At

LBL-37690
UC-405
Preprint

Lawrence Berkeley Laboratory

UNIVERSITY OF CALIFORNIA

Physics Division

Mathematics Department

To be submitted for publication

Stable and Efficient Algorithms for Structured Systems of Linear Equations

M. Gu

September 1995

SCAN-9510078

CERN LIBRARIES, GENEVA

SW9542

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor The Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, or The Regents of the University of California.

Lawrence Berkeley National Laboratory
is an equal opportunity employer.

LBL-37690

Stable and Efficient Algorithms for Structured Systems of Linear Equations¹

Ming Gu

Department of Mathematics and Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

September 1995

¹Supported by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

Stable and Efficient Algorithms for Structured Systems of Linear Equations

Ming Gu*

Abstract

Recently Heinig shows that structured matrices such as the Toeplitz and Hankel matrices can be transformed into a new class of structured matrices called Cauchy-like matrices using the FFT or other trigonometric transforms, and partial pivoting can be performed on the Cauchy-like matrices. Gohberg, Kailath and Olshevsky show that a fast variation of Gaussian elimination with partial pivoting (GEPP) can be performed on the Cauchy-like matrices, and demonstrate numerically that this variation is numerically stable. Sweet and Brent show that the error growth in this variation could be much larger than would be encountered with straightforward GEPP in certain cases. In this paper, we present a modified algorithm that avoids such extra error growth and can perform a fast variation of Gaussian Elimination with Complete Pivoting (GECP). Our analysis shows that it is both efficient and numerically stable, provided that the element growth in the computed factorization is not large. We also present a more efficient variation of this algorithm and discuss implementation techniques that further reduce the execution time. Our numerical experiments show that this variation is highly efficient and numerically stable.

Keywords: displacement rank, error analysis, fast algorithms, generators, Hankel matrices, iterative refinement, pivoting, structured matrices, Toeplitz matrices.

1 Introduction

Let matrices Ω and $\Lambda \in \mathbf{R}^{n \times n}$ be given. The Sylvester type *displacement equation* for a matrix $M \in \mathbf{R}^{n \times n}$ is

$$\Omega \cdot M - M \cdot \Lambda = G \tag{1.1}$$

*Department of Mathematics and Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720. The author was supported in part by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

where $G = A \cdot B$, with $A \in \mathbf{R}^{n \times \alpha}$ and $B \in \mathbf{R}^{\alpha \times n}$. The matrix pair (A, B) (or the matrix G) is the *generator* of M with respect to Ω and Λ ; $\alpha \leq n$ is the *displacement rank* with respect to Ω and Λ if G has rank α ; and M is considered to possess a *displacement structure* with respect to Ω and Λ if $\alpha \ll n$. Such displacement equations first appeared in [17]; and the concept of displacement structure was first introduced in [19]. The most general form of displacement structure, which includes equation (1.1) as a special case, was introduced in [20].

1.1 Fast Algorithms for Structured Matrices

The coefficient matrices in many linear systems of equations arising from signal processing, control theory and interpolation applications often have such displacement structures. For example, the Cauchy-like matrix is a matrix of the following form (see [10, 15]):

$$C = \left(\frac{a_i^T \cdot b_j}{\omega_i - \lambda_j} \right)_{1 \leq i, j \leq n} \quad (a_i, b_j \in \mathbf{R}^\alpha),$$

here we assume that $\omega_i \neq \lambda_j$ for $1 \leq i, j \leq n$. Equivalently, we can define a Cauchy-like matrix to be the unique solution to the displacement equation

$$\Omega \cdot C - C \cdot \Lambda = A \cdot B$$

with

$$\Omega = \text{diag}(\omega_1, \dots, \omega_n), \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n) \quad \text{and} \quad A = \begin{pmatrix} a_1^T \\ \vdots \\ a_n^T \end{pmatrix}, \quad B = (b_1, \dots, b_n).$$

In particular, C is a Cauchy matrix if $a_i^T \cdot b_j = 1$ for all i and j . We note that while the rank of C can be as large as n , its displacement rank is at most α .

Other classes of structured matrices include the Toeplitz matrices and the Hankel matrices. A Toeplitz matrix T is a matrix whose entries are constant along every diagonal ($T = (t_{i-j})_{1 \leq i, j \leq n}$); and a Hankel matrix H is a matrix whose entries are constant along every anti-diagonal ($H = (h_{i+j-2})_{1 \leq i, j \leq n}$). These two classes of matrices are included in the larger class of Toeplitz-plus-Hankel matrices. A Toeplitz-plus-Hankel matrix is the sum of a Toeplitz and a Hankel matrix.

There are many fast algorithms that solve the Toeplitz (or Hankel, or Toeplitz-plus-Hankel) system of linear equations in $O(n^2)$ floating point operations, as opposed to $O(n^3)$ floating point operations normally required for a general dense matrix; there are also super-fast algorithms that require only $O(n \log^2 n)$ floating point operations. However, all these fast and super-fast algorithms are in general numerically unstable for indefinite systems. For discussions of some of these methods, see [3, 8, 22] and the references therein. Attempts to overcome this numerical instability result in

algorithms that could require (n^3) floating point operations in the worst case [4, 14, 22].

Recently, Heinig [15] shows that certain displacement equations the Toeplitz matrix satisfies allow transforming it via fast Fourier or trigonometric transforms into Cauchy-like matrices, which can in turn be inverted in $O(n^2)$ floating point operations; Heinig further shows that partial pivoting can be incorporated in the process of inversion. However, experiments show that the resulting algorithm can still be numerically unstable for ill-conditioned Toeplitz matrices. Gohberg, Kailath, and Olshevsky [10] further show that a Cauchy-like matrix can be rapidly triangular-factorized using Gaussian Elimination with Partial Pivoting (GEPP) in $O(n^2)$ floating point operations; and demonstrate numerically that the resulting algorithm for solving Toeplitz systems is numerically stable. They also show that Hankel matrix, and the Toeplitz-plus-Hankel matrix can also be transformed via fast trigonometric transforms into Cauchy-like matrices.

Sweet and Brent [23] have done an error analysis for the algorithms of [10]. They show that the error propagation of Algorithm GKO, the algorithm of choice in [10] for factorizing the Toeplitz matrix, depends not only on the magnitudes of the triangular factors L and U in the LU factorization of the corresponding Cauchy-like matrix, but also on the generator for this Cauchy-like matrix. They show that in some cases the generator can suffer large internal element growth and cause a corresponding growth in the backward and forward error; their results imply that Algorithm GKO is less numerically stable than the straightforward GEPP on the Cauchy-like matrix.

1.2 Main Results

In this paper, we show how to avoid such internal element growth in the generator when factorizing the Cauchy-like matrix; we demonstrate how to triangular-factorize this Cauchy-like matrix using a variation of Gaussian Elimination with Complete Pivoting (GECP) in $O(n^2)$ floating point operations (see §2). We compare a different choice of displacement equation for the Toeplitz and Toeplitz-plus-Hankel matrices with those in [10, 15] in terms of efficiency and numerical accuracy in factorizing the resulting Cauchy-like matrix; and based on our analysis and with this choice, we provide a new algorithm for factorizing a Toeplitz or Toeplitz-plus-Hankel matrix (see §3) that performs about 50% less floating point operations than Algorithm GKO of [10]. We perform an error analysis for fast Cauchy-like matrix factorization algorithms and show that this new algorithm is numerically stable, provided that the magnitude of the triangular factor U in the LU factorization is not large (see §4). We report numerical experiments to support these results (see §5).

We also discuss some implementation techniques that significantly reduce the amount of memory traffic during the execution of this new algorithm. Our numerical experiments indicate that they make the new algorithm up to a factor of 2 faster (see §5).

1.3 Overview

In §2 we review the fast algorithm of [10] for Cauchy-like matrices; we present a fast algorithm, Algorithm 2, that performs a variation of GECP on such matrices and avoids internal element growth in the generator; and we provide a variation of Algorithm 2 that is more efficient. In §3 we compare different choices of displacement equation for the Toeplitz and Toeplitz-plus-Hankel matrices in terms of efficiency and numerical accuracy in factorizing the resulting Cauchy-like matrix; based on Algorithm 2 and a new choice of displacement equation, we provide a new algorithm, Algorithm 4, for solving the Toeplitz and Toeplitz-plus-Hankel system of linear equations. In §4 we perform an error analysis for Algorithms 2 and 4. In §5 we present numerical experiments with Algorithm 4 and compare this algorithm with some other available algorithms. And in §6 we discuss some extensions, draw conclusions, and ask some open problems.

1.4 Notation and Conventions

For a matrix A , $|A|$ is the matrix of moduli of the $\{a_{i,j}\}$; $A_{p:q,s:k}$ is a submatrix of A that selects rows p to q of columns s to k ; $A_{:,s:k}$ and $A_{s:k,:}$ select s^{th} through k^{th} rows and columns, respectively; and when $s = k$, we replace $s : k$ by s . Without loss of generality we assume A to be real unless it is specified to be complex. Our discussion for real matrices generally carries over to the complex case.

We will use the *max norm*, the ∞ -*norm*, the *2-norm*, and the *Frobenius norm*:

$$\|A\|_{\max} = \max_{i,j} |a_{i,j}|, \quad \|A\|_{\infty} = \max_i \sum_j |a_{i,j}|, \quad \|A\|_2 = \max_{\|u\|_2=1} \|A \cdot u\|_2, \quad \|A\|_F = \sqrt{\sum_{i,j} |a_{i,j}|^2}.$$

For a matrix $A \in \mathbf{R}^{n \times m}$, the following inequalities hold:

$$\frac{\|A\|_F}{\sqrt{n \cdot m}} \leq \|A\|_{\max} \leq \|A\|_2 \quad \text{and} \quad \frac{\|A\|_2}{\sqrt{n}} \leq \|A\|_{\infty} \leq \sqrt{m} \cdot \|A\|_2. \quad (1.2)$$

P is a permutation matrix; and $P(j, k)$ denotes the permutation that interchanges the j^{th} and k^{th} rows of a matrix.

ϵ is the machine precision, and n is the order of the matrix to be factorized.

A *flop* is a floating-point operation $x \circ y$, where x and y are floating-point numbers and $\circ$ is one of $+$, $-$, $\times$, and $\div$. Taking the absolute value or comparing two floating-point numbers is also counted as a flop.

In our error analysis, we take the usual model of arithmetic:¹

$$\text{fl}(x \circ y) = (x \circ y)(1 + \eta), \quad (1.3)$$

¹This model excludes some CRAY machines that do not have a guard digit. Our error analysis still holds for such machines with a few easy modifications.

where $\text{fl}(x \circ y)$ is the floating point result of the operation $\circ$; and $|\eta| \leq \epsilon$. For simplicity, we ignore the possibility of overflow and underflow.

Let $\bar{A} = A - a \cdot b^T$, where A is a matrix and a and b are vectors, $\text{fl}(\bar{A})$ is the result of computing $\bar{A}$ in finite precision.

2 Gaussian Elimination with Pivoting for Cauchy-like Matrices

Given a matrix $C \in \mathbf{R}^{n \times n}$, the first step of Gaussian elimination is to zero-out the first column of C below the diagonal entry:

$$C = \begin{pmatrix} \gamma_1 & u^T \\ r & \tilde{C}_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ l & I \end{pmatrix} \cdot \begin{pmatrix} \gamma_1 & u^T \\ 0 & C^{(2)} \end{pmatrix}, \quad (2.4)$$

where γ_1 is the pivot; $l = r/\gamma_1$; and $C^{(2)} = \tilde{C}_2 - l \cdot u^T$ is the Schur complement of γ_1 . Gaussian elimination then recursively applies this step to $C^{(2)}$. At the end of this procedure, C is factored into $C = L \cdot U$, where L is a lower triangular matrix and U is an upper triangular matrix.

The following theorem shows that if C is a Cauchy-like matrix with displacement rank α , so is $C^{(2)}$. The algorithms of Gohberg, Kailath, and Olshevsky [10] are based on this theorem. More general forms of it appear in [10, 11, 12, 20], and a variation of it appears in [15].

Theorem 1 *Let matrix C in (2.4) satisfy the displacement equation*

$$\Omega \cdot C - C \cdot \Lambda = A \cdot B \quad (2.5)$$

with $\Omega = \text{diag}(\omega_1, \Omega_2)$ and $\Lambda = \text{diag}(\lambda_1, \Lambda_2) \in \mathbf{R}^{n \times n}$ diagonal; $A = \begin{pmatrix} a_1^T \\ \tilde{A}_2 \end{pmatrix} \in \mathbf{R}^{n \times \alpha}$; and $B = \begin{pmatrix} b_1 & \tilde{B}_2 \end{pmatrix} \in \mathbf{R}^{\alpha \times n}$. Assume that $\gamma_1 \neq 0$. Then $C^{(2)}$ satisfies the displacement equation

$$\Omega_2 \cdot C^{(2)} - C^{(2)} \cdot \Lambda_2 = A^{(2)} \cdot B^{(2)} \quad (2.6)$$

with $A^{(2)} = \tilde{A}_2 - l \cdot a_1^T \in \mathbf{R}^{(n-1) \times \alpha}$ and $B^{(2)} = \tilde{B}_2 - b_1 \cdot u^T / \gamma_1 \in \mathbf{R}^{\alpha \times (n-1)}$.

Hence one step of Gaussian elimination on the matrix C in Theorem 1 involves computing the first row and column γ_1 , r , and u of C from equation (2.5) and computing the vector l . To recursively apply this procedure to $C^{(2)}$, its generator $(A^{(2)}, B^{(2)})$ is then computed from equation (2.6).

2.1 Partial Pivoting

Partial pivoting is a strategy to reduce the element growth in the LU factorization. To perform partial pivoting on the first column of C , one finds its largest magnitude entry $(k_{\max}, 1)$, permute it to the $(1, 1)$ entry to get $P(1, k_{\max}) \cdot C$, and then applies one elimination step to $P(1, k_{\max}) \cdot C$. Let C be a Cauchy-like matrix satisfying equation (2.5). Then for every k ,

$$(P(1, k) \cdot \Omega \cdot P(1, k)) \cdot (P(1, k) \cdot C) - (P(1, k) \cdot C) \cdot \Lambda = (P(1, k) \cdot A) \cdot B,$$

where $(P(1, k) \cdot \Omega \cdot P(1, k))$ is again a diagonal matrix. In particular, this implies that $P(1, k_{\max}) \cdot C$ is a Cauchy-like matrix. Algorithm 1 below performs fast GEPP for a Cauchy-like matrix C . It is suggested by Gohberg, Kailath, and Olshevsky [10]. The recursions for computing A and B (without explicitly computing L and U) and the partial pivoting idea are from [15].

Algorithm 1 Fast GEPP for a Cauchy-like matrix.

```

 $L := 0; U := 0; P := I;$ 
for  $k = 1$  to  $n$  do
   $L_{k:n,k} := (\Omega_{k:n,k:n} - \lambda_k I)^{-1} \cdot A_{k:n,:} \cdot B_{:,k};$ 
   $k_{\max} := \operatorname{argmax}_{k \leq j \leq n} |L_{j,k}|;$ 
  if  $k_{\max} > k$  then
     $P := P \cdot P(k, k_{\max}); \Omega := P(k, k_{\max}) \cdot \Omega \cdot P(k, k_{\max});$ 
     $A = P(k, k_{\max}) \cdot A; L_{:,1:k} = P(k, k_{\max}) \cdot L_{:,1:k};$ 
     $U_{k,k} = L_{k,k}; U_{k,k+1:n} := A_{k,:} \cdot B_{:,k+1:n} \cdot (\omega_k I - \Lambda_{k+1:n,k+1:n})^{-1};$ 
     $L_{k,k} = 1; L_{k+1:n,k} := L_{k+1:n,k} / U_{k,k};$ 
     $A_{k+1:n,:} = A_{k+1:n,:} - L_{k+1:n,k} \cdot A_{k,:}; B_{:,k+1:n} = B_{:,k+1:n} - B_{:,k+1} \cdot U_{k,k+1:n} / U_{k,k}.$ 
  endfor

```

Remark 1: If the input data A , B , Ω and Λ are real, Algorithm 1 costs about $(4\alpha + 2.5)n^2$ flops; there is also potentially about $n^2/2$ swaps of memory locations. For a matrix transformed into a Cauchy-like matrix from a Toeplitz-plus-Hankel matrix (see §1 and §3), the displacement rank α is at most 4. In this case, Algorithm 1 costs about $18.5n^2$ flops.

Remark 2: If the input data are complex, Algorithm 1 costs about $(16\alpha + 12)n^2$ flops²; there is also potentially about n^2 swaps of memory locations. For a matrix transformed into a Cauchy-like matrix from a Toeplitz matrix (see §3), the displacement rank α is at most 2. In this case, Algorithm 1 costs about $44n^2$ flops.

²We count a complex addition or subtraction as 2 flops; a complex multiplication as 6 flops; a complex division as 10 flops; and the total cost of taking absolute value and performing comparison as 4 flops.

We observe that Algorithm 1 produces the same LU factorization as that of straightforward GEPP on C . Hence, one potential problem with Algorithm 1 is the *element growth* in the LU factorization. Let U be the upper triangular matrix in the LU factorization, and let $g_{PP} \equiv \|U\|_{\max}/\|C\|_{\max}$ be the *element growth factor*. It is well-known that $g_{PP} \leq 2^{n-1}$ for GEPP, and although very rare, this bound is attainable for certain dense matrices [13, pages 115-116]. It is not clear whether this bound is attainable for Cauchy-like matrices with low displacement rank. When large element growth does occur, the computed LU factorizations can be very inaccurate.

2.2 Complete Pivoting

Complete pivoting may in general further reduce the element growth in the LU factorization. To perform complete pivoting on C , one finds its largest magnitude entry $(k_{\max}, j_{\max})$ in the entire matrix, permute it to the $(1,1)$ entry to get $P(1, k_{\max}) \cdot C \cdot P(1, j_{\max})$, and then applies the elimination step to this permuted matrix. Let C be a Cauchy-like matrix satisfying equation (2.5). Then for every $1 \leq k, j \leq n$,

$$\begin{aligned} & (P(1, k) \cdot \Omega \cdot P(1, k)) \cdot (P(1, k) \cdot C \cdot P(1, j)) \\ - & (P(1, k) \cdot C \cdot P(1, j)) \cdot (P(1, j) \cdot \Lambda \cdot P(1, j)) = (P(1, k) \cdot A) \cdot (B \cdot P(1, j)) . \end{aligned}$$

In particular, this equation implies that $P_{1, k_{\max}} \cdot M \cdot P_{1, j_{\max}}$ is still a Cauchy-like matrix.

However, finding the largest magnitude entry $(k_{\max}, j_{\max})$ of C costs $O(n^2)$ flops in general. If this is done on every step of Gaussian elimination, then the total cost will be $O(n^3)$, which is too expensive.

On the other hand, it is not absolutely necessary to use *the* largest magnitude entry as pivot in order to reduce element growth. Any entry sufficiently large in magnitude should do.

Define

$$\xi_{\max} = \max_{1 \leq i, j \leq n} |\omega_i - \lambda_j| , \quad \xi_{\min} = \min_{1 \leq i, j \leq n} |\omega_i - \lambda_j| , \quad \text{and } \rho = \frac{\xi_{\max}}{\xi_{\min}} . \quad (2.7)$$

The following lemma tells us where to look for such an entry.

Lemma 2.1 *Let C be a Cauchy-like matrix satisfying equation (2.5), and let the $k_{\max}^{\text{th}}$ column be the largest 2-norm column of $A \cdot B$. Then*

$$\|C\|_{\max} \leq \sqrt{n} \cdot \rho \cdot \|C_{:, k_{\max}}\|_{\infty} \quad \text{and} \quad \|C\|_F \leq n \cdot \rho \cdot \|C_{:, k_{\max}}\|_{\infty} .$$

Proof. Let $|G_{i_G, k_{\max}}| = \|G_{:, k_{\max}}\|_{\infty}$, where $G = A \cdot B$. Then for any $1 \leq s, k \leq n$,

$$|C_{s, k}| = \frac{|G_{s, k}|}{|\omega_s - \lambda_k|} \leq \frac{\|G_{:, k}\|_2}{|\omega_s - \lambda_k|} \leq \frac{\|G_{:, k_{\max}}\|_2}{|\omega_s - \lambda_k|}$$

$$\begin{aligned}
&\leq \frac{\sqrt{n} \cdot |G_{i_G, k_{\max}}|}{|\omega_s - \lambda_k|} = \frac{\sqrt{n} \cdot |\omega_{i_G} - \lambda_{k_{\max}}|}{|\omega_s - \lambda_k|} \cdot |C_{i_G, k_{\max}}| \\
&\leq \frac{\sqrt{n} \cdot \xi_{\max}}{\xi_{\min}} \cdot \|C_{:, k_{\max}}\|_{\infty}.
\end{aligned}$$

Hence the first assertion of the lemma follows immediately.

For the second assertion, we have

$$\begin{aligned}
\|C\|_F^2 &= \sum_{s,k} |C_{s,k}|^2 = \sum_{s,k} \frac{|G_{s,k}|^2}{|\omega_s - \lambda_k|^2} \leq \frac{\sum_{s,k} |G_{s,k}|^2}{\xi_{\min}^2} \\
&\leq \frac{n \cdot \sum_s |G_{s, k_{\max}}|^2}{\xi_{\min}^2} \leq \frac{n \cdot \sum_s |C_{s, k_{\max}}|^2 \cdot \xi_{\max}^2}{\xi_{\min}^2} \\
&\leq \frac{n^2 \cdot \|C_{:, k_{\max}}\|_{\infty}^2 \cdot \xi_{\max}^2}{\xi_{\min}^2}.
\end{aligned}$$

To finish the proof, we take square roots on both sides. $\blacksquare$

To find the column $k_{\max}$ in Lemma 2.1, we QR-factorize A to get $A = \mathcal{A} \cdot R$, where $\mathcal{A} \in \mathbf{R}^{n \times \alpha}$ is column orthogonal, and R is upper-triangular. We then compute $B = R \cdot B$. It follows that

$$A \cdot B = \mathcal{A} \cdot B. \quad (2.8)$$

Since $\mathcal{A}$ is column orthogonal, the j^{th} columns of $A \cdot B$ and B have the same 2-norm, for $1 \leq j \leq n$. Hence we obtain $k_{\max}$ by looking for the largest 2-norm column of B . This allows us to perform a variation of GECP in Algorithm 2 below, which assumes that the matrix A is column orthogonal on input.

Algorithm 2 Fast GECP for a Cauchy-like matrix.

$L := 0; U := 0; P := I; Q := I;$

for $k = 1$ to n **do**

$j_{\max} := \operatorname{argmax}_{k \leq j \leq n} \|B_{:,j}\|_2;$

if $j_{\max} > k$ **then**

$Q := P(k, j_{\max}) \cdot Q; \Lambda := P(k, j_{\max}) \cdot \Lambda \cdot P(k, j_{\max});$

$B := B \cdot P(k, j_{\max}); U_{1:k,:} = U_{1:k,:} \cdot P(k, j_{\max});$

$L_{k:n,k} := (\Omega_{k:n,k:n} - \lambda_k I)^{-1} \cdot A_{k:n,:} \cdot B_{:,k};$

$k_{\max} := \operatorname{argmax}_{k \leq j \leq n} |L_{j,k}|;$

if $k_{\max} > k$ **then**

$P := P \cdot P(k, k_{\max}); \Omega := P(k, k_{\max}) \cdot \Omega \cdot P(k, k_{\max});$

$A := P(k, k_{\max}) \cdot A; L_{:,1:k} := P_{k,k_{\max}} \cdot L_{:,1:k};$

$U_{k,k} := L_{k,k}; U_{k,k+1:n} := A_{k,:} \cdot B_{:,k+1:n} \cdot (\omega_k I - \Lambda_{k+1:n,k+1:n})^{-1};$

$L_{k,k} := 1; L_{k+1:n,k} := L_{k+1:n,k} / U_{k,k};$

$A_{k+1:n,:} := A_{k+1:n,:} - L_{k+1:n,k} \cdot A_{k,:}; B_{:,k+1:n} := B_{:,k+1:n} - B_{:,k+1} \cdot U_{k,k+1:n} / U_{k,k};$

$A_{k+1:n,:} := \mathcal{A} \cdot R$ (QR factorization of $A_{k+1:n,:}$); $B := R \cdot B_{:,k+1:n}$;
 $A_{k+1:n,:} := \mathcal{A}$, $B_{:,k+1:n} := B$.
endfor

For the rest of §2.2, we derive an upper bound on the element growth factor for Algorithm 2, using techniques similar to those used by Wilkinson [25] to bound the growth factor for the straightforward GECP. In §2.3 we will discuss Algorithm 2 in more detail; and in §4.5 we will show that Algorithm 2 is numerically stable provided that the U matrix is not large in norm.

Let

$$\mathcal{W}(k) = \left(k \prod_{s=2}^k s^{1/(s-1)} \right)^{1/2} = O\left(k^{\frac{1}{2} + \frac{1}{4} \ln k}\right),$$

which is Wilkinson's upper bound on the growth factor for GECP on a $k \times k$ matrix. Although $\mathcal{W}(k)$ is not a polynomial in k , it does not grow very fast either [25].

We will need the following well-known result.

Lemma 2.2 (Householder [18, page 15]) *For any $C \in \mathbf{R}^{n \times n}$, we have*

$$|\det C| \leq \left(\frac{\|C\|_F}{\sqrt{n}} \right)^n.$$

Theorem 2 *Let C be a Cauchy-like matrix satisfying (2.5), and let $C = P \cdot L \cdot U \cdot Q$ be the LU factorization generated by Algorithm 2 in exact arithmetic. Then the element growth factor $g_{CP} \equiv \|U\|_{\max}/\|C\|_{\max}$ satisfies*

$$g_{CP} \leq \sqrt{n} \cdot \rho^{2 + \sum_{k=1}^{n-1} 1/k} \cdot \mathcal{W}(n). \quad (2.9)$$

Proof. Without loss of generality we assume that pivoting has been done before hand, so that Algorithm 2 does not perform any pivoting.

For $1 \leq k \leq n$, let $C^{(k)} \in \mathbf{R}^{(n-k+1) \times (n-k+1)}$ be the Cauchy-like matrix to be factored at the k^{th} step in Algorithm 2, with γ_k being the pivot (the $(1,1)$ entry of $C^{(k)}$). We note that $C^{(1)} = C$ in this notation.

Since Algorithm 2 performs partial pivoting, we have $|\gamma_k| = \|C_{:,1}^{(k)}\|_{\infty}$; since the first column of the generator for $C^{(k)}$ has the largest column 2-norm, we have $\|C^{(k)}\|_F \leq (n-k+1) \cdot \rho \cdot |\gamma_k|$ according to Lemma 2.1. It follows from Lemma 2.2 that

$$\begin{aligned}
 |\det(C^{(k)})| &\leq \left(\frac{\|C^{(k)}\|_F}{\sqrt{n-k+1}} \right)^{n-k+1} \leq \left(\frac{(n-k+1) \cdot \rho \cdot |\gamma_k|}{\sqrt{n-k+1}} \right)^{n-k+1} \\
 &= \left(\sqrt{n-k+1} \cdot \rho \cdot |\gamma_k| \right)^{n-k+1}.
 \end{aligned}$$

On the other hand,

$$|\det(C^{(k)})| = |\gamma_k| \cdots |\gamma_n|.$$

Comparing these two relations, we have

$$|\gamma_k| \cdots |\gamma_n| \leq \left(\rho \cdot \sqrt{n-k+1} \cdot |\gamma_k| \right)^{n-k+1}, \quad 1 \leq k \leq n. \quad (2.10)$$

Since

$$\sum_{k=1}^s \frac{1}{(n-k)(n-k+1)} + \frac{1}{n} = \frac{1}{n-s},$$

taking the product of the $(n-k)(n-k+1)^{st}$ root of (2.10) with $k = 1, 2, \dots, n-1$ and the n^{th} root of (2.10) with $k = 1$, we have

$$\begin{aligned} \prod_{s=1}^{n-1} |\gamma_s|^{1/(n-s)} \cdot \gamma_n &\leq \left(\prod_{k=1}^{n-1} \left(\rho \sqrt{n-k+1} \cdot |\gamma_k| \right)^{1/(n-k)} \right) \cdot \left(\rho \sqrt{n} \cdot \gamma_1 \right) \\ &= \rho^{1+\sum_{k=1}^{n-1} 1/(n-k)} \cdot \left(n \cdot \prod_{k=1}^{n-1} (n-k+1)^{1/(n-k)} \right)^{1/2} \cdot \left(\prod_{k=1}^{n-1} |\gamma_k|^{1/(n-k)} \right) \cdot |\gamma_1|, \end{aligned}$$

which simplifies to

$$|\gamma_n| \leq |\gamma_1| \cdot \rho^{1+\sum_{k=1}^{n-1} 1/k} \cdot \left(n \cdot \prod_{k=2}^n k^{1/(k-1)} \right)^{1/2} = |\gamma_1| \cdot \rho^{1+\sum_{k=1}^{n-1} 1/k} \cdot \mathcal{W}(n).$$

Repeating the same argument allows us to conclude that

$$|\gamma_s| \leq |\gamma_1| \cdot \rho^{1+\sum_{k=1}^{n-1} 1/k} \cdot \mathcal{W}(n), \quad 1 \leq s \leq n.$$

It now follows from Lemma 2.1 that

$$\|C^{(s)}\|_{\max} \leq \sqrt{n} \cdot \rho \cdot |\gamma_s| \leq |\gamma_1| \cdot \sqrt{n} \cdot \rho^{2+\sum_{k=1}^{n-1} 1/k} \cdot \mathcal{W}(n).$$

To complete the proof, we observe that the s^{th} row of the upper triangular matrix U is the first row of $C^{(s)}$. Hence

$$\|U\|_{\max} \leq |\gamma_1| \cdot \sqrt{n} \cdot \rho^{2+\sum_{k=1}^{n-1} 1/k} \cdot \mathcal{W}(n).$$

The assertion of the theorem follows immediately from the fact that $|\gamma_1| \leq \|C\|_{\max}$.

Remark 3: The determinant argument in the proof of Theorem 2 ignores the fact that C is a Cauchy-like matrix; hence the upper bound provided by (2.9) could be much larger than necessary, especially for Cauchy-like matrices with low displacement rank.

Remark 4: Since $\sum_{k=1}^{n-1} 1/k = \ln n + O(1)$, the bound (2.9) simplifies to

$$g_{CP} \leq n^{\ln \rho + \frac{1}{4} \ln n + O(1)}.$$

Assume that $\rho = O(n^\beta)$ for a constant β , then

$$g_{CP} \leq n^{\frac{4\beta+1}{4} \ln n + O(1)}.$$

If C is transformed into a Cauchy-like matrix from a Toeplitz matrix (or a Toeplitz-plus-Hankel matrix) via any of the transforms discussed in §3, then $1 \leq \beta \leq 3$. Although this upper bound is much larger than $\mathcal{W}(n)$, it is still much smaller than 2^{n-1} .

2.3 Further Considerations on Algorithm 2

In addition to the potential element growth in the LU factorization, Sweet and Brent [23] show that the generator (A, B) updated as in Algorithm 1 could also grow so that

$$\| |A_{k:n,:}| \cdot |B_{:,k:n}| \|_2 \gg \|A_{k:n,:} \cdot B_{:,k:n}\|_2$$

for some k . And if this happens, the backward and forward error could become large.

However, such element growth in the generator can easily be avoided. Since $A_{k:n,:}$ is kept column orthogonal for all k in Algorithm 2, it follows that

$$\begin{aligned} \| |A_{k:n,:}| \cdot |B_{:,k:n}| \|_2 &\leq \| |A_{k:n,:}| \|_2 \cdot \| |B_{:,k:n}| \|_2 \leq \|A_{k:n,:}\|_F \cdot \|B_{:,k:n}\|_F \\ &\leq \sqrt{\alpha} \cdot \|B_{:,k:n}\|_F \leq \alpha \cdot \|B_{:,k:n}\|_2 = \alpha \cdot \|A_{k:n,:} \cdot B_{:,k:n}\|_2. \end{aligned}$$

Hence keeping $A_{k:n,:}$ column orthogonal for all k also has the additional advantage of avoiding potential element growth within the generator (A, B) . In fact such growth can be avoided as long as $A_{k:n,:}$ is well-conditioned.

From a practical point of view, it does not seem necessary to column orthogonalize $A_{k:n,:}$ at every step just to keep it well-conditioned; nor does it seem necessary to perform pivoting on the columns at every step to reduce element growth. As a practical modification to Algorithm 2, the following algorithm performs these operations only once in every K steps, where K is a user provided positive integer. It assumes that the matrix A is initially column orthogonal.

Algorithm 3 Practical Modification to Algorithm 2.

$L := 0; U := 0; P := I; Q := I;$

for $k = 1$ to n **do**

if $(\text{mod}(k, K) = 1)$ **then**

$j_{\max} := \arg\max_{k \leq j \leq n} \|B_{:,j}\|_2;$

if $j_{\max} > k$ **then**

$Q := P(k, j_{\max}) \cdot Q; \Lambda := P(k, j_{\max}) \cdot \Lambda \cdot P(k, j_{\max});$

$B := B \cdot P(k, j_{\max}); U_{1:k,:} = U_{1:k,:} \cdot P(k, j_{\max});$

$L_{k:n,k} := (\Omega_{k:n,k:n} - \lambda_k I)^{-1} \cdot A_{k:n,:} \cdot B_{:,k};$

```

 $k_{\max} := \operatorname{argmax}_{k \leq j \leq n} |L_{j,k}|;$ 
if  $k_{\max} > k$  then
     $P := P \cdot P(k, k_{\max}); \Omega := P(k, k_{\max}) \cdot \Omega \cdot P(k, k_{\max});$ 
     $A := P(k, k_{\max}) \cdot A; L_{:,1:k} := P_{k,k_{\max}} \cdot L_{:,1:k};$ 
     $U_{k,k} := L_{k,k}; U_{k,k+1:n} := A_{k,:} \cdot B_{:,k+1:n} \cdot (\omega_k I - \Lambda_{k+1:n,k+1:n})^{-1};$ 
     $L_{k,k} := 1; L_{k+1:n,k} := L_{k+1:n,k} / U_{k,k};$ 
     $A_{k+1:n,:} := A_{k+1:n,:} - L_{k+1:n,k} \cdot A_{k,:}; B_{:,k+1:n} := B_{:,k+1:n} - B_{:,k+1} \cdot U_{k,k+1:n} / U_{k,k};$ 
if  $(\operatorname{mod}(k, K) = 0)$  then
     $A_{k+1:n,:} := \mathcal{A} \cdot R$  (QR factorization of  $A_{k+1:n,:}$ );  $\mathcal{B} := R \cdot B_{:,k+1:n};$ 
     $A_{k+1:n,:} := \mathcal{A}, B_{:,k+1:n} := \mathcal{B}.$ 
endfor

```

Remark 5: The cost for recomputing $A_{k+1:n,:}$ and $B_{:,k+1:n}$ through QR factorizations is about $5/2\alpha^2 n^2$ flops in real arithmetic and $10\alpha^2 n^2$ flops in complex arithmetic. However, if α is large and if QR factorization is performed at every step, these costs can be brought down to $O(\alpha n^2)$ by using QR updating techniques (see [13, §12]). Our main interest in this paper is to use Algorithm 3 to factorize the Cauchy-like matrix that is transformed from a Toeplitz-plus-Hankel matrix (cf. §1 and §3). For such matrices α is at most 4. In our implementation, we recompute the QR factorization every $K = 10$ steps.

3 Factorizing Toeplitz-plus-Hankel-like Matrices

3.1 Factorizing Toeplitz-plus-Hankel-like Matrices

Define

$$Y_{\delta_1, \delta_2} = \begin{pmatrix} \delta_1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & 1 \\ 0 & \cdots & 0 & 1 & \delta_2 \end{pmatrix}, \quad (3.11)$$

and $\Omega = Y_{1,1}$ and $\Lambda = Y_{1,-1}$. It is easy to verify that every Toeplitz-plus-Hankel matrix satisfies the displacement equation (1.1) with G having non-zero entries only in its first and last rows and columns, thus a matrix of rank at most 4. Hence the displacement rank of a Toeplitz-plus-Hankel matrix is at most 4 (cf. [10, 16]). In particular, these results are true for every Toeplitz or Hankel matrix.

Lemma 3.1 *Let $M \in \mathbf{R}^{n \times n}$ be a matrix satisfying the displacement equation*

$$Y_{1,1} \cdot M - M \cdot Y_{1,-1} = A \cdot B, \quad (3.12)$$

where $A \in \mathbf{R}^{n \times \alpha}$ and $B \in \mathbf{R}^{\alpha \times n}$. Then $\mathcal{Q}_{1,1}^T \cdot M \cdot \mathcal{Q}_{1,-1}$ is a Cauchy-like matrix:

$$\mathcal{D}_{1,1} \cdot \left(\mathcal{Q}_{1,1}^T \cdot M \cdot \mathcal{Q}_{1,-1} \right) - \left(\mathcal{Q}_{1,1}^T \cdot M \cdot \mathcal{Q}_{1,-1} \right) \cdot \mathcal{D}_{1,1} = \left(\mathcal{Q}_{1,1}^T \cdot A \right) \cdot \left(B \cdot \mathcal{Q}_{1,-1} \right), \quad (3.13)$$

where

$$\mathcal{Q}_{1,1} = \sqrt{\frac{2}{n}} \cdot \left(q_j \cos \frac{(2k-1)(j-1)\pi}{2n} \right)_{1 \leq k,j \leq n}$$

and

$$\mathcal{Q}_{1,-1} = \left(\sqrt{\frac{2}{n}} \cdot \left(\cos \frac{(2k-1)(2j-1)\pi}{4n} \right) \right)_{1 \leq k,j \leq n}$$

are orthogonal matrices with $q_1 = \frac{1}{2}$ and $q_j = 1$ for $2 \leq j \leq n$; and

$$\begin{aligned} \mathcal{D}_{1,1} &= 2 \cdot \text{diag} \left(1, \cos \frac{\pi}{n}, \dots, \cos \frac{(n-1)\pi}{n} \right), \\ \mathcal{D}_{1,-1} &= 2 \cdot \text{diag} \left(\cos \frac{\pi}{2n}, \cos \frac{3\pi}{2n}, \dots, \cos \frac{(2n-1)\pi}{2n} \right). \end{aligned}$$

Proof. It can be checked that

$$Y_{1,1} = \mathcal{Q}_{1,1} \cdot \mathcal{D}_{1,1} \cdot \mathcal{Q}_{1,1}^T, \quad Y_{1,-1} = \mathcal{Q}_{1,-1} \cdot \mathcal{D}_{1,-1} \cdot \mathcal{Q}_{1,-1}^T.$$

The lemma follows immediately by substituting the above equation into equation (3.12), and multiplying by $\mathcal{Q}_{1,1}^T$ from the left and by $\mathcal{Q}_{1,-1}$ from the right. ■

We call a matrix M *Toeplitz-plus-Hankel-like* if it satisfies the displacement equation (3.12) with $\alpha \ll n$ (cf. [10]). To solve a Toeplitz-plus-Hankel-like linear system of equations

$$M \cdot x = z,$$

one can transform M into a Cauchy-like matrix using Lemma 3.1; factorize this matrix by using any of the methods discussed in §2 to obtain a factorization of the form

$$M = \mathcal{Q}_{1,1}^T \cdot P \cdot L \cdot U \cdot Q \cdot \mathcal{Q}_{1,-1}^T; \quad (3.14)$$

and compute the solution to the linear system using this factorization. The idea of transforming a Toeplitz matrix into a Cauchy-like matrix was first proposed by Heinig [15]; and the idea of transforming a Toeplitz-plus-Hankel matrix into a Cauchy-like matrix was first proposed by Gohberg, Kailath, and Olshevsky [10].

We summarize the above into Algorithm 4 that follows, assuming that M satisfies equation (3.12) with A column orthogonal.

Algorithm 4 Solving $M \cdot x = z$.

1. Set $\Omega := \mathcal{D}_{1,1}$; $\Lambda := \mathcal{D}_{1,-1}$; and compute $A := \mathcal{Q}_{1,1}^T \cdot A$; $B := B \cdot \mathcal{Q}_{1,-1}$.

2. Compute the factorization (3.14) by applying one of Algorithms 1, 2, and 3 with Ω , Λ , A and B .
3. Compute $x = Q_{1,-1} \cdot Q^T \cdot U^{-1} \cdot L^{-1} \cdot P^T \cdot Q_{1,1} \cdot z$.

Both $Q_{1,1}$ and $Q_{1,-1}$ are fast trigonometric transform matrices, hence the cost of Step 1 is about $O(\alpha n \log_2 n)$ flops via 2α such transforms; similarly the cost of Step 3 is about $2n^2$ flops via 2 fast trigonometric transforms, two permutations, and forward and backward-substitution. The bulk of the cost is in Step 2, factorizing the Cauchy-like matrix in (3.13).

For a real Toeplitz-plus-Hankel matrix, the displacement rank α is at most 4 in equation (3.13). When Algorithm 3 is used in Step 2, the cost for Step 2 is about $18.5n^2 + O(n^2/K)$ flops for a user specified integer K (see Remark 5). Hence Algorithm 4 takes about $20.5n^2 + O(n^2/K)$ flops to solve a Toeplitz-plus-Hankel system of equations. This is also true for a Toeplitz or a Hankel system of equations.

3.2 Comparison with Previous Methods

The Toeplitz-plus-Hankel matrix satisfies other displacement equations, too. It is known that [16, 10] for $\Omega = Y_{0,0}$ and $\Lambda = Y_{1,1}$ every Toeplitz-plus-Hankel matrix satisfies the displacement equation (1.1) with G having non-zero entries only in its first and last rows and columns. It is known that matrix $Y_{0,0}$ can be diagonalized using fast trigonometric transform matrices (see for example [2]):

$$Y_{0,0} = Q_{0,0} \cdot \mathcal{D}_{0,0} \cdot Q_{0,0}^T,$$

where $Q_{0,0} = \frac{2}{n+1} \cdot \left(\sin \frac{kj\pi}{n+1} \right)_{1 \leq k,j \leq n}$, and $\mathcal{D}_{0,0} = 2 \cdot \text{diag} \left(\cos \frac{\pi}{n+1}, \dots, \cos \frac{n\pi}{n+1} \right)$. Golberg, Kailath, and Olshevsky [10] suggest that to solve a Toeplitz-plus-Hankel system of linear equations, one transforms the coefficient matrix to a Cauchy-like matrix C that satisfies

$$\mathcal{D}_{0,0} \cdot C - C \cdot \mathcal{D}_{1,-1} = A \cdot B, \quad (3.15)$$

with $\text{rank}(A), \text{rank}(B) \leq 4$; and one then applies Algorithm 1 to C . The resulting algorithm is named Algorithm TpH.

However, Algorithm TpH has some disadvantages over Algorithm 4. It can be shown that the parameter ρ defined in (2.7) is $O(n^3)$ for equation (3.15) and $O(n^2)$ for (3.13). Our upper bound on g_{CP} in §2.2 and error analysis in §4 suggest that the smaller ρ is, the smaller the potential element growth and backward error. Hence Algorithm TpH could be potentially less accurate than Algorithm 4. Another disadvantage for Algorithm TpH is that in order for the fast trigonometric transforms with $Q_{0,0}$ and $Q_{1,1}$ to be very efficient, both n and $n+1$ must be products of small prime numbers; whereas for Algorithm 4, it is sufficient that n be a product of small prime numbers.

If one wants to solve a Toeplitz system of linear equations, then other displacement structures may be used. Define

$$Z_\delta = \begin{pmatrix} 0 & 0 & \cdots & 0 & \delta \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix},$$

and let $\Omega = Z_1$ and $\Lambda = Z_{-1}$. Kailath, Kung, Morf [19] show that every Toeplitz matrix satisfies the displacement equation (1.1) with G having non-zero entries only in its first row and last column, a matrix of rank at most 2. Hence the displacement rank of a Toeplitz matrix is at most 2 with respect to Z_1 and Z_{-1} . The following result can be found in [15].

Proposition 1 *Let $M \in \mathbf{R}^{n \times n}$ be a matrix satisfying the displacement equation*

$$Z_1 \cdot M - M \cdot Z_{-1} = A \cdot B, \quad (3.16)$$

where $A \in \mathbf{R}^{n \times \alpha}$ and $B \in \mathbf{R}^{\alpha \times n}$. Then $\mathcal{F} \cdot M \cdot D_0^{-1} \cdot \mathcal{F}^$ is a Cauchy-like matrix:*

$$\mathcal{D}_1 \cdot (\mathcal{F} \cdot M \cdot D_0^{-1} \cdot \mathcal{F}^*) - (\mathcal{F} \cdot M \cdot D_0^{-1} \cdot \mathcal{F}^*) \cdot \mathcal{D}_{-1} = (\mathcal{F} \cdot A) \cdot (B \cdot D_0^* \cdot \mathcal{F}^*), \quad (3.17)$$

where $\mathcal{F} = \sqrt{\frac{1}{n}} \cdot \left(e^{\frac{2\pi i}{n}(k-1)(j-1)} \right)_{1 \leq k, j \leq n}$ is the normalized Inverse Discrete Fourier Transform matrix,

$$\mathcal{D}_1 = \text{diag} \left(1, e^{\frac{2\pi i}{n}}, \dots, e^{\frac{2\pi i}{n}(n-1)} \right) \quad \mathcal{D}_{-1} = \text{diag} \left(e^{\frac{\pi i}{n}}, e^{\frac{3\pi i}{n}}, \dots, e^{\frac{(2n-1)\pi i}{n}} \right).$$

Heinig [15] suggests that for a Toeplitz matrix T , one can convert it into the Cauchy-like matrix in equation (3.17); and Gohberg, Kailath, and Olshevsky [10] suggest that one can rapidly factorize this Cauchy-like matrix using Algorithm 1. The resulting algorithm is called Algorithm GKO in [10]. Since the cost of a fast algorithm for factorizing a Cauchy-like matrix depends linearly on the displacement rank (see Remarks 1 and 2), this method is more efficient than Algorithm 4 if T is given to be a complex matrix.

However, the situation is different if T is real (as often happens in practice). The total cost of complex forward and backward-substitution is about $8n^2$ flops; and the total cost of factorizing the Cauchy-like matrix in (3.17) is about $44n^2$ flops for Algorithm 1 (see Remark 2). Using the above procedure, a Toeplitz system is thus solved in about $52n^2$ flops. This is about 2.5 times the cost of Algorithm 4. Furthermore, operating in complex arithmetic requires twice as much storage, a potentially big price to pay for large matrices.

On the other hand, Algorithm GKO does have an advantage over Algorithm 4: it can be shown that the parameter ρ defined in (2.7) is $O(n)$ for equation (3.17), thus Algorithm GKO could be more accurate. We will address this issue in §5.

4 Error Analysis

In this section, we do a backward error analysis for Algorithms 1 through 3 by establishing an ∞ -norm upper bound on the matrix H in the equation

$$\hat{L} \cdot \hat{U} = C + H, \quad (4.18)$$

where C is the Cauchy-like matrix to be factored; $\hat{L} \cdot \hat{U}$ is the computed LU factorization; and we assume that no pivoting is done. In the following, we first establish some notation and then analyze error propagation by using induction. At the end of this section we will briefly discuss error propagation for Algorithm 4.

4.1 Notation

At the k^{th} step of elimination in finite arithmetic, let $\hat{C}^{(k)} = \begin{pmatrix} \gamma_k & (u^{(k)})^T \\ r^{(k)} & \tilde{C}_k \end{pmatrix}$ be the Cauchy-like matrix satisfying the displacement equation

$$\Omega_k \cdot \hat{C}^{(k)} - \hat{C}^{(k)} \cdot \Lambda_k = \hat{A}^{(k)} \cdot \hat{B}^{(k)} \quad (4.19)$$

with $\Omega_k = \text{diag}(\omega_k, \Omega_{k+1})$ and $\Lambda_k = \text{diag}(\lambda_k, \Lambda_{k+1}) \in \mathbf{R}^{(n-k+1) \times (n-k+1)}$ diagonal;

$$\hat{A}^{(k)} = \begin{pmatrix} (\hat{a}_k^{(k)})^T \\ \hat{A}_{k+1} \end{pmatrix} = \begin{pmatrix} (\hat{a}_k^{(k)})^T \\ \vdots \\ (\hat{a}_n^{(k)})^T \end{pmatrix}, \quad \hat{B}^{(k)} = \begin{pmatrix} \hat{b}_k^{(k)} & \tilde{B}_{k+1} \end{pmatrix} = \begin{pmatrix} \hat{b}_k^{(k)} & \dots & \hat{b}_n^{(k)} \end{pmatrix}.$$

For $k = 1$, we drop the superscripts so that $\hat{C}^{(1)} = C$, $\hat{A}^{(1)} = A$, $\hat{B}^{(1)} = B$, etc., and equation (4.19) reduces to (2.5).

To perform elimination, we write

$$\hat{C}^{(k)} = \begin{pmatrix} 1 & 0 \\ l^{(k)} & I \end{pmatrix} \cdot \begin{pmatrix} \gamma_k & (u^{(k)})^T \\ 0 & C^{(k+1)} \end{pmatrix},$$

where $l^{(k)} = r^{(k)}/\gamma_k$ and $C^{(k+1)} = \tilde{C}_{k+1} - l^{(k)} \cdot (u^{(k)})^T$ satisfies the displacement equation

$$\Omega_{k+1} \cdot C^{(k+1)} - C^{(k+1)} \cdot \Lambda_{k+1} = A^{(k+1)} \cdot B^{(k+1)}$$

with $A^{(k+1)} = \tilde{A}_{k+1} - l^{(k)} \cdot (\hat{a}_k^{(k)})^T$ and $B^{(k+1)} = \tilde{B}_{k+1} - \hat{b}_k^{(k)} \cdot (u^{(k)})^T / \gamma_k$.

Let the computed γ_k , $r^{(k)}$, and $u^{(k)}$ be $\hat{\gamma}_k$, $\hat{r}^{(k)}$, and $\hat{u}^{(k)}$; and let $\hat{l}^{(k)} = \text{fl}(\hat{r}^{(k)}/\hat{\gamma}_k)$. For $k = 1$, we write $r = r^{(1)}$, $u = u^{(1)}$, $l = l^{(1)}$; and $\hat{r} = \hat{r}^{(1)}$, $\hat{u} = \hat{u}^{(1)}$, $\hat{l} = \hat{l}^{(1)}$.

Let $\bar{G}^{(k+1)} = \bar{A}^{(k+1)} \cdot \bar{B}^{(k+1)}$ with $\bar{A}^{(k+1)} = \tilde{A}_{k+1} - \hat{l}^{(k)} \cdot (\hat{a}_k^{(k)})^T$ and $\bar{B}^{(k+1)} = \tilde{B}_{k+1} - \hat{b}_k^{(k)} \cdot (\hat{u}^{(k)})^T / \hat{\gamma}_k$. The generator at the $(k+1)^{\text{st}}$ step is $\hat{G}^{(k+1)} = \hat{A}^{(k+1)} \cdot \hat{B}^{(k+1)}$. For Algorithm 1, $\hat{A}^{(k+1)} = \text{fl}(\bar{A}^{(k+1)})$ and $\hat{B}^{(k+1)} = \text{fl}(\bar{B}^{(k+1)})$; and for Algorithm 3, $\hat{A}^{(k+1)}$ is the computed Q factor in the QR factorization of $\text{fl}(\bar{A}^{(k+1)})$ and $\hat{B}^{(k+1)}$ is

the product of the R factor and $\text{fl}(\bar{B}^{(k+1)})$. We further define $\bar{C}^{(k+1)}$ to be the matrix satisfying the displacement equation

$$\Omega_{k+1} \cdot \bar{C}^{(k+1)} - \bar{C}^{(k+1)} \cdot \Lambda_{k+1} = \bar{G}^{(k+1)}. \quad (4.20)$$

Define

$$\tau = \max_{2 \leq k \leq n} \|\hat{G}^{(k)} - \bar{G}^{(k)}\|_\infty, \quad \mu = \max_{2 \leq k \leq n} \|\hat{C}^{(k)}\|_{\max}; \quad (4.21)$$

and

$$\nu = \max_{1 \leq k \leq i \leq n} \frac{|\hat{a}_i^{(k)}|^T \cdot |\hat{b}_k^{(k)}|}{|\hat{\gamma}_k|}, \quad \psi = \frac{\max_{1 \leq k \leq n} \|\hat{A}^{(k)} \cdot \hat{B}^{(k)}\|_{\max}}{\max_{1 \leq k \leq n} \|\hat{A}^{(k)} \cdot \hat{B}^{(k)}\|_{\max}}. \quad (4.22)$$

τ is a measure of the accuracy in computing the generators; μ is a measure of element growth in the LU factorization, since it is easy to show that

$$\mu \leq \| |\hat{L}| \cdot |\hat{U}| \|_{\max} + O(\epsilon); \quad (4.23)$$

$\psi \geq 1$ is of order 1 in general, but it could happen that $\psi \gg 1$ if both $\hat{A}^{(k)}$ and $\hat{B}^{(k)}$ are ill-conditioned for some k . We will further discuss these 4 parameters in §4.4 and §4.5.

Lemma 4.1 *For any $1 \leq k \leq i, j \leq n$,*

$$|\hat{a}_i^{(k)}|^T \cdot |\hat{b}_j^{(k)}| \leq \xi_{\max} \cdot \psi \cdot \mu.$$

Proof. Let $|(\hat{a}_s^{(f)})^T \cdot \hat{b}_m^{(f)}| = \max_{1 \leq h \leq n} \|\hat{A}^{(h)} \cdot \hat{B}^{(h)}\|_{\max}$. Then

$$\begin{aligned} |\hat{a}_i^{(k)}|^T \cdot |\hat{b}_j^{(k)}| &\leq \psi \cdot |(a_s^{(f)})^T \cdot b_m^{(f)}| \leq \psi \cdot \|C^{(f)}\|_{\max} \cdot |\omega_s - \lambda_m| \\ &\leq \xi_{\max} \cdot \psi \cdot \mu. \end{aligned}$$

■

4.2 Error Propagation for One Step of Elimination

Let $\hat{L}^{(2)} \cdot \hat{U}^{(2)} = \hat{C}^{(2)} + H^{(2)}$ be the computed LU factorization of $\hat{C}^{(2)}$. Then the computed LU factorization of C satisfies

$$\hat{L} = \begin{pmatrix} 1 & 0 \\ \hat{l} & \hat{L}^{(2)} \end{pmatrix} \text{ and } \hat{U} = \begin{pmatrix} \hat{\gamma}_1 & \hat{u}^T \\ 0 & \hat{U}^{(2)} \end{pmatrix}.$$

It follows that

$$\begin{aligned} \hat{L} \cdot \hat{U} &= \begin{pmatrix} \hat{\gamma}_1 & \hat{u}^T \\ \hat{l}\hat{\gamma}_1 & \hat{L}^{(2)} \cdot \hat{U}^{(2)} + \hat{l} \cdot \hat{u}^T \end{pmatrix} \\ &= \begin{pmatrix} \gamma_1 & u^T \\ r & \tilde{C}_2 \end{pmatrix} + \begin{pmatrix} \hat{\gamma}_1 - \gamma_1 & (\hat{u} - u)^T \\ \hat{l}\hat{\gamma}_1 - r & H^{(2)} + (\hat{C}^{(2)} - \bar{C}^{(2)}) + (\bar{C}^{(2)} - \tilde{C}_2 + \hat{l} \cdot \hat{u}^T) \end{pmatrix}. \end{aligned}$$

Since $M = \begin{pmatrix} \gamma_1 & u^T \\ r & \tilde{C}_2 \end{pmatrix}$, equation (4.18) and the last equation imply

$$H = \begin{pmatrix} \hat{\gamma}_1 - \gamma_1 & (\hat{u} - u)^T \\ \hat{l}\hat{\gamma}_1 - r & H^{(2)} + (\hat{C}^{(2)} - \bar{C}^{(2)}) + (\bar{C}^{(2)} - \tilde{C}_2 + \hat{l} \cdot \hat{u}^T) \end{pmatrix}. \quad (4.24)$$

For the rest of §4.2, we bound $|\hat{\gamma}_1 - \gamma_1|$, $|\hat{u} - u|$, $|\hat{l}\hat{\gamma}_1 - r|$, $|\hat{C}^{(2)} - \bar{C}^{(2)}|$, and $|\bar{C}^{(2)} - \tilde{C}_2 + \hat{l} \cdot \hat{u}^T|$. We obtain an upper bound on H by induction in §4.3.

Displacement equation (4.19) for $k = 1$ implies that

$$\gamma_1 = (\omega_1 - \lambda_1)^{-1} \cdot a_1^T \cdot b_1, \quad r = (\Omega_1 - \lambda_1 I)^{-1} \cdot \tilde{A}_2 \cdot b_1, \quad u = (\omega_1 I - \Lambda_1)^{-1} \cdot \tilde{B}_2^T \cdot a_1. \quad (4.25)$$

For both Algorithm 1 and Algorithm 3, the errors in these quantities and l can be bounded as follows, using our model of arithmetic (1.3) and Lemma 4.1:

$$\begin{aligned} |\hat{\gamma}_1 - \gamma_1| &\leq \alpha\eta|\omega_1 - \lambda_1|^{-1} \cdot |a_1|^T \cdot |b_1| \leq \frac{\alpha\eta\psi\mu\xi_{\max}}{\xi_{\min}}, \\ |\hat{r} - r| &\leq \alpha\eta|\Omega_1 - \lambda_1 I|^{-1} \cdot |\tilde{A}_2| \cdot |b_1| \leq \frac{\alpha\eta\psi\mu\xi_{\max}}{\xi_{\min}} \cdot e, \\ |\hat{u} - u| &\leq \alpha\eta|\omega_1 I - \Lambda_1|^{-1} \cdot |\tilde{B}_2^T| \cdot |a_1| \leq \frac{\alpha\eta\psi\mu\xi_{\max}}{\xi_{\min}} \cdot e, \end{aligned} \quad (4.26)$$

and $|\hat{l} - \hat{r}/\hat{\gamma}_1| \leq \eta|\hat{l}|$, where³ η is a small multiple of ϵ , and $e = (1, \dots, 1)^T$. These relations imply that

$$\begin{aligned} |\hat{l}\hat{\gamma}_1 - r| &\leq |\hat{l}\hat{\gamma}_1 - \hat{r}| + |\hat{r} - r| \\ &\leq \eta|\hat{\gamma}_1| \cdot |\hat{l}| + \frac{\alpha\eta\psi\mu\xi_{\max}}{\xi_{\min}} \cdot e. \end{aligned} \quad (4.27)$$

Since

$$\Omega_2 \cdot \bar{C}^{(2)} - \bar{C}^{(2)} \cdot \Lambda_2 = \bar{G}^{(2)} \quad \text{and} \quad \Omega_2 \cdot \hat{C}^{(2)} - \hat{C}^{(2)} \cdot \Lambda_2 = \hat{G}^{(2)},$$

subtract these two equations we have

$$\Omega_2 \cdot (\hat{C}^{(2)} - \bar{C}^{(2)}) - (\hat{C}^{(2)} - \bar{C}^{(2)}) \cdot \Lambda_2 = \hat{G}^{(2)} - \bar{G}^{(2)}.$$

By definition (4.21), this implies

$$\|\hat{C}^{(2)} - \bar{C}^{(2)}\|_{\infty} \leq \frac{\tau}{\xi_{\min}}. \quad (4.28)$$

Finally, we bound $\bar{H}^{(2)} \equiv \bar{C}^{(2)} - \tilde{C}_2 + \hat{l} \cdot \hat{u}^T$. Write $\hat{l} = (\hat{l}_2, \dots, \hat{l}_n)^T$ and $\hat{u} = (\hat{u}_2, \dots, \hat{u}_n)^T$. According to equations (4.19) and (4.20), the $(i-1, j-1)$ entries of

³Throughout §4 we use the same η in several similar error bounds; hence η is in fact the maximum of all these different η 's.

matrices $\bar{C}^{(2)}$, $\bar{C}$, and $\hat{l} \cdot \hat{u}^T$ are $(a_i^T - \hat{l}_i \cdot a_1^T) \cdot (b_j - b_1 \cdot \hat{u}_j / \hat{\gamma}_1) / (\omega_i - \lambda_j)$, $a_i^T \cdot b_j / (\omega_i - \lambda_j)$, and $\hat{l}_i \cdot \hat{u}_j$ for $2 \leq i, j \leq n$. Thus

$$\begin{aligned} |\bar{H}_{i-1,j-1}^{(2)}| &= \left| \frac{(a_i^T - \hat{l}_i \cdot a_1^T) \cdot (b_j - b_1 \cdot \hat{u}_j / \hat{\gamma}_1) - a_i^T \cdot b_j}{\omega_i - \lambda_j} + \hat{l}_i \cdot \hat{u}_j \right| \\ &= \left| \frac{\hat{l}_i \hat{u}_j (\omega_i - \lambda_j + a_1^T \cdot b_1 / \hat{\gamma}_1) - \hat{l}_i a_1^T \cdot b_j - \hat{u}_j \cdot a_i^T b_1 / \hat{\gamma}_1}{\omega_i - \lambda_j} \right|. \end{aligned}$$

Equation (4.25) implies that

$$a_1^T \cdot b_1 = (\omega_1 - \lambda_1) \cdot \gamma_1, \quad a_1^T \cdot b_j = (\omega_1 - \lambda_j) \cdot u_j, \quad \text{and} \quad a_i^T \cdot b_1 = (\omega_i - \lambda_1) \cdot r_i.$$

Plugging these relations into above and simplifying, we have

$$\begin{aligned} |\bar{H}_{i-1,j-1}^{(2)}| &= \left| \frac{\hat{l}_i \hat{u}_j \cdot \frac{(\omega_1 - \lambda_1)(\gamma_1 - \hat{\gamma}_1)}{\hat{\gamma}_1} - \hat{l}_i (\omega_1 - \lambda_j)(u_j - \hat{u}_j) - \hat{u}_j \cdot \frac{(\omega_i - \lambda_1)(r_i - \hat{l}_i \hat{\gamma}_1)}{\hat{\gamma}_1}}{\omega_i - \lambda_j} \right| \\ &\leq \frac{|\hat{l}_i| \cdot |\hat{u}_j| \cdot \frac{|\omega_1 - \lambda_1| \cdot |\gamma_1 - \hat{\gamma}_1|}{|\hat{\gamma}_1|} + |\hat{l}_i| \cdot |u_j - \hat{u}_j| \cdot |\omega_1 - \lambda_j| + |\hat{u}_j| \cdot \frac{|\omega_i - \lambda_1| \cdot |r_i - \hat{l}_i \hat{\gamma}_1|}{|\hat{\gamma}_1|}}{\xi_{\min}}. \end{aligned}$$

By relation (4.26) and definition (4.22), we have

$$\frac{|\omega_1 - \lambda_1| \cdot |\gamma_1 - \hat{\gamma}_1|}{|\hat{\gamma}_1|} \leq \frac{\alpha \eta |a_1|^T \cdot |b_1|}{|\hat{\gamma}_1|} \leq \alpha \eta \nu;$$

further by using relation (4.26), we have

$$\frac{|\omega_i - \lambda_1| \cdot |r_i - \hat{l}_i \hat{\gamma}_1|}{|\hat{\gamma}_1|} \leq \eta |\omega_i - \lambda_1| \cdot |\hat{l}_i| + \frac{\alpha \eta |a_i|^T \cdot |b_1|}{|\hat{\gamma}_1|} \leq \eta \xi_{\max} \cdot |\hat{l}_i| + \alpha \eta \nu.$$

Plugging these relations into the last bound on $|\bar{H}_{i-1,j-1}^{(2)}|$, rewriting the result in matrix form, and simplifying, we have

$$\begin{aligned} |\bar{H}^{(2)}| &\leq \frac{\alpha \eta \nu \cdot |\hat{l}| \cdot |\hat{u}|^T + |\hat{l}| \cdot |u - \hat{u}|^T \cdot |\omega_1 I - \Lambda_2| + \eta \xi_{\max} |\hat{l}| \cdot |\hat{u}|^T + \alpha \eta \nu \cdot e \cdot |\hat{u}|^T}{\xi_{\min}} \\ &\leq \frac{(\alpha \nu + \xi_{\max}) \eta}{\xi_{\min}} \cdot |\hat{l}| \cdot |\hat{u}|^T + \frac{\alpha \eta}{\xi_{\min}} \cdot |\hat{l}| \cdot |a_1|^T \cdot |\tilde{B}_2| + \frac{\alpha \nu \eta}{\xi_{\min}} \cdot e \cdot |\hat{u}|^T, \end{aligned}$$

where we have used relations (4.26). According to Lemma 4.1, the last relation can be further simplified to

$$\begin{aligned} |\bar{H}^{(2)}| &\leq \frac{(\alpha \nu + \xi_{\max}) \eta}{\xi_{\min}} \cdot |\hat{l}| \cdot |\hat{u}|^T + \frac{\alpha \eta \psi \mu \xi_{\max}}{\xi_{\min}} \cdot |\hat{l}| \cdot e^T + \frac{\alpha \nu \eta}{\xi_{\min}} \cdot e \cdot |\hat{u}|^T, \\ &\leq \frac{\alpha \eta (\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot (|\hat{l}| + e) \cdot (|\hat{u}| + \mu \cdot e)^T. \end{aligned} \tag{4.29}$$

4.3 Error Analysis for Factorizing Cauchy-like Matrices

Let $X_n \in \mathbf{R}^{n \times n}$ and $Y_n \in \mathbf{R}^{n \times n}$ be lower and upper triangular matrices such that

$$X_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 1 & \cdots & 1 & 1 \end{pmatrix} \quad \text{and} \quad Y_n = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & 1 \end{pmatrix}.$$

We also define

$$\Delta^{(k)} = \sum_{s=k+1}^n \begin{pmatrix} 0 & 0 \\ 0 & |\bar{C}^{(s)} - \hat{C}^{(s)}| \end{pmatrix}.$$

The following theorem gives an upper bound on $\|H\|_\infty$ in (4.18).

Theorem 3 *The backward error H in the LU factorization of a Cauchy-like matrix C in (4.18) satisfies*

$$\|H\|_\infty \leq \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot \left\| (|\hat{L}| + X_n) \cdot (|\hat{U}| + \mu \cdot Y_n) \right\|_\infty + \frac{n \cdot \tau}{\xi_{\min}}, \quad (4.30)$$

where ν , ψ , μ , and τ are defined in equations (4.21) and (4.22).

Proof. We shall first show that

$$|H| \leq \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot (|\hat{L}| + X_n) \cdot (|\hat{U}| + \mu \cdot Y_n) + \frac{\Delta^{(1)}}{\xi_{\min}} \quad (4.31)$$

by using induction on n . We shall then prove the theorem by taking ∞ -norm on both sides of (4.31).

Relation (4.31) clearly holds for all Cauchy-like matrices of dimension $n = 1, 2$; and we assume it holds for $n - 1$ as well. In light of equation (4.24), we have

$$|H| \leq \begin{pmatrix} |\hat{\gamma}_1 - \gamma_1| & |\hat{u} - u|^T \\ |\hat{l}\hat{\gamma}_1 - r| & |H^{(2)}| + |\hat{C}^{(2)} - \bar{C}^{(2)}| + |\bar{C}^{(2)} - \tilde{C}_2 + \hat{l} \cdot \hat{u}^T| \end{pmatrix}.$$

Plugging relations (4.26) through (4.29) into above, we have

$$\begin{aligned} |H| &\leq \begin{pmatrix} 0 & 0 \\ 0 & |H^{(2)}| \end{pmatrix} + \begin{pmatrix} |\hat{\gamma}_1 - \gamma_1| & |\hat{u} - u|^T \\ |\hat{l}\hat{\gamma}_1 - r| & 0 \end{pmatrix} \\ &\quad + \begin{pmatrix} 0 & 0 \\ 0 & |\hat{C}^{(2)} - \bar{C}^{(2)}| \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & |\bar{C}^{(2)} - \tilde{C}_2 + \hat{l} \cdot \hat{u}^T| \end{pmatrix} \\ &\leq \begin{pmatrix} 0 & 0 \\ 0 & |H^{(2)}| \end{pmatrix} + \frac{\alpha\eta\psi\xi_{\max}}{\xi_{\min}} \cdot \begin{pmatrix} \mu & \mu \cdot e^T \\ \mu \cdot e & 0 \end{pmatrix} + \eta \cdot \begin{pmatrix} 0 & 0 \\ |\hat{l}| \cdot |\hat{\gamma}_1| & 0 \end{pmatrix} \end{aligned}$$

$$\begin{aligned}
& + \begin{pmatrix} 0 & 0 \\ 0 & |\hat{C}^{(2)} - \bar{C}^{(2)}| \end{pmatrix} + \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot \begin{pmatrix} 0 & 0 \\ 0 & (|\hat{l}| + e) \cdot (|\hat{u}| + \mu \cdot e)^T \end{pmatrix} \\
\leq & \begin{pmatrix} 0 & 0 \\ 0 & |H^{(2)}| \end{pmatrix} + \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot \begin{pmatrix} 2 \\ |\hat{l}| + e \end{pmatrix} \cdot (|\hat{\gamma}_1| + \mu \cdot (|\hat{u}| + \mu \cdot e)^T) \\
& + \begin{pmatrix} 0 & 0 \\ 0 & |\hat{C}^{(2)} - \bar{C}^{(2)}| \end{pmatrix}, \tag{4.32}
\end{aligned}$$

where we have used the fact that $\xi_{\max} \geq \xi_{\min}$ and $\psi \geq 1$. The induction hypothesis implies that

$$|H^{(2)}| \leq \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot (|\hat{L}^{(2)}| + X_{n-1}) \cdot (|\hat{U}^{(2)}| + \mu \cdot Y_{n-1}) + \frac{\Delta^{(2)}}{\xi_{\min}}.$$

Plugging this relation into (4.32), we have

$$\begin{aligned}
|H| & \leq \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot \begin{pmatrix} 0 \\ |\hat{L}^{(2)}| + X_{n-1} \end{pmatrix} \cdot (0 \quad |\hat{U}^{(2)}| + \mu \cdot Y_{n-1}) \\
& + \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot \begin{pmatrix} 2 \\ |\hat{l}| + e \end{pmatrix} \cdot (|\hat{\gamma}_1| + \mu \cdot (|\hat{u}| + \mu \cdot e)^T) \\
& + \frac{\Delta^{(1)}}{\xi_{\min}} \\
& = \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot \begin{pmatrix} 1+1 & 0 \\ |\hat{l}| + e & |\hat{L}^{(2)}| + X_{n-1} \end{pmatrix} \cdot \begin{pmatrix} |\hat{\gamma}_1| + \mu & (|\hat{u}| + \mu \cdot e)^T \\ 0 & |\hat{U}^{(2)}| + \mu \cdot Y_{n-1} \end{pmatrix} \\
& + \frac{\Delta^{(2)}}{\xi_{\min}} \\
& = \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot (|\hat{L}| + X_n) \cdot (|\hat{U}| + \mu \cdot Y_n) + \frac{\Delta^{(1)}}{\xi_{\min}}.
\end{aligned}$$

Hence relation (4.31) holds for all n . Taking ∞ -norm on both of its sides,

$$\begin{aligned}
\|H\|_{\infty} & \leq \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot \left\| (|\hat{L}| + X_n) \cdot (|\hat{U}| + \mu \cdot Y_n) \right\|_{\infty} + \frac{\|\Delta^{(1)}\|_{\infty}}{\xi_{\min}} \\
& \leq \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot \left\| (|\hat{L}| + X_n) \cdot (|\hat{U}| + \mu \cdot Y_n) \right\|_{\infty} + \frac{n \cdot \tau}{\xi_{\min}}.
\end{aligned}$$

■

It follows from equation (4.23) that μ is an upper bound on the element growth in the computed LU factorization. Thus Theorem 3 shows that the backward error in the computed LU factorization is bounded by $(\alpha\eta(\nu + \psi \cdot \xi_{\max}))/\xi_{\min}$ times the element growth in $|\hat{L}|$ and $|\hat{U}|$ plus the error in computing the generators.

4.4 Error Analysis for Algorithm 1

In this subsection, we assume that partial pivoting has been done before hand, so that Algorithm 1 does not perform any pivoting.

For Algorithm 1, the generators are computed as

$$\hat{A}^{(k+1)} = \text{fl}(\bar{A}^{(k+1)}) \quad \text{and} \quad \hat{B}^{(k+1)} = \text{fl}(\bar{B}^{(k+1)}) ,$$

where $\bar{A}^{(k+1)} = \tilde{A}_{k+1} - \hat{l}^{(k)} \cdot (\hat{a}_k^{(k)})^T$ and $\bar{B}^{(k+1)} = \tilde{B}_{k+1} - \hat{b}_k^{(k)} \cdot (\hat{u}^{(k)})^T / \hat{\gamma}_k$. It follows that

$$\begin{aligned} |\hat{A}^{(k+1)} - \bar{A}^{(k+1)}| &\leq \eta \left(|\tilde{A}_{k+1}| + |\hat{l}^{(k)}| \cdot |\hat{a}_k^{(k)}|^T \right) \\ |\hat{B}^{(k+1)} - \bar{B}^{(k+1)}| &\leq \eta \left(|\tilde{B}_{k+1}| + |\hat{b}_k^{(k)}| \cdot |\hat{u}^{(k)}|^T / |\hat{\gamma}_k| \right) . \end{aligned}$$

Hence

$$\begin{aligned} |\hat{G}^{(k+1)} - \bar{G}^{(k+1)}| &= |\hat{A}^{(k+1)} \cdot \hat{B}^{(k+1)} - \bar{A}^{(k+1)} \cdot \bar{B}^{(k+1)}| \\ &\leq |\hat{A}^{(k+1)} - \bar{A}^{(k+1)}| \cdot |\bar{B}^{(k+1)}| + |\bar{A}^{(k+1)}| \cdot |\hat{B}^{(k+1)} - \bar{B}^{(k+1)}| + O(\epsilon^2) \\ &\leq 2\eta \left(|\tilde{A}_{k+1}| + |\hat{l}^{(k)}| \cdot |\hat{a}_k^{(k)}|^T \right) \cdot \left(|\tilde{B}_{k+1}| + |\hat{b}_k^{(k)}| \cdot |\hat{u}^{(k)}|^T / |\hat{\gamma}_k| \right) + O(\epsilon^2) \\ &= 2\eta \left(|\tilde{A}_{k+1}| \cdot |\tilde{B}_{k+1}| + (|\tilde{A}_{k+1}| \cdot |\hat{b}_k^{(k)}| / |\hat{\gamma}_k|) \cdot |\hat{u}^{(k)}|^T \right) \\ &\quad + 2\eta |\hat{l}^{(k)}| \cdot \left(|\hat{a}_k^{(k)}|^T \cdot |\tilde{B}_{k+1}| + (|\hat{a}_k^{(k)}|^T \cdot |\hat{b}_k^{(k)}| / |\hat{\gamma}_k|) \cdot |\hat{u}^{(k)}|^T \right) + O(\epsilon^2) . \end{aligned}$$

Using definitions (4.21) and (4.22), Lemma 4.1 and the fact that $|\hat{l}^{(k)}| \leq e + O(\epsilon)$, we have

$$\begin{aligned} \|\hat{G}^{(k+1)} - \bar{G}^{(k+1)}\|_{\max} &\leq 2\eta \left(\|\tilde{A}_{k+1}\| \cdot \|\tilde{B}_{k+1}\| + \nu \|e \cdot |\hat{u}^{(k)}|^T\|_{\max} \right) \\ &\quad + 2\eta \left(\|e \cdot |\hat{a}_k^{(k)}|^T\| \cdot \|\tilde{B}_{k+1}\| + \nu \|e \cdot |\hat{u}^{(k)}|^T\|_{\max} \right) + O(\epsilon^2) \\ &\leq 2\eta (\psi \mu \xi_{\max} + \nu \mu + \psi \mu \xi_{\max} + \nu \mu) + O(\epsilon^2) \\ &= 4\eta \mu (\nu + \psi \xi_{\max}) + O(\epsilon^2) . \end{aligned} \tag{4.33}$$

According to definition (4.21), this implies that for Algorithm 1

$$\tau \leq 4n\eta\mu(\nu + \psi\xi_{\max}) + O(\epsilon^2) .$$

Plugging this into (4.24), and using the fact that $\|\hat{L} + X_n\|_{\infty} \leq 2n + O(\epsilon)$, we have

Theorem 4 *For Algorithm 1, the backward error H in the LU factorization of C in (4.18) satisfies*

$$\|H\|_{\infty} \leq \frac{2n\eta \cdot (\alpha + 2) \cdot (\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot \left(\|\hat{U}\|_{\infty} + n \cdot \mu \right) + O(\epsilon^2) .$$

One expects ψ to be of the order 1 in general. The fact that Algorithm 1 performs partial pivoting means that

$$|\hat{\gamma}_k| = \left| \text{fl} \left(\frac{(a_k^{(k)})^T \cdot b_k^{(k)}}{\omega_k - \lambda_k} \right) \right| \geq \left| \text{fl} \left(\frac{(a_i^{(k)})^T \cdot b_k^{(k)}}{\omega_i - \lambda_k} \right) \right|$$

for $1 \leq k \leq i \leq n$. Comparing this with the definition for ν in (4.22), one expects ν to be of the order $\xi_{\max}$ in general. Hence Theorem 4 suggests that in general the backward error for Algorithm 1 is of the order $\epsilon \cdot \xi_{\max} / \xi_{\min} \cdot \|\hat{U}\|_{\infty}$.

However, if both $\hat{A}^{(k)}$ and $\hat{B}^{(k)}$ are ill-conditioned for some k , it could happen that $\psi \gg 1$ and $\nu \gg \xi_{\max}$. If this happens, then the backward error for Algorithm 1 could be much larger.

On the other hand, if the straightforward GEPP is applied to C , then the backward error is basically $\epsilon \cdot \|\hat{U}\|_{\infty}$. Thus Algorithm 1 appears to be less numerically stable than straightforward GEPP on C . These conclusions are consistent with those of Sweet and Brent [23].

4.5 Error Analysis for Algorithm 2

In this subsection, we assume that partial pivoting has been done before hand, so that Algorithm 2 does not perform any pivoting.

For Algorithm 2, $\hat{A}^{(k+1)}$ is the computed Q factor in the QR factorization of $\text{fl}(\bar{A}^{(k+1)})$ and $\hat{B}^{(k+1)}$ is the product of the R factor and $\text{fl}(\bar{B}^{(k+1)})$. In finite arithmetic, let $\text{fl}(\bar{A}^{(k+1)}) = \hat{A}^{(k+1)} \cdot R + E_1$ be the QR factorization of $\text{fl}(\bar{A}^{(k+1)})$, and $\hat{B}^{(k+1)} = R \cdot \text{fl}(\bar{B}^{(k+1)}) + E_2$. It is known that [13] the error matrices satisfy

$$\|E_1\|_2 \leq \eta_1 \alpha \cdot n \cdot \|\text{fl}(\bar{A}^{(k+1)})\|_2 \quad \text{and} \quad \|E_2\|_{\infty} \leq \eta_2 \alpha \cdot \|R\|_{\infty} \cdot \|\text{fl}(\bar{B}^{(k+1)})\|_{\infty}, \quad (4.34)$$

where η_1 and η_2 are small multiples of ϵ . We observe that, after some algebra,

$$\hat{A}^{(k+1)} \cdot \hat{B}^{(k+1)} - \text{fl}(\bar{A}^{(k+1)}) \cdot \text{fl}(\bar{B}^{(k+1)}) = -E_1 \cdot \text{fl}(\bar{B}^{(k+1)}) + \hat{A}^{(k+1)} \cdot E_2. \quad (4.35)$$

In the following, we shall derive an upper bound for τ . To this end, we need to derive norm bounds for some of the related quantities. Since Algorithm 2 performs row pivoting and keeps $\hat{A}^{(k)}$ numerically column orthogonal at every step, we have $\|\hat{l}^{(k)}\|_{\max} \leq 1 + O(\epsilon)$,

$$\|\text{fl}(\bar{A}^{(k+1)})\|_{\infty} \leq \|\hat{A}_{k+1}\|_{\infty} + \|\hat{l}^{(k)} \cdot (\hat{a}_k^{(k)})^T\|_{\infty} + O(\epsilon) \leq 2\sqrt{\alpha} + O(\epsilon), \quad (4.36)$$

and

$$\|R\|_{\infty} = \|(\hat{A}^{(k+1)})^T \cdot (\tilde{A}_{k+1} - \hat{l}^{(k)} \cdot (\hat{a}_k^{(k)})^T)\|_{\infty} + O(\epsilon) \leq \sqrt{\alpha} \cdot (\sqrt{n} + 1) + O(\epsilon). \quad (4.37)$$

Since $\hat{A}^{(k)}$ is numerically column orthogonal, it follows that

$$\|(\Omega_k - \lambda_k I)^{-1} \cdot \hat{A}^{(k)} \cdot \hat{b}_k^{(k)}\|_2 \geq \frac{\|\hat{A}^{(k)} \cdot \hat{b}_k^{(k)}\|_2}{\xi_{\max}} = \frac{\|\hat{b}_k^{(k)}\|_2}{\xi_{\max}} + O(\epsilon).$$

The fact that Algorithm 2 performs row pivoting gives

$$\begin{aligned} |\hat{\gamma}_k| &= \|(\Omega_k - \lambda_k I)^{-1} \cdot \hat{A}^{(k)} \cdot \hat{b}_k^{(k)}\|_{\max} + O(\epsilon) \\ &\geq \frac{1}{\sqrt{n}} \cdot \|(\Omega_k - \lambda_k I)^{-1} \cdot \hat{A}^{(k)} \cdot \hat{b}_k^{(k)}\|_2 + O(\epsilon) \\ &\geq \frac{\|\hat{b}_k^{(k)}\|_2}{\sqrt{n} \cdot \xi_{\max}} + O(\epsilon). \end{aligned} \quad (4.38)$$

With these relations, we get

$$\begin{aligned} \|\mathfrak{f}(\bar{B}^{(k+1)})\|_{\infty} &\leq \|\tilde{B}_{k+1}\|_{\infty} + \|\hat{b}_k^{(k)} \cdot (\hat{u}^{(k)})^T\|_{\infty} / |\hat{\gamma}_k| + O(\epsilon) \\ &\leq \sqrt{\alpha} \cdot \|\hat{B}^{(k)}\|_2 + \|\hat{b}_k^{(k)}\|_2 \cdot \|\hat{U}\|_{\infty} / |\hat{\gamma}_k| + O(\epsilon) \\ &= \sqrt{\alpha} \cdot \|\hat{A}^{(k)} \cdot \hat{B}^{(k)}\|_2 + \|\hat{b}_k^{(k)}\|_2 \cdot \|\hat{U}\|_{\infty} / |\hat{\gamma}_k| + O(\epsilon) \\ &\leq \sqrt{\alpha} \cdot n \cdot \|\hat{A}^{(k)} \cdot \hat{B}^{(k)}\|_{\max} + \sqrt{n} \cdot \xi_{\max} \cdot \|\hat{U}\|_{\infty} + O(\epsilon) \\ &\leq \sqrt{\alpha} \cdot n \cdot \xi_{\max} \|\hat{C}^{(k)}\|_{\max} + \sqrt{n} \cdot \xi_{\max} \cdot \|\hat{U}\|_{\infty} + O(\epsilon) \\ &\leq \sqrt{\alpha} \cdot n \cdot \xi_{\max} \cdot \mu + \sqrt{n} \cdot \xi_{\max} \cdot \|\hat{U}\|_{\infty} + O(\epsilon). \end{aligned} \quad (4.39)$$

To obtain an upper bound on τ , we now take ∞ -norm on both sides of (4.35). Using relations (4.34) through (4.37) and (4.39), the right hand side of (4.35) is bounded above by

$$\begin{aligned} &\|E_1\|_{\infty} \cdot \|\mathfrak{f}(\bar{B}^{(k+1)})\|_{\infty} + \|\hat{A}^{(k+1)}\|_{\infty} \cdot \|E_2\|_{\infty} \\ &\leq \|E_1\|_{\infty} \cdot \|\mathfrak{f}(\bar{B}^{(k+1)})\|_{\infty} + \sqrt{\alpha} \cdot \|E_2\|_{\infty} + O(\epsilon^2) \\ &\leq \eta_1 \alpha n \|\mathfrak{f}(\bar{A}^{(k+1)})\|_{\infty} \cdot \|\mathfrak{f}(\bar{B}^{(k+1)})\|_{\infty} + \eta_2 \alpha^{\frac{3}{2}} \cdot \|R\|_{\infty} \cdot \|\mathfrak{f}(\bar{B}^{(k+1)})\|_{\infty} + O(\epsilon^2) \\ &= \alpha \cdot \left(\eta_1 \cdot n \cdot \|\mathfrak{f}(\bar{A}^{(k+1)})\|_{\infty} + \eta_2 \cdot \sqrt{\alpha} \cdot \|R\|_{\infty} \right) \cdot \|\mathfrak{f}(\bar{B}^{(k+1)})\|_{\infty} + O(\epsilon^2) \\ &\leq \alpha \cdot \left(2\eta_1 \cdot n \cdot \sqrt{\alpha} + \eta_2 \cdot \alpha \cdot (\sqrt{n} + 1) \right) \cdot \left(\sqrt{\alpha} \cdot n \cdot \mu \cdot \xi_{\max} + \sqrt{n} \cdot \|\hat{U}\|_{\infty} \cdot \xi_{\max} \right) + O(\epsilon^2) \\ &\leq 4\bar{\eta} \cdot (\alpha \cdot n)^{\frac{3}{2}} \cdot (n \cdot \mu + \|\hat{U}\|_{\infty}) \cdot \xi_{\max} + O(\epsilon^2), \end{aligned}$$

where $\bar{\eta} = \max\{\eta, \eta_1, \eta_2\}$, and we have used the fact that $\alpha \leq n$. Hence

$$\|\hat{A}^{(k+1)} \cdot \hat{B}^{(k+1)} - \mathfrak{f}(\bar{A}^{(k+1)}) \cdot \mathfrak{f}(\bar{B}^{(k+1)})\|_{\infty} \leq 4\bar{\eta} \cdot (\alpha \cdot n)^{\frac{3}{2}} \cdot (n \cdot \mu + \|\hat{U}\|_{\infty}) \cdot \xi_{\max} + O(\epsilon^2).$$

In addition, a derivation similar to that for equation (4.33) gives

$$\|\mathfrak{f}(\bar{A}^{(k+1)}) \cdot \mathfrak{f}(\bar{B}^{(k+1)}) - \bar{A}^{(k+1)} \cdot \bar{B}^{(k+1)}\|_{\infty} \leq 4n\eta\mu(\nu + \psi\xi_{\max}) + O(\epsilon^2).$$

Combining these two relations, we get

$$\begin{aligned}
\|\hat{G}^{(k+1)} - \bar{G}^{(k+1)}\|_\infty &= \|\hat{A}^{(k+1)} \cdot \hat{B}^{(k+1)} - \bar{A}^{(k+1)} \cdot \bar{B}^{(k+1)}\|_\infty \\
&\leq \|\hat{A}^{(k+1)} \cdot \hat{B}^{(k+1)} - \mathfrak{fl}(\bar{A}^{(k+1)}) \cdot \mathfrak{fl}(\bar{B}^{(k+1)})\|_\infty \\
&\quad + \|\mathfrak{fl}(\bar{A}^{(k+1)}) \cdot \mathfrak{fl}(\bar{B}^{(k+1)}) - \bar{A}^{(k+1)} \cdot \bar{B}^{(k+1)}\|_\infty \\
&\leq 4\bar{\eta} \cdot (\alpha \cdot n)^{\frac{3}{2}} \cdot (n \cdot \mu + \|\hat{U}\|_\infty) \cdot \xi_{\max} + 4\bar{\eta}n\mu(\nu + \psi\xi_{\max}) + O(\epsilon^2) .
\end{aligned}$$

According to definition (4.21), this implies that for Algorithm 2

$$\tau \leq 4\bar{\eta} \cdot (\alpha \cdot n)^{\frac{3}{2}} \cdot (n \cdot \mu + \|\hat{U}\|_\infty) \cdot \xi_{\max} + 4\bar{\eta}n\mu(\nu + \psi\xi_{\max}) + O(\epsilon^2) . \quad (4.40)$$

Theorem 5 *For Algorithm 2, the backward error H in the LU factorization of C in (4.18) satisfies*

$$\|H\|_\infty \leq 8\sqrt{\alpha}\bar{\eta}(\alpha + 2) \cdot n^2 \cdot \rho \cdot (\|\hat{U}\|_\infty + n \cdot \mu) + O(\epsilon^2) , \quad (4.41)$$

where ρ is defined in (2.7).

Proof. We first derive upper bounds for ν and ψ , and then finish the proof by plugging relation (4.40) and these bounds into (4.30).

Since $\hat{A}^{(k)}$ is numerically column orthogonal,

$$|\hat{a}_i^{(k)}|^T \cdot |\hat{b}_k^{(k)}| \leq \|\hat{a}_i^{(k)}\|_2 \cdot \|\hat{b}_k^{(k)}\|_2 \leq \|\hat{b}_k^{(k)}\|_2 + O(\epsilon) .$$

Combining this and equation (4.38) to get

$$\frac{|\hat{a}_i^{(k)}|^T \cdot |\hat{b}_k^{(k)}|}{|\hat{\gamma}_k|} \leq \sqrt{n} \cdot \xi_{\max} + O(\epsilon) .$$

By definition (4.22), this implies

$$\nu \leq \sqrt{n} \cdot \xi_{\max} + O(\epsilon) . \quad (4.42)$$

On the other hand,

$$\begin{aligned}
\| |\hat{A}^{(k)}| \cdot |\hat{B}^{(k)}| \|_{\max} &\leq \| |\hat{A}^{(k)}| \cdot |\hat{B}^{(k)}| \|_2 \leq \| |\hat{A}^{(k)}| \|_2 \cdot \| |\hat{B}^{(k)}| \|_2 \\
&\leq \| \hat{A}^{(k)} \|_F \cdot \| \hat{B}^{(k)} \|_F = \sqrt{\alpha} \| \hat{B}^{(k)} \|_F + O(\epsilon) ,
\end{aligned}$$

and

$$\| \hat{A}^{(k)} \cdot \hat{B}^{(k)} \|_{\max} \geq \frac{1}{n} \| \hat{A}^{(k)} \cdot \hat{B}^{(k)} \|_F = \frac{1}{n} \| \hat{B}^{(k)} \|_F + O(\epsilon) .$$

Consequently,

$$\frac{\| |\hat{A}^{(k)}| \cdot |\hat{B}^{(k)}| \|_{\max}}{\| \hat{A}^{(k)} \cdot \hat{B}^{(k)} \|_{\max}} \leq \sqrt{\alpha} \cdot n + O(\epsilon) .$$

By definition (4.22), this implies

$$\psi \leq \sqrt{\alpha} \cdot n + O(\epsilon). \quad (4.43)$$

Plugging relations (4.40) through (4.43) into (4.30), we get

$$\begin{aligned} \|H\|_\infty &\leq \frac{\alpha\eta(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot 2n \cdot (\|\hat{U}\|_\infty + n \cdot \mu) + \frac{n\tau}{\xi_{\min}} + O(\epsilon^2) \\ &\leq \frac{2n\bar{\eta}(\alpha + 2)(\nu + \psi \cdot \xi_{\max})}{\xi_{\min}} \cdot (\|\hat{U}\|_\infty + n \cdot \mu) \\ &\quad + \frac{4\bar{\eta} \cdot (\alpha \cdot n)^{\frac{3}{2}}}{\xi_{\min}} \cdot (\|\hat{U}\|_\infty + n \cdot \mu) + O(\epsilon^2) \\ &\leq \frac{2n\bar{\eta}(\alpha + 2)(\sqrt{n} \cdot \xi_{\max} + \sqrt{\alpha} \cdot n \cdot \xi_{\max})}{\xi_{\min}} \cdot (\|\hat{U}\|_\infty + n \cdot \mu) \\ &\quad + \frac{4\bar{\eta} \cdot (\alpha \cdot n)^{\frac{3}{2}}}{\xi_{\min}} \cdot (\|\hat{U}\|_\infty + n \cdot \mu) + O(\epsilon^2) \\ &\leq 8\sqrt{\alpha}\bar{\eta} \cdot (\alpha + 2) \cdot n^2 \cdot \rho \cdot (\|\hat{U}\|_\infty + n \cdot \mu) + O(\epsilon^2). \end{aligned}$$

■

Remark 6: More detailed error analysis shows that the $O(\epsilon^2)$ term in (4.41) is bounded by $p_1(n)\epsilon$ times the first term, where $p_1(n)$ is a low degree polynomial in n .

Remark 7: Throughout this analysis, we never used the fact that Algorithm 2 performs pivoting on the columns as well. Hence, Theorem 5 still holds if Algorithm 2 is modified to only perform partial pivoting.

Remark 8: An upper bound similar to (4.41) holds for Algorithm 3 as well, provided that

$$\| |\hat{A}^{(k)}| \cdot |\hat{B}^{(k)}| \|_2 \approx \| \hat{A}^{(k)} \cdot \hat{B}^{(k)} \|_2$$

for all k , as is the case in our numerical experiments.

Remark 9: If C is transformed from a Toeplitz-plus-Hankel matrix via equation (3.13), then $\rho = O(n^2)$. In this case, the upper bound (4.41) is a factor of $O(\alpha^{\frac{3}{2}} n^2)$ larger than the upper bound for the backward error in straightforward GEPP and GECP, which is about $5\epsilon \cdot n^2 \cdot \|\hat{U}\|_\infty$ (see [13, page 115]). Our numerical experiments indicate that for such matrices, Algorithm 2 is sometimes less accurate than straightforward GEPP, and the lost accuracy can be recovered by one step of iterative refinement [13, §3.5]. See §5 for more details.

Finally, we perform a brief error analysis for Algorithm 4. Let M be a Toeplitz-plus-Hankel-like matrix satisfying equation (3.12) with A column orthogonal. In finite arithmetic, Algorithm 4 factorizes M by performing the following computations.

- Compute $\hat{\Omega} = \text{fl}(\mathcal{D}_{1,1})$ and $\hat{\Lambda} = \text{fl}(\mathcal{D}_{1,-1})$.
- Compute $\hat{\mathcal{A}} = \text{fl}(\mathcal{Q}_{1,1}^T \cdot A)$ and $\hat{\mathcal{B}} = \text{fl}(B \cdot \mathcal{Q}_{1,-1})$.

- Compute the LU factorization for $\hat{C}$, where $\hat{C}$ is the Cauchy-like matrix that satisfies the displacement equation

$$\hat{\Omega} \cdot \hat{C} - \hat{C} \cdot \hat{\Lambda} = \hat{A} \cdot \hat{B}.$$

It is easy to show that

$$\|\mathcal{Q}_{1,1}^T \cdot M \cdot \mathcal{Q}_{1,-1} - \hat{C}\|_{\infty} \leq \eta_3 \cdot p_2(n) \cdot \|M\|_{\infty},$$

where η_3 is a small multiple of ϵ and $p_2(n)$ is a low degree polynomial in n . Thus the reduction from a Toeplitz-plus-Hankel-like matrix satisfying equation (3.12) to a Cauchy-like matrix satisfying (3.13) is numerically stable. In other words, Algorithm 4 is numerically stable if and only if the algorithm it uses in Step 2 (Algorithm 1, 2, or 3) is stable.

5 Numerical Experiments

We have implemented Algorithm 4 in Fortran and have performed a large number of numerical experiments with it to investigate its behavior in finite arithmetic and to compare it with other available algorithms. In this section we discuss some implementation issues and report some of these numerical experiments. We chose Algorithm 3 with $K = 10$ in Step 2 of Algorithm 4.

5.1 Implementation Issues

A natural way to implement Algorithm 3 is to keep permutations P and Q in vectors and keep both L and U in a single matrix W by storing L in the strict lower triangular part of W (excluding the diagonal) and U upper triangular part (including the diagonal).

However, arrays are stored column-wise in Fortran. Note that U is generated row by row in Algorithm 3. In order to store U , columns of W have to be moved into and brought out of fast memory for most steps of elimination for large n . This causes a significant amount of memory traffic between slow and fast memory levels in the memory hierarchy. For more detailed discussions on memory traffic, see for example [9, §2.6].

We reduce this memory traffic by storing rows of U column-wise in Algorithm 3. Let $S \in \mathbf{R}^{n \times n}$ be the matrix that is 1 on the main anti-diagonal and 0 everywhere else. For $n = 2$,

$$S = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

It follows that $\tilde{U} \equiv S \cdot U^T \cdot S$ is an upper triangular matrix, whose k^{th} column is the $(n - k + 1)^{st}$ row of U in the reverse order. The backward substitution procedure

for computing $U^{-1} \cdot y$ in Algorithm 3 can be rewritten as a forward substitution as $S \cdot \left((\tilde{U}^T)^{-1} \cdot (S \cdot y) \right)$. Our numerical experiments indicate that this technique speeds up both Algorithm 3 and Algorithm 4 by up to a factor of 2 (see Table 1).

Our numerical experiments indicate that Algorithm 2 is slightly less accurate than straightforward GEPP in many cases. Hence we perform one step of iterative refinement for Algorithm 4 to get (see [13, §3.5]):

Algorithm 5 Solving $M \cdot x = z$ with iterative refinement.

1. Compute the factorization $M = Q_{1,1}^T \cdot P \cdot L \cdot U \cdot Q \cdot Q_{1,-1}^T$ and the solution $x^{(1)} = Q_{1,-1} \cdot Q^T \cdot U^{-1} \cdot L^{-1} \cdot P^T \cdot Q_{1,1} \cdot z$ using Algorithm 4.
2. Compute the residual $r^{(1)} = z - M \cdot x^{(1)}$.
3. Compute the refined solution $x^{(2)} = x^{(1)} + Q_{1,-1} \cdot Q^T \cdot U^{-1} \cdot L^{-1} \cdot P^T \cdot Q_{1,1} \cdot r^{(1)}$.
4. Compute $k_{\max} = \operatorname{argmin}_{1 \leq k \leq 2} \|z - M \cdot x^{(k)}\|$.
5. Return $x^{(k_{\max})}$ as the computed solution.

The norm in Step 4 can be any operator norm.

Our numerical experiments show that Algorithm 5 is in general more accurate than Algorithm 4. Since the residual vectors $z - M \cdot x^{(k)}$ can be computed in $O(n \log_2 n)$ flops using convolution (see [21]), the extra cost for computing $x^{(k_{\max})}$ involves basically a forward and a backward substitution, about $2n^2$ flops, an increase of about 10% of the cost of Algorithm 4 (see §3.1).

5.2 Numerical Results

The computations were done on an IBM RS6000 workstation in double precision where the machine precision is $\epsilon \approx 1.1 \times 10^{-16}$.

We compared the following algorithms:

- **GEPP-I**: LAPACK [1] subroutines DGETRF+DGETRS for solving a general dense linear systems of equations using GEPP, with Fortran BLAS and without iterative refinement; cost: $O(n^3)$ flops.
- **GEPP-II**: LAPACK routines DGETRF+DGETRS for solving a general dense linear systems of equations using GEPP, with optimized BLAS and one step of iterative refinement; cost: $O(n^3)$ flops.
- **LEVIN**: The algorithm available on Netlib; cost: $O(n^2)$ flops.
- **NEW-I**: Implementation of Algorithm 4 by storing rows of U row-wise and with no iterative refinement; cost: $O(n^2)$ flops.

- **NEW-II:** Implementation of Algorithm 4 by storing rows of U column-wise and with one step of iterative refinement; cost: $O(n^2)$ flops.

We solve Toeplitz linear systems of equations $T \cdot x = z$ for random right hand side vectors z and the following types of Toeplitz matrices $T = (t_{k-j})_{1 \leq k, j \leq n}$:

- *Type 1:* $\{t_k\}$ randomly generated from uniform distribution on $(0, 1)$. A Type 1 matrix is usually well-conditioned.
- *Type 2:* $t_0 = 2\omega$ and $t_k = \frac{\sin(2\pi\omega k)}{\pi k}$ for $k \neq 0$. $\omega \in [0, 1/2]$ is a parameter. A Type 2 matrix is also called the Prolate matrix in [10, 24]; it is very ill-conditioned for small ω . In our experiments we took $\omega = 0.25$.
- *Type 3:* $t_k = a^{k^2}$ with $0 < a < 1$. A Type 3 matrix is also called the Gauss matrix in [10]; it is very ill-conditioned for a close to 1. In our experiments we took $a = 0.95$.
- *Type 4:* t_0 is randomly generated from uniform distribution in $(0.9, 1)$; $t_k = -t_0$ for $k > 0$; $t_k = 0$ for $-n/2 < k < 0$; and the rest are randomly generated from uniform distribution in $(0, 1)$. The straightforward GEPP produces huge element growth on a Type 4 matrix.

Our numerical results are summarized in Table 1. NEW-II is faster than NEW-I by a factor of up to 2 for large n and for all four types of matrices; and is more accurate than NEW-I for Types 1 and 4 matrices. On the other hand, GEPP-II is as accurate as GEPP-I, but is up to a factor of 2 faster. For $n = 2560$, NEW-II is up to 17 times faster than GEPP-I and up to 10 times faster than GEPP-II, respectively; whereas LEVIN is only up to 3 times faster than GEPP-I and up to 2 times faster than GEPP-II, respectively. Both NEW-I and NEW-II solve all linear systems successfully, whereas GEPP-I, GEPP-II, and LEVIN fail on Type 4 matrices.

6 Extensions and Conclusions

We have presented a fast algorithm for solving Toeplitz or Toeplitz-plus-Hankel systems of linear equations and shown it to be numerically stable, provided that the element growth in the computed factorization is not large. We have presented practical modifications to this algorithm and discussed implementation techniques that further improve its efficiency. Our numerical experiments show that the resulting algorithm is both stable and efficient; and the cost for performing pivoting for Cauchy-like matrices can be kept a small fraction of the total cost.

The algorithms presented in this paper can be modified to solve *Mosaic Toeplitz* or *Block Toeplitz* systems of linear equations (see [7, 10]).

Our techniques to avoid internal element growth in the generators can be easily extended to the *generalized Schur algorithm* for factorizing more generally structured

MATRIX TYPE	ORDER n	EXECUTION TIME (SECONDS)				
		GEPP-I	GEPP-II	LEVIN	NEW-I	NEW-II
Type 1	160	$.3 \times 10^{-1}$	$.3 \times 10^{-1}$	$.1 \times 10^0$	$.3 \times 10^{-1}$	$.3 \times 10^{-1}$
	320	$.2 \times 10^0$	$.2 \times 10^0$	$.4 \times 10^0$	$.9 \times 10^{-1}$	$.1 \times 10^0$
	640	$.2 \times 10^1$	$.2 \times 10^1$	$.2 \times 10^1$	$.5 \times 10^0$	$.4 \times 10^0$
	1280	$.2 \times 10^2$	$.1 \times 10^2$	$.7 \times 10^1$	$.2 \times 10^1$	$.1 \times 10^1$
	2560	$.1 \times 10^3$	$.7 \times 10^2$	$.3 \times 10^2$	$.1 \times 10^2$	$.6 \times 10^1$
Type 2	160	$.3 \times 10^{-1}$	$.3 \times 10^{-1}$	$.1 \times 10^0$	$.2 \times 10^{-1}$	$.3 \times 10^{-1}$
	320	$.2 \times 10^0$	$.1 \times 10^0$	$.5 \times 10^0$	$.9 \times 10^{-1}$	$.9 \times 10^{-1}$
	640	$.2 \times 10^1$	$.1 \times 10^1$	$.2 \times 10^1$	$.5 \times 10^0$	$.4 \times 10^0$
	1280	$.2 \times 10^2$	$.1 \times 10^2$	$.7 \times 10^1$	$.2 \times 10^1$	$.1 \times 10^1$
	2560	$.1 \times 10^3$	$.7 \times 10^2$	$.3 \times 10^2$	$.1 \times 10^2$	$.6 \times 10^1$
Type 3	160	$.3 \times 10^{-1}$	$.3 \times 10^{-1}$	$.1 \times 10^0$	$.3 \times 10^{-1}$	$.3 \times 10^{-1}$
	320	$.2 \times 10^0$	$.2 \times 10^0$	$.5 \times 10^0$	$.9 \times 10^{-1}$	$.9 \times 10^{-1}$
	640	$.1 \times 10^1$	$.1 \times 10^1$	$.2 \times 10^1$	$.5 \times 10^0$	$.3 \times 10^0$
	1280	$.5 \times 10^1$	$.1 \times 10^2$	$.7 \times 10^1$	$.2 \times 10^1$	$.1 \times 10^1$
	2560	$.3 \times 10^2$	$.7 \times 10^2$	$.3 \times 10^2$	$.1 \times 10^2$	$.5 \times 10^1$
Type 4	160	$.2 \times 10^{-1}$	$.2 \times 10^{-1}$	$.1 \times 10^0$	$.3 \times 10^{-1}$	$.3 \times 10^{-1}$
	320	$.1 \times 10^0$	$.1 \times 10^0$	$.4 \times 10^0$	$.9 \times 10^{-1}$	$.1 \times 10^0$
	640	$.1 \times 10^1$	$.1 \times 10^1$	$.2 \times 10^1$	$.5 \times 10^0$	$.3 \times 10^0$
	1280	$.1 \times 10^2$	$.8 \times 10^1$	$.7 \times 10^1$	$.2 \times 10^1$	$.1 \times 10^1$
	2560	*	*	*	$.1 \times 10^2$	$.6 \times 10^1$
MATRIX TYPE	ORDER n	$\frac{\ Tx - b\ _1}{\sqrt{n} \cdot \epsilon \cdot (\ T\ _1 \cdot \ x\ _1 + \ b\ _1)}$				
		GEPP-I	GEPP-II	LEVIN	NEW-I	NEW-II
Type 1	160	$.8 \times 10^{-1}$	$.9 \times 10^{-1}$	$.4 \times 10^1$	$.1 \times 10^2$	$.9 \times 10^{-1}$
	320	$.9 \times 10^{-1}$	$.8 \times 10^{-1}$	$.3 \times 10^2$	$.4 \times 10^2$	$.1 \times 10^0$
	640	$.8 \times 10^{-1}$	$.1 \times 10^0$	$.1 \times 10^3$	$.4 \times 10^2$	$.5 \times 10^{-1}$
	1280	$.2 \times 10^0$	$.2 \times 10^0$	$.2 \times 10^2$	$.7 \times 10^2$	$.2 \times 10^0$
	2560	$.9 \times 10^{-1}$	$.9 \times 10^{-1}$	$.3 \times 10^2$	$.2 \times 10^3$	$.9 \times 10^{-1}$
Type 2	160	$.8 \times 10^{-1}$	$.1 \times 10^0$	$.1 \times 10^1$	$.5 \times 10^0$	$.5 \times 10^0$
	320	$.9 \times 10^{-1}$	$.9 \times 10^{-1}$	$.1 \times 10^1$	$.4 \times 10^0$	$.4 \times 10^0$
	640	$.8 \times 10^{-1}$	$.1 \times 10^0$	$.7 \times 10^0$	$.2 \times 10^0$	$.2 \times 10^0$
	1280	$.7 \times 10^{-1}$	$.9 \times 10^{-1}$	$.2 \times 10^1$	$.2 \times 10^0$	$.2 \times 10^0$
	2560	$.7 \times 10^{-1}$	$.7 \times 10^{-1}$	$.7 \times 10^3$	$.7 \times 10^0$	$.7 \times 10^0$
Type 3	160	$.2 \times 10^{-1}$	$.3 \times 10^{-1}$	$.8 \times 10^{-1}$	$.1 \times 10^1$	$.1 \times 10^1$
	320	$.2 \times 10^{-1}$	$.2 \times 10^{-1}$	$.7 \times 10^{-1}$	$.9 \times 10^0$	$.9 \times 10^0$
	640	$.1 \times 10^{-1}$	$.2 \times 10^{-1}$	$.6 \times 10^{-1}$	$.1 \times 10^1$	$.1 \times 10^1$
	1280	$.9 \times 10^{-2}$	$.1 \times 10^{-1}$	$.5 \times 10^{-1}$	$.5 \times 10^0$	$.5 \times 10^0$
	2560	$.7 \times 10^{-2}$	$.9 \times 10^{-2}$	$.3 \times 10^{-1}$	$.5 \times 10^0$	$.5 \times 10^0$
Type 4	160	$.4 \times 10^{15}$	$.4 \times 10^{15}$	$.6 \times 10^{13}$	$.3 \times 10^1$	$.1 \times 10^0$
	320	$.2 \times 10^{15}$	$.2 \times 10^{15}$	$.4 \times 10^{13}$	$.5 \times 10^1$	$.2 \times 10^{-1}$
	640	$.2 \times 10^{15}$	$.2 \times 10^{15}$	$.2 \times 10^{13}$	$.1 \times 10^2$	$.4 \times 10^{-1}$
	1280	$.1 \times 10^{15}$	$.1 \times 10^{15}$	$.9 \times 10^{13}$	$.5 \times 10^1$	$.1 \times 10^0$
	2560	*	*	*	$.4 \times 10^1$	$.2 \times 10^{-1}$

Table 1: Execution Times and Relative Residuals

matrices (see [6, 20, 21]), so is the technique to store the rows of the U matrix column-wise.

Recently, Chandrasekaran and Sayed [5] propose a new fast algorithm for factorizing the Toeplitz matrix based on the QR factorization of a larger structured matrix and show that it is numerically stable. This algorithm appears to perform more flops than Algorithm 4, but does not have the potential problem of having large element growth in the computed factorization.

We end this paper by asking two open questions.

1. The upper bounds for element growth on GEPP and the variation of GECP in Algorithm 2 on a Cauchy-like matrix are 2^{n-1} and $\rho^{2+\sum_{k=1}^{n-1} 1/k} \cdot \mathcal{W}(n)$, respectively (see § 2). Do sharper bounds exist for Cauchy-like matrices with low displacement rank?
2. There are super-fast algorithms for solving Toeplitz or Toeplitz-plus-Hankel systems of linear equations in $O(n \log_2^2 n)$ flops, but they are unstable in general (see [3]). Are there numerically stable super-fast algorithms for such problems?

Acknowledgements. The author is grateful to Profs. S. Chandrasekaran, J. Demmel, and Dr. V. Olshevsky for helpful discussions, and Profs. S. Chandrasekaran, I. Gohberg, G. Heinig, T. Kailath, and A. H. Sayed, and Dr. V. Olshevsky for preprints of their papers.

References

- [1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, second edition, 1994.
- [2] E. Bozzo and C. Di Fiore. On the use of certain matrix algebras associated with discrete trigonometric transformations in matrix displacement decomposition. Preprint, 1993.
- [3] J. R. Bunch. Stability of methods for solving Toeplitz systems of equations. *SIAM J. Sci. Stat. Comp.*, 6:349–364, 1985.
- [4] T. F. Chan and P. C. Hansen. A lookahead Levinson algorithm for general Toeplitz systems. *IEEE Proc. Signal Processing*, 40:1079–1090, 1992.
- [5] S. Chandrasekaran and A. H. Sayed. A fast and stable solver for nonsymmetric structured systems. Preprint, submitted for publication, 1995.

- [6] S. Chandrasekaran and A. H. Sayed. Stabilizing the fast generalized Schur algorithm. Preprint, submitted for publication, 1995.
- [7] J. Chun and T. Kailath. Generalized displacement structure for block-Toeplitz, Toeplitz-block, and Toeplitz-derived matrices. *SIAM J. Matrix Anal. Appl.*, 15:114–128, 1994.
- [8] G. Cybenko. The numerical stability of Levinson-Durbin algorithm for Toeplitz systems of equations. *SIAM J. Sci. Stat. Comput.*, 1:303–319, 1980.
- [9] J. Demmel. Berkeley Lecture Notes in Numerical Linear Algebra. Mathematics Department, University of California, 1993.
- [10] I. Gohberg, T. Kailath, and V. Olshevsky. Fast Gaussian elimination with partial pivoting for matrices with displacement structure. *Mathematics of Computation*, 1995. to appear.
- [11] I. Gohberg and V. Olshevsky. Fast algorithm for matrix Nehari problem. In U. Helmke, R. Mennicken, and J. Sauer, editors, *Systems and Networks: Mathematical Theory and Applications, Proceedings of the International Symposium MTNS-93*, volume 2, pages 687–690, 1994.
- [12] I. Gohberg and V. Olshevsky. Fast state space algorithms for matrix Nehari and Nehari-Takagi interpolation problems. *Integral Equations and Operator Theory*, 20:44–83, 1994.
- [13] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 2nd edition, 1989.
- [14] M. H. Gutknecht and Hochbruck M. Look-ahead Levinson and Schur algorithms for non-Hermitian Toeplitz systems. IPS Research Rept. 93-11, IPS Supercomputing, ETH-Zürich, August 1993.
- [15] G. Heinig. Inversion of generalized Cauchy matrices and other classes of structured matrices. In *Linear Algebra in Signal Processing, IMA volumes in Mathematics and its Applications*, volume 69, pages 95 – 114, 1994.
- [16] G. Heinig, P. Jankowski, and K. Rost. Fast inversion of Toeplitz-plus-Hankel matrices. *Numer. Math.*, 52:665–682, 1988.
- [17] G. Heinig and K. Rost. Algebraic methods for Toeplitz-like matrices and operators. *Operator Theory*, 13:109–127, 1984.
- [18] A. S. Householder. *The Theory of Matrices in Numerical Analysis*. Dover Publications, Inc., 1964.

- [19] T. Kailath, S. Kung, and M. Morf. Displacement ranks of matrices and linear equations. *J. Math. Anal. and Appl.*, 68:395–407, 1979.
- [20] T. Kailath and A. H. Sayed. Fast algorithms for generalized displacement structures. In H. Kimura and S. Kodama, editors, *Recent Advances in Mathematical Theory of Systems, Control, Networks and Signal Processing*, volume II, pages 27–32. Mita Press, Japan, 1992.
- [21] T. Kailath and A. H. Sayed. Displacement structure: Theory and applications. *SIAM Review*, 1995. to appear.
- [22] D. R. Sweet. The use of pivoting to improve the numerical performance of Toeplitz matrix algorithms. *SIAM J. Matrix Anal. Appl.*, 14:468–493, 1993.
- [23] D. R. Sweet and R. P. Brent. Error analysis of a fast partial pivoting method for structured matrices. In T. Luk, editor, *Advanced Signal Processing algorithms, Proc. of SPIE*, volume 2363, pages 266–280, 1995.
- [24] J. M. Varah. The Prolate matrix. *Lin. Alg. Appl.*, 187:269–278, 1993.
- [25] J. H. Wilkinson. Error analysis of direct methods of matrix inversion. *J. ACM*, 10:281–330, 1961.