recursively on the Schur complement of H in the usual manner.

6.2 Stability

The analysis of Algorithm 4 is a straightforward generalization of the analysis in §4. In particular if $H_Z(A,r)$ is the inverse displacement operator in the higher displacement rank case, the same inductive argument shows that if (20) holds then the algorithm is stable.

Lemma 1 can also be adapted to Algorithm 4.

Lemma 3 *Let $A^{(k)}$ be the generator matrix after the scaling of step 3 in iteration k of Algorithm 4. (i.e. the initial generator matrix is $A^{(0)}$ and the matrix generated by application k of step 3 is $A^{(k)}$). Then*

$$\begin{aligned} \|A^{(k)}\|_F^2 &\leq \|A^{(1)}\|_F^2 + 2\|C(1:k-1,:)\|_F^2 \\ &\leq \|A^{(0)}\|_F^2 + 2\|C(1:k-1,:)\|_F^2 \end{aligned}$$

where $C^{\mathrm{T}}C = H$ is the Cholesky factorization of H.

Proof: We consider the effects of steps 3, 4 and 6 on $\|A\|_F$. We have defined $A^{(k)}$ as being the matrix occuring after the scaling of step 3. Let $\hat{A}^{(k)}$ be the generator matrix after step 4 and $\tilde{A}^{(k)}$ be the shorter generator matrix after step 6. Since the algorithm uses an orthogonal symplectic transformation, step 4 does not change the norm. Thus

$$\|\hat{A}^{(k)}\|_F = \|A^{(k)}\|_F.$$

We consider the combination of step 6 with the scaling of step 3 in the next iteration. After step 4, the generators are in proper form. The arguments from Lemma 1 show that

$$\begin{aligned} \|\tilde{A}_1^{(k)}(:,1)\|_2 \|\tilde{A}_2^{(k)}(:,1)\|_2 &\leq \|\hat{A}_1^{(k)}(:,1)\|_2 \cdot \|\hat{A}_2^{(k)}(:,1)\|_2 + \|C(k,:)\|_2^2 \\ &\leq \frac{1}{2}(\|\hat{A}_1^{(k)}(:,1)\|_2^2 + \|\hat{A}_2^{(k)}(:,1)\|_2^2) + \|C(k,:)\|_2^2. \end{aligned}$$

Since $\|A_1^{(k+1)}(:,1)\|_2 = \|A_2^{(k+1)}(:,1)\|_2$ this implies that

$$\begin{aligned} \|A_1^{(k+1)}(:,1)\|_2^2 + \|A_2^{(k+1)}(:,1)\|_2^2 &= 2\|A_1^{(k+1)}(:,1)\|_2 \cdot \|A_2^{(k+1)}(:,1)\|_2 \\ &= 2\|\tilde{A}_1^{(k)}(:,1)\|_2 \cdot \|\tilde{A}_2^{(k)}(:,1)\|_2 \\ &\leq \|\hat{A}_1^{(k)}(:,1)\|_2^2 + \|\hat{A}_2^{(k)}(:,1)\|_2^2 + 2\|C(k,:)\|_2^2. \end{aligned}$$

Since steps 3 and 6 act only on these two columns

$$\begin{aligned} \|A^{(k+1)}\|_F^2 &\leq \|\hat{A}^{(k)}\|_F^2 + 2\|C(k,:)\|_2^2 \\ &\leq \|A^{(k)}\|_F^2 + 2\|C(k,:)\|_2^2. \end{aligned}$$

The second inequality follows from the easily verified fact that step 3 cannot increase the Frobenius norm of the generator matrix. The lemma follows inductively. ■

We keep the notation $\hat{A}$ and $\tilde{A}$ defined in the proof of the lemma and letting r and $\tilde{r}$ be the last column of the matrix defined at the corresponding steps of the algorithm. Since

step 3 operates on only two columns of A, it can be shown using the error expansion used to show (24) that

$$\begin{aligned}\left\| A^{(k+1)} J_r \left(A^{(k+1)}\right)^{\mathrm{T}} - \tilde{A}^{(k)} J_r \left(\tilde{A}^{(k)}\right)^{\mathrm{T}} \right\|_F &\leq 4\epsilon \left\| A_1^{(k+1)}(:,1) \right\|_2 \cdot \left\| A_2^{(k+1)}(:,1) \right\|_2 \\ &\leq 4\epsilon \|A^{(k+1)}\|_F^2 \\ &\leq 4\epsilon \left(\|A^{(0)}\|_F^2 + 2\|C(1:k,:)\|_F^2 \right).\end{aligned}$$

Lemma 2 implies that

$$\left\| H_Z \left(A^{(k+1)}, r^{(k+1)}\right) - H_Z \left(\tilde{A}^{(k)}, \tilde{r}^{(k+1)}\right) \right\|_m \leq 2n\epsilon \left(\|A^{(0)}\|_F^2 + 2\|C\|_F^2 \right). \tag{39}$$

Basic results on the application of plane rotations and Householder transformations from [17] give

$$\left\| \hat{A}^{(k+1)} J_r \left(\hat{A}^{(k+1)}\right)^{\mathrm{T}} - A^{(k+1)} J_r \left(A^{(k+1)}\right)^{\mathrm{T}} \right\|_F \leq c\epsilon \|A^{(k+1)}\|_F^2$$

where c is a constant. Thus

$$\left\| H_Z \left(\hat{A}^{(k+1)}, r^{(k+1)}\right) - H_Z \left(A^{(k+1)}, r^{(k+1)}\right) \right\|_m \leq \frac{c}{2}\epsilon \left(\|A^{(0)}\|_F^2 + 2\|C\|_F^2 \right). \tag{40}$$

Step 6 changes only two vectors, r and A. Further, the update only uses information from three vectors: $A_1(:,1)$, $A_2(:,1)$ and r. This is identical to the computation analyzed in §4.5. The arguments apply without modification to show that

$$\left\| \breve{H}_S - H_Z \left(\tilde{A}^{(k+1)}, \tilde{r}^{(k+1)}\right) \right\|_m \leq 3n\epsilon \|A^{(0)}\|_F^2 + (3n+5)\epsilon \|C\|_F^2 \tag{41}$$

where $\breve{H}_S$ is the exact Schur complement of $H_Z(\hat{A}^{(k+1)}, r^{(k+1)})$. Since the generators are in proper form after step 4, (27) applies with $\hat{A}^{(k+1)}$ taking the place of $\tilde{A}$. Using (39), (40), (41) and (27) and assuming that $\|A^{(0)}\|_F^2$ is not too much larger than $\|H\|$ gives an inequality of the form (20). The inductive argument of §4.1 then applies to show that Algorithm 4 is stable.

7 Conclusions

We have described displacement structure algorithms for the factorization of Hankel-like matrices. The algorithms for positive definite matrices use orthogonal symplectic transformations and admit a reasonably straightforward error analysis when a scaling step is used to prevent a potential imbalance in the scaling of the generator column vectors. We have not been able to show that this scaling step is necessary for stability; in practice it can often be skipped without introducing excessive backward errors.

The algorithm for indefinite matrices uses a very simple look-ahead step that follows easily from the displacement formulation. The approach is much simpler than others that

have been proposed in the literature. We have also suggested the size of $\|H_{11}^{-1}H_{21}^{\mathrm{T}}\|$ as a criterion for choosing a look-ahead step size and shown how to estimate this quantity efficiently using a Schur algorithm for a mixed Hankel-Toeplitz displacement. Experiments suggest that controlling the size of this quantity is sufficient to get stability. Results on block Gaussian elimination suggest that this is also likely to be necessary for stability.

References

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

[2] A. W. Bojanczyk, R. P. Brent, F. R. De Hoog, and D. R. Sweet. On the stability of the Bareiss and related Toeplitz factorization algorithms. *SIAM J. Matrix Anal. Appl.*, 16:40–57, 1995.

[3] A. W. Bojanczyk and G. Heinig. A multi-step algorithm for Hankel matrices. *Journal of Complexity*, 10:142–164, 1994.

[4] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *Journal of Algorithms*, 1:259–295, 1980.

[5] J. R. Bunch. A note on the stable decomposition of skew symmetric matrices. *Mathematics of Computation*, 158:475–480, 1982.

[6] R. Byers. A Hamiltonian QR algorithm. *SIAM Journal on Scientific and Statistical Computing*, 7:212–229, 1986.

[7] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, 3nd edition, 1996.

[8] M. H. Gutknecht. Stable row recurrences for the Padé table and generically superfast look-ahead solvers for non-hermitian Toeplitz systems. Research Report 92-14, Interdisciplinary Project Center for Supercomputing, Eidgenössische Technische Hochschule Züich, 1992.

[9] N. J. Higham. Analysis of the Cholesky decomposition of a semi-definite matrix. In M. G. Cox and S. J. Hammarling, editors, *Reliable Numerical Computation*. Oxford University Press, 1989.

[10] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, 1996.

[11] C. C. Paige and C. Van Loan. A Schur decomposition for Hamiltonian matrices. *Linear Algebra and its Applications*, 41:11–32, 1981.

[12] D. Pal and T. Kailath. Fast triangular factorization and inversion of Hankel and related matrices with arbitrary rank profile. *SIAM Journal on Matrix Analysis and Applications*, 15:451–478, 1994.

[13] J. L. Phillips. The triangular decomposition of Hankel matrices. *Mathematics of Computation*, 25:599–602, 1971.

[14] J. Rissanen. Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with application to factoring positive matrix polynomials. *Mathematics of Computation*, 27:147–154, 1973.

[15] W. F. Trench. An algorithm for inversion of finite Hankel matrices. *J. SIAM*, 13:1102–1107, 1965.

[16] E. Tyrtyshnikov. How bad are Hankel matrices? *Numerische Mathematik*, 67:261–269, 1994.

[17] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, England, 1965.