SIAM J. MATRIX ANAL. APPL. © 2007 Society for Industrial and Applied Mathematics
Vol. 29, No. 3, pp. 731–751

ACCURATE COMPUTATIONS WITH TOTALLY NONNEGATIVE MATRICES*

PLAMEN KOEV†

Abstract. We consider the problem of performing accurate computations with rectangular $(m \times n)$ totally nonnegative matrices. The matrices under consideration have the property of having a unique representation as products of nonnegative bidiagonal matrices. Given that representation, one can compute the inverse, LDU decomposition, eigenvalues, and SVD of a totally nonnegative matrix to high relative accuracy in $\mathcal{O}(\max(m^3, n^3))$ time—much more accurately than conventional algorithms that ignore that structure. The contribution of this paper is to show that the high relative accuracy is preserved by operations that preserve the total nonnegativity—taking a product, re-signed inverse (when $m = n$), converse, Schur complement, or submatrix of a totally nonnegative matrix, any of which costs at most $\mathcal{O}(\max(m^3, n^3))$. In other words, the class of totally nonnegative matrices for which we can do numerical linear algebra very accurately in $\mathcal{O}(\max(m^3, n^3))$ time (namely, those for which we have a product representation via nonnegative bidiagonals) is closed under the operations listed above.

Key words. high relative accuracy, totally positive matrix, totally nonnegative matrix, bidiagonal decomposition

AMS subject classifications. 65F15, 15A18

DOI. 10.1137/04061903X

1. Introduction.

The matrices with all minors nonnegative are called *totally nonnegative* and appear in a wide variety of applications [5, 10, 12, 14, 15, 18, 25]. They are often very ill conditioned, which means that conventional matrix algorithms such as LAPACK [1] may deliver little or no accuracy when solving totally nonnegative linear systems or computing inverses, eigenvalues, or SVDs.

Our goal is to derive algorithms for performing accurate and efficient computations with $m \times n$ totally nonnegative matrices. The types of computations we would like to perform include computing the inverse, LDU decomposition, eigenvalues, and SVD. By *accurate* we mean that each quantity must be computed *to high relative accuracy*—it must have a correct sign and leading digits. By *efficient* we mean *in at most* $\mathcal{O}(\max(m^3, n^3))$ *time.*

It turns out that the problem of performing accurate computations with totally nonnegative matrices is very much a *representation problem.* If, instead of representing a matrix by its entries, we represent it as a product of nonnegative bidiagonal matrices

$$A = L^{(1)}L^{(2)} \cdots L^{(m-1)}DU^{(n-1)}U^{(n-2)} \cdots U^{(1)}; \tag{1.1}$$

then given the entries of $L^{(k)}$, D, and $U^{(k)}$, we can compute A^{-1}, the LDU decomposition, the eigenvalues, and the SVD of A accurately and efficiently (see section 3).

The existence and uniqueness of the bidiagonal decomposition (1.1) is critical to the design of our algorithms. Therefore we restrict the class of totally nonnegative matrices under consideration to only those that are *leading contiguous submatrices of square nonsingular totally nonnegative matrices.* If the matrix under consideration

*Received by the editors November 16, 2004; accepted for publication (in revised form) by Uwe Helmke December 28, 2006; published electronically June 12, 2007. This work was supported by National Science Foundation grant DMS-0314286.
http://www.siam.org/journals/simax/29-3/61903.html

†Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139 (plamen@math.mit.edu).

is square ($m = n$), the above restriction means that the matrix itself is *nonsingular totally nonnegative.*

We will call the matrices in the above described class TN for short.

The representation (1.1) is intrinsic [19] and immediately reveals the TN structure of an $m \times n$ TN matrix A. The $m \cdot n$ nontrivial nonnegative entries in the factors of (1.1) parameterize the set of all $m \times n$ TN matrices and determine the quantities that we would like to compute (the entries of A^{-1}, the entries of the LDU decomposition, the eigenvalues, and the SVD) accurately (section 3).

TN matrices can be obtained in a variety of ways as a result of matrix operations that preserve the total nonnegativity. The following result is well known [19, 20, 25].

Proposition 1.1. *If $A = [a_{ij}]$ is TN, then so are the product AB, where B is also TN; the* converse $[a_{m+1-i,n+1-j}]$*; the matrix R from a QR decomposition of A such that $r_{ii} > 0$; the Schur complement of a_{11} in A; and any submatrix of A. If A is also square, then the* re-signed inverse $[(-1)^{i+j}a_{ij}]^{-1}$ *is TN. If A also is square and symmetric, then one step of QR iteration with no pivoting preserves the TN structure in A, provided that R has a positive diagonal.*

If A is a TN matrix obtained from other TN matrices by any sequence of these operations, the question becomes: Can we perform accurate matrix computations with A? In other words, if these other TN matrices are represented by their corresponding bidiagonal decompositions (1.1), can we accurately and efficiently compute the bidiagonal decomposition of A?

Our main contribution in this paper is to answer this question affirmatively. In section 5 we present accurate and efficient algorithms that perform these computations. These algorithms prove the following theorem.

Theorem 1.2. *Let every TN matrix be represented by its bidiagonal decomposition* (1.1)*. Then the set of TN matrices with which we can perform accurate and efficient matrix computations, including computing the inverse, LDU decomposition, eigenvalues (when $m = n$), and SVD, is closed under all transformations listed in Proposition* 1.1 *that preserve the TN structure.*

For example, we could take the product of the Hilbert matrix and the Pascal matrix, compute a Schur complement, take a submatrix of its converse, and then compute the SVD of the resulting matrix highly accurately, all in $\mathcal{O}(\max(m^3, n^3))$ time. In contrast, on examples similar to this one, the conventional algorithms may fail to compute even the largest singular value accurately (see section 7).

As an application of Theorem 1.2, in section 6 we derive a new algorithm for computing the bidiagonal decomposition of a TN generalized Vandermonde matrix based on removing appropriate columns of an ordinary Vandermonde matrix. This is a major improvement over previous such algorithms in [8, 34].

In the design of our algorithms we take the following approach.

First, we identify the source of large relative errors in conventional matrix algorithms. Relative accuracy in these algorithms is lost due to *subtractive cancellation* in the subtraction of approximate same-sign quantities. Conversely, the relative accuracy is preserved in multiplication, division, addition, and taking of square roots.

Second, we perform any and all transformations listed in Proposition 1.1 as a combination of the following *elementary elimination transformations* (EETs):

EET1: Subtracting a multiple of a row (column) from the next in order to create a zero in such a way that the transformed matrix is still TN;

EET2: Adding a multiple of a row (column) to the previous one;

EET3: Adding a multiple of a row (column) to the next one;

EET4: Scaling by a positive diagonal matrix.

Each of these EETs preserves the total nonnegativity [19].

Third, instead of applying an EET directly on a TN matrix A, we carry it out *implicitly* by transforming the entries of its bidiagonal decomposition, and arrange the computations in such a way that subtractions are not required. Thus the accuracy is preserved.

This paper is organized as follows. In section 2 we review the bidiagonal decompositions of TN matrices. In section 3 we review algorithms for accurate computations with TN matrices, given their bidiagonal decompositions. In section 4 we review algorithms from [27] for performing EET1 and EET2 and present new algorithms for performing EET3 and EET4. In section 5 we present algorithms for computing accurate bidiagonal decompositions of derivative TN matrices, obtained as described in Proposition 1.1. We present our new algorithm for computing the bidiagonal decomposition of a generalized Vandermonde matrix in section 6. In section 7 we present numerical results demonstrating the accuracy of our algorithms. We draw conclusions and present open problems in section 8.

Note on notation. Throughout this paper we use MATLAB [32] notation for vectors and submatrices.

2. Bidiagonal decompositions of TN matrices.

The TN matrices possess a very elegant structure, which is not revealed by their entries. Additionally, small relative perturbations in the entries of a TN matrix A can cause enormous relative perturbations in the small eigenvalues, singular values, and entries of A^{-1} [27, section 1]. Thus the matrix entries are ill suited as parameters in numerical computations with TN matrices.

Instead, following [27], we choose to represent a TN matrix as a product of nonnegative bidiagonal matrices. This representation arises naturally in the process of *Neville elimination*, which we now review, following [19] (see also [35]).

In the process of Neville elimination a matrix is reduced to upper triangular form using only *adjacent* rows. A zero is introduced in position $(m, 1)$ by subtracting a multiple $b_{m1} = a_{m1}/a_{m-1,1}$ of row $m-1$ from row m. Subtracting the multiple $b_{m-1,1} = a_{m-1,1}/a_{m-2,1}$ of row $m-2$ from row $m-1$ creates a zero in position $(m-1, 1)$, and so on. The total nonnegativity is preserved during Neville elimination [19], and therefore all multipliers b_{ij} are nonnegative.

This yields the decomposition

$$A = \left(\prod_{k=1}^{m-1} \prod_{j=m-k+1}^{m} E_j(b_{j,k+j-m}) \right) \cdot U,$$

where U is $m \times n$ upper triangular and

$$E_j(x) \equiv \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & x & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

is $m \times m$ and differs from the identity only in the $(j, j-1)$ entry.

Applying the same process to A^T, we obtain the decomposition

$$A = \left(\prod_{k=1}^{m-1} \prod_{j=m-k+1}^{m} E_j(b_{j,k+j-m}) \right) \cdot D \cdot \left(\prod_{1}^{k=n-1} \prod_{n-k+1}^{j=n} E_j^T(b_{k+j-n,j}) \right), \tag{2.1}$$

where D is a diagonal $m \times n$ matrix and E_j^T are $n \times n$. In the notation of (2.1) and throughout this paper, $\prod_1^{k=n-1}$ indicates that the product is taken for k from $n-1$ down to 1. Although somewhat nonstandard, this notation allows us to preserve the symmetry in (2.1).

The matrices

$$L^{(k)} \equiv \prod_{j=m-k+1}^{m} E_j(b_{j,k+j-m}) = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & b_{k,m-k+1} & 1 & & & \\ & & b_{k+1,m-k+2} & 1 & & \\ & & & \ddots & \ddots & \\ & & & & b_{m,m-1} & 1 \end{bmatrix}$$

and

$$U^{(k)} \equiv \prod_{n-k+1}^{j=n} E_j^T(b_{k+j-n,j}) = \begin{bmatrix} 1 & & & & & \\ & \ddots & b_{n-k+1,k} & & & \\ & & 1 & b_{n-k+2,k+1} & & \\ & & & 1 & \ddots & \\ & & & & \ddots & b_{n-1,n} \\ & & & & & 1 \end{bmatrix}$$

are $m \times m$ lower- and $n \times n$ upper bidiagonal, respectively. The decomposition (2.1) now becomes

$$A = L^{(1)} \cdots L^{(n-1)} \cdot D \cdot U^{(n-1)} \cdots U^{(1)}.$$

We denote the off-diagonal entries in $L^{(k)}$ and $U^{(k)}$ as

$$l_i^{(k)} \equiv L_{i+1,i}^{(k)} = b_{i+1,k+i+1-m} \quad \text{and} \quad u_i^{(k)} \equiv U_{i,i+1}^{(k)} = b_{k+i+1-n,i+1}. \tag{2.2}$$

We will use either $l_i^{(k)}$ or $b_{i+1,k+i+1-m}$ to denote the nontrivial entries of $L^{(k)}$ (and similarly with $U^{(k)}$). In different contexts one notation may be more convenient than the other, so we will keep (2.2) in mind when switching back and forth.

We now present the fundamental structure theorem for TN matrices.

THEOREM 2.1 (Gasca and Peña [19]). *An $m \times n$ matrix A is TN if and only if it can be uniquely factored as*

$$A = L^{(1)} \cdots L^{(m-1)} \cdot D \cdot U^{(n-1)} \cdots U^{(1)}, \tag{2.3}$$

where D is an $m \times n$ diagonal matrix with diagonal entries $d_i, i = 1, 2, \ldots, \min(n, m)$; $L^{(k)}$ are $U^{(k)}$ are $m \times m$ unit lower $n \times n$ unit upper bidiagonal matrices, respectively, such that the following hold:

1. *$d_i > 0$ for $i = 1, 2, \ldots, \min(m, n)$;*
2. *$l_i^{(k)} = 0,\ i < m - k$; $u_i^{(k)} = 0,\ i < n - k$; and $l_i^{(k)} = u_i^{(k)} = 0,\ i > m + n - k$;*
3. *$l_i^{(k)} \geq 0,\ m - k \leq i \leq m + n - k$, and $u_i^{(k)} \geq 0,\ n - k \leq i \leq m + n - k$;*
4. *$l_i^{(k)} = 0$ implies $l_{i+1}^{(k-1)} = 0$; $u_i^{(k)} = 0$ implies $u_{i+1}^{(k-1)} = 0$.*

We will refer to Theorem 2.1 to verify whether a particular decomposition of a TN matrix A as a product of bidiagonal matrices is in fact its unique bidiagonal decomposition.

Following [27], we denote the bidiagonal decomposition (2.3) of a TN matrix A as $\mathcal{BD}(A)$. We store the nontrivial entries of $\mathcal{BD}(A)$ compactly in an $m \times n$ array, which we also refer to as $\mathcal{BD}(A)$:

$$(\mathcal{BD}(A))_{ij} = \begin{cases} l_{i-1}^{(n-i+j)}, & i > j, \\ u_{j-1}^{(n-j+i)}, & i < j, \\ d_i, & i = j. \end{cases}$$

The (i, j)th entry in $\mathcal{BD}(A)$ equals the multiplier (b_{ij}) used to set the (i, j)th entry in A to zero (when $i \neq j$), or the ith entry on the diagonal of D (when $i = j$).

For example,

$$\begin{bmatrix} 2 & 6 \\ 8 & 29 \\ 48 & 209 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ & 6 & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ 4 & 1 & \\ & 7 & 1 \end{bmatrix} \begin{bmatrix} 2 & \\ & 5 \\ & \end{bmatrix} \begin{bmatrix} 1 & 3 \\ & 1 \end{bmatrix}$$

is stored as

$$\mathcal{BD}\left(\begin{bmatrix} 2 & 6 \\ 8 & 29 \\ 48 & 209 \end{bmatrix}\right) = \left\{\begin{matrix} 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{matrix}\right\}.$$

This notation is convenient since we can formally transpose $\mathcal{BD}(A)$ to obtain $\mathcal{BD}(A^T) = (\mathcal{BD}(A))^T$ [27, section 4].

In the language of the $m \times n$ array $B = \mathcal{BD}(A)$, conditions 1–4 in Theorem 2.1 are equivalent to the following:

1. $b_{ii} > 0, i = 1, 2, \ldots, \min(m, n)$;
2. $b_{ij} = 0$, unless $1 \leq i \leq m$ and $1 \leq j \leq n$;
3. $b_{ij} \geq 0,\ i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n$;
4. $b_{ij} = 0$ implies $b_{i+1,j} = 0$ if $i < j$, and $b_{i,j+1} = 0$ if $i > j$.

If the TN matrix A is also *totally positive* (i.e., if all its minors are positive), then the entries $b_{ij}, i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n$, are products and quotients of minors of A [27, section 3], [16]:

$$\begin{aligned} b_{ij} &= \frac{\det A(i-j+1 : i, 1 : j)}{\det A(i-j+1 : i-1, 1 : j-1)} \cdot \frac{\det A(i-j : i-2, 1 : j-1)}{\det A(i-j : i-1, 1 : j)}, \quad i > j, \\ b_{ii} &= \frac{\det A(1 : i, 1 : i)}{\det A(1 : i-1, 1 : i-1)}; \\ b_{ji} &= \frac{\det A(1 : j, i-j+1 : i)}{\det A(1 : j-1, i-j+1 : i-1)} \cdot \frac{\det A(1 : j-1, i-j : i-2)}{\det A(1 : j, i-j : i-1)}, \quad i > j. \end{aligned} \tag{2.4}$$

One can use the formulas (2.4) to compute explicit formulas for the bidiagonal decompositions of Vandermonde [10, 23, 33], [27, section 3], Cauchy [4], [27, section 3], Cauchy–Vandermonde [29, 30, 31], generalized Vandermonde [8], and Bernstein–Vandermonde [28] matrices.

Neville elimination is just one of eight analogous, but slightly different methods to eliminate a TN matrix A using only adjacent rows and columns [19, section 4]. Each method yields a decomposition of A as a product of nonnegative bidiagonal matrices with analogous but different nonzero patterns. In section 5.1 we show how to obtain an accurate $\mathcal{BD}(A)$ (and then perform accurate computations with A) starting with *any* decomposition of A as a product of nonnegative bidiagonal matrices. Therefore the particular choice of elimination pattern in Neville elimination and the resulting nonzero pattern in the factors of the decomposition (2.3) do not result in any loss of generality.

3. Performing accurate matrix computations given $\boldsymbol{\mathcal{BD}(A)}$.

The entries of $\mathcal{BD}(A)$ determine accurately the entries of the inverse, the entries of the LDU decomposition, and the values of any minor, eigenvalue, or singular value. Furthermore, given $\mathcal{BD}(A)$, many matrix computations with A can be performed accurately and efficiently. We review these results below.

3.1. Computing the inverse.

If A is a square $n\times n$ TN matrix, we can compute its inverse accurately by inverting (2.1):

$$A^{-1} = \left(\prod_{i=1}^{n-1}\prod_{j=n-i+1}^{n} E_j^T(-b_{i+j-n,j})\right)\cdot D^{-1}\cdot\left(\prod_{1}^{i=n-1}\prod_{n-i+1}^{j=n} E_j(-b_{j,i+j-n})\right). \tag{3.1}$$

Using (3.1) and the Cauchy–Binet identity [13, Vol. 1, p. 9], we conclude that each entry of A^{-1} is a linear function in each entry b_{ij} of $\mathcal{BD}(A)$ with either nonnegative or nonpositive coefficients. Therefore small relative perturbations in the b_{ij} cause small relative perturbations in any entry of A^{-1}. In other words, $\mathcal{BD}(A)$ determines every entry of A^{-1} accurately.

We can form A^{-1} by multiplying out (3.1) in $\mathcal{O}(n^3)$ time. Each entry of A^{-1} will be computed accurately, since the multiplication (3.1) involves no subtractive cancellation. (All matrices in (3.1), their partial products, and A^{-1} have checkerboard sign patterns.)

3.2. Solving $\boldsymbol{Ax = b}$.

We can use (3.1) to compute the solution to $Ax = b$ in $\mathcal{O}(n^2)$ time by multiplying out the expression

$$x = A^{-1}b = \left(\prod_{i=1}^{n-1}\prod_{j=n-i+1}^{n} E_j^T(-b_{i+j-n,j})\right) D^{-1}\left(\prod_{1}^{i=n-1}\prod_{n-i+1}^{j=n} E_j(-b_{j,i+j-n})\right) b \tag{3.2}$$

right-to-left. The computed solution $\hat{x}$ has a small componentwise relative backward error [4]; i.e., a matrix $\hat{A}$ exists such that $\hat{A}\hat{x} = b$ and $|A-\hat{A}| \le \mathcal{O}(\epsilon)|A|$, where the inequality is meant componentwise.

If b has alternating sign pattern (i.e., sign $b_i = (-1)^i$ or sign $b_i = (-1)^{i-1}$), then (3.2) involves no subtractive cancellation, and each component of x is computed accurately [23].

This approach for solving $Ax = b$ is the basis of the so-called Björck–Pereyra-type methods for solving structured TN linear systems. Derived originally for Vandermonde linear systems [3], these methods received deserved attention because of their remarkable accuracy[1] [22]. Generalizations were later developed for Cauchy [4],

[1]In the scope of Newton interpolation with positive and increasing nodes (i.e., the conditions under which the corresponding Vandermonde matrix is TN), the accuracy observation dates back to 1963 and was made by Kahan and Farkas [24].

Cauchy–Vandermonde [29, 30, 31], generalized Vandermonde [8], and Bernstein–Vandermonde [28] matrices. Each of these methods is either explicitly or implicitly based on a decomposition of the corresponding A^{-1} as a product of simple bidiagonal matrices analogous to (3.1).

3.3. Computing a minor. The value of any minor of a TN matrix A is determined accurately by $\mathcal{BD}(A)$ [6, section 9]. It can be computed accurately and efficiently given $\mathcal{BD}(A)$—see section 5.8.

3.4. Computing the LDU decomposition. Let A be a square TN nonsingular $n \times n$ matrix. Define

$$L \equiv L^{(1)} \cdots L^{(n-1)} \quad \text{and} \quad U \equiv U^{(n-1)} \cdots U^{(1)}. \tag{3.3}$$

Now (2.3) implies that $A = LDU$ is the LDU decomposition of A. The Cauchy–Binet identity and (3.3) imply that $\mathcal{BD}(A)$ determines each entry of L, D, and U accurately. Multiplying out (3.3) involves no subtractions and yields every entry of L and U accurately. The decompositions $\mathcal{BD}(L)$ and $\mathcal{BD}(U)$ are given by (3.3).

3.5. Computing the eigenvalues and the SVD. In [27, section 7] we proved that $\mathcal{BD}(A)$ accurately determines the eigenvalues and the SVD of a TN matrix A. In the same paper we presented algorithms for computing the eigenvalues and the SVD of A accurately and efficiently, given $\mathcal{BD}(A)$. These algorithms implicitly reduce both the eigenvalue and SVD problems to the bidiagonal SVD problem using only EETs. The resulting bidiagonal SVD problem is then solved accurately using known means [7, 11].

4. Performing EETs accurately. Let the TN matrix C be obtained from the $m \times n$ TN matrix A by applying an EET to A. In this section we show how, given $\mathcal{BD}(A)$, the decomposition $\mathcal{BD}(C)$ can be computed without performing any subtractions.

In [27, section 4.1] we showed that EET1 is equivalent to simply setting an entry of $\mathcal{BD}(A)$ to zero; EET2 involved some "bulge chasing" in $\mathcal{BD}(A)$ [27, section 4.2] and cost at most $6(m + 2)$ operations.

Next, we show how to perform EET3 and EET4 accurately.

4.1. Adding a multiple of a row to the next one. Let A be TN and C be obtained from A by adding a multiple of row $i - 1$ of A to row i:

$$C = E_i(x)A, \quad x > 0.$$

In this section we show how to accurately compute $\mathcal{BD}(C)$, given x and $\mathcal{BD}(A)$.

The following lemma shows how to compute the bidiagonal decomposition of the product of two bidiagonal matrices. It is the main building block of Algorithm 4.2 later in this section.

LEMMA 4.1. *Let B and C be $n \times n$ unit lower bidiagonal matrices with offdiagonal entries $b_i \geq 0$ and $c_i \geq 0$, $i = 1, 2, \ldots, n - 1$, respectively, such that $b_i = 0$ whenever $c_{i-1} = 0$. Then there exist bidiagonal matrices B' and C' with off-diagonal entries $b'_i \geq 0$ and $c'_i \geq 0$, $i = 1, 2, \ldots, n - 1$, respectively, such that $B'C' = BC$ and $b'_1 = 0$. Furthermore one can compute b'_i and c'_i without performing any subtractions in not more than $4n$ arithmetic operations.*

Proof. We compare the entries on both sides of $B'C' = BC$,

$$\begin{bmatrix} 1 & & & & \\ 0 & 1 & & & \\ & b_2' & 1 & & \\ & & \ddots & \ddots & \\ & & & b_{n-1}' & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ c_1' & 1 & & & \\ & c_2' & 1 & & \\ & & \ddots & \ddots & \\ & & & c_{n-1}' & 1 \end{bmatrix} \tag{4.1}$$
$$= \begin{bmatrix} 1 & & & & \\ b_1 & 1 & & & \\ & b_2 & 1 & & \\ & & \ddots & \ddots & \\ & & & b_{n-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ c_1 & 1 & & & \\ & c_2 & 1 & & \\ & & \ddots & \ddots & \\ & & & c_{n-1} & 1 \end{bmatrix},$$

to obtain $c_1' = b_1 + c_1$,

$$\begin{aligned} b_i' &= \frac{b_i c_{i-1}}{c_{i-1}'}, \\ c_i' &= b_i + c_i - b_i', \end{aligned} \tag{4.2}$$

for $i = 2, 3, \ldots, \min\{j | b_j = 0\}$, and $b_i' = b_i, c_i' = c_i$ otherwise. The subtraction in (4.2) can be eliminated by introducing auxiliary variables $d_i \equiv b_i - b_i'$. Then $d_1 = b_1 - b_1' = b_1$ and

$$\begin{aligned} d_i &= b_i - b_i' \\ &= b_i - \frac{b_i c_{i-1}}{c_{i-1}'} \\ &= \frac{b_i}{c_{i-1}'}(c_{i-1}' - c_{i-1}) \\ &= \frac{b_i}{c_{i-1}'}(b_{i-1} - b_{i-1}') \\ &= \frac{b_i d_{i-1}}{c_{i-1}'}. \end{aligned} \tag{4.3}$$

The subtraction-free (and therefore accurate) version of (4.2) is

$$\begin{aligned} b_i' &= \frac{b_i c_{i-1}}{c_{i-1}'}, \\ d_i &= \frac{b_i d_{i-1}}{c_{i-1}'}, \\ c_i' &= c_i + d_i. \end{aligned}$$

This computation clearly costs not more than $4n$ arithmetic operations. Since $c_i' = 0$ implies $b_{i+1}' = 0$, the product $B'C'$ is $\mathcal{BD}(BC)$. $\square$

We implement the procedure from Lemma 4.1 in Algorithm 4.1 below. We overwrite d_i by d_{i+1}, and the arrays b and c by b' and c', respectively. The quantity $e = b_{i+1}/c_i'$ is computed only once and used to update both b_{i+1} and d_{i+1}, thus saving one division.

Algorithm 4.1. *The following subtraction-free algorithm implements the procedure from Lemma* 4.1. *It returns the index i where the recurrence* (4.4) *is terminated.*

```
function $[b, c, i] = \mathtt{dqd2}(b, c)$
$t = c_1$
$c_1 = b_1 + c_1$
$d = b_1$
$b_1 = 0$
$i = 1$
while $(i < \mathtt{length}(b))$ and $(b_{i+1} > 0)$
    $e = b_{i+1}/c_i$
    $d = ed$
    $b_{i+1} = et$
    $t = c_{i+1}$
    $c_{i+1} = c_{i+1} + d$
    $i = i + 1$
end
```

Note: The only way the product BC differs from $\mathcal{BD}(BC)$ is in that $b_1 \neq 0$. The purpose of Algorithm 4.1 is to make b_1 zero without changing the product BC. No other zeros are introduced in B or C, and no nonzeros are introduced in B. At most one nonzero may be introduced in C. Algorithm 4.1 returns the index i where this may have happened ($c_i = 0$ on input, $c_i > 0$ on output). Although such an introduction of a nonzero in C causes no problems in the scope of Lemma 4.1, it may require additional work and bulge chasing in Algorithm 4.2 below, which uses Algorithm 4.1 as an intermediate step.

Theorem 4.2. *Let A be an $m \times n$ TN matrix. Given $x > 0$ and $\mathcal{BD}(A)$, the decomposition $\mathcal{BD}(E_i(x)A)$ can be computed without performing any subtractions in at most $4m$ arithmetic operations.*

Proof. Let $\mathcal{BD}(A)$ be given as in (2.3),

$$A = L^{(1)}L^{(2)}\cdots L^{(m-1)}DU^{(n-1)}U^{(n-2)}\cdots U^{(1)},$$

and let $L \equiv L^{(1)}L^{(2)}\cdots L^{(m-1)}$. The TN matrix $E_i(x)L$ is TN unit lower triangular. The idea is to compute the decomposition $\mathcal{BD}(E_i(x)\cdot L)$:

$$E_i(x)L = \mathcal{L}^{(1)}\mathcal{L}^{(2)}\cdots\mathcal{L}^{(m-1)}.$$

Then $\mathcal{BD}(E_i(x)A)$ is

$$E_i(x)A = \mathcal{L}^{(1)}\mathcal{L}^{(2)}\cdots\mathcal{L}^{(m-1)}DU^{(n-1)}U^{(n-2)}\cdots U^{(1)}.$$

We use Lemma 4.1 and Algorithm 4.1 to "chase the bulge" $E_i(x)$:

$$\begin{aligned} E_i(x)L &= E_i(x)L^{(1)}L^{(2)}\cdots L^{(m-1)} \\ &= \mathcal{L}^{(1)}E_{i_1}(x_1)L^{(2)}\cdots L^{(m-1)} \\ &= \mathcal{L}^{(1)}\mathcal{L}^{(2)}E_{i_2}(x_2)\cdots L^{(m-1)} \\ &= \dots \\ &= \mathcal{L}^{(1)}\mathcal{L}^{(2)}\cdots\mathcal{L}^{(m-1)}. \end{aligned}$$

We start with $k = 1$ and repeat the following process. We apply Algorithm 4.1 to the trailing principal submatrices of $E_i(x)$ and $L^{(k)}$ consisting of rows and columns i though n. The only nonzero in $E_i(x)$ disappears, and we obtain a new matrix $\bar{L}^{(k)} = E_i(x)L^{(k)}$.

If one of these three condition holds:

1. $k = m - 1$, or
2. no nonzeros were introduced in $\bar{L}^{(k)}$ that were not in $L^{(k)}$, or
3. a nonzero $\bar{l}_j^{(k)}$ was introduced in $\bar{L}^{(k)}$, but $l_{j-1}^{(k+1)} \neq 0$,

then we set $\mathcal{L}^{(k)} \equiv \bar{L}^{(k)}$; the "bulge chasing" is thus over, and we are done.

Otherwise (a nonzero $\bar{l}_j^{(k)}$ was introduced in $\bar{L}^{(k)}$, and $l_{j-1}^{(k+1)} = 0$, $k < m-1$), we have $\bar{L}^{(k)} = \mathcal{L}^{(k)} \cdot E_j(\bar{l}_j^{(k)})$, where $\mathcal{L}^{(k)}$ has the same nonzero pattern as $L^{(k)}$. We set $i = j$, $x = l_j^{(k)}$, increase k by one, and repeat the same process.

The computation of $\mathcal{BD}(E_i(x)A)$ is subtraction-free. At most $2n-3$ entries in $\mathcal{BD}(A)$ are changed at not more than two arithmetic operations per entry (see Algorithm 4.1). The total cost therefore does not exceed $4n$. ∎

The following algorithm implements the procedure from Theorem 4.2.

ALGORITHM 4.2. *Let A be an $m \times n$ TN matrix and $B = \mathcal{BD}(A)$. The following subtraction-free algorithm computes $\mathcal{BD}(E_i(x)A)$ in at most $4n$ time. For simplicity we assume that $b_{jl} = 0$ for $j \notin \{1, 2, \ldots, m\}$ or $l \notin \{1, 2, \ldots, n\}$.*

```
function $B = $ TNAddToNext$(B, x, i)$
$[m, n] = $ size$(B)$
$z = 0$
$b_{i0} = x$
while $(z < \min(i-1, n))$ and $(b_{i-1,z} = 0)$
    for $j = 1 : m - i + 1$
        $[c_j, d_j] = b_{j+i+1, z+j-1:z+j}$
    end
    $[c, d, q] = $ dqd2$(c, d)$
    for $j = 1 : m - i + 1$
        $b_{j+i+1, z+j-1:z+j} = [c_j, d_j]$
    end
    $i = i + q - 1$
    $z = z + q$
end
```

4.2. Multiplication by a diagonal matrix. The product of a diagonal matrix $F = \text{diag}(f_1, \ldots, f_m)$, $f_i > 0, i = 1, 2, \ldots, m$, and an $m \times n$ TN matrix A is TN. We now show how to compute $\mathcal{BD}(FA)$, given F and $\mathcal{BD}(A)$.

We propagate F through the factors $L^{(k)}$ in $\mathcal{BD}(A)$ using

$$\begin{bmatrix} f_1 & & & \\ & f_2 & & \\ & & \ddots & \\ & & & f_m \end{bmatrix} \begin{bmatrix} 1 & & & \\ c_1 & 1 & & \\ & \ddots & \ddots & \\ & & c_{m-1} & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ b_1 & 1 & & \\ & \ddots & \ddots & \\ & & b_{m-1} & 1 \end{bmatrix} \begin{bmatrix} f_1 & & & \\ & f_2 & & \\ & & \ddots & \\ & & & f_m \end{bmatrix},$$

where $b_i = c_i f_{i+1} / f_i$, $i = 1, 2, \ldots, m-1$.

ALGORITHM 4.3. *Given $B = BD(A)$ and the vector $(f_1, f_2, \ldots, f_m)$, the following algorithm computes $\mathcal{BD}(\mathrm{diag}(f_1, f_2, \ldots, f_m) \cdot A)$ using only multiplications and divisions in at most $2mn$ time.*

```
function $B = \texttt{TNDiagonalScale}(f, B)$
$[m, n] = \texttt{size}(B)$
$b_{11} = b_{11} f_1$
for $i = 2 : m$
    if $i \leq n$
        $b_{ii} = b_{ii} f_i$
    end
    $b_{i,1:\min(i-1,n)} = b_{i,1:\min(i-1,n)} \cdot f_i / f_{i-1}$
end
```

5. The bidiagonal decomposition of derivative TN matrices. Let A be a TN matrix obtained from other TN matrices using one of the operations listed in Proposition 1.1 that preserve the total nonnegativity.

In this section we present accurate and efficient subtraction-free algorithms for computing $\mathcal{BD}(A)$, given the corresponding bidiagonal decompositions of the input TN matrices.

MATLAB implementation of all algorithms for performing accurate computations with TN matrices presented in this paper and [27] are available online from [26].

5.1. A product of EETs. Let the TN matrix A be given as

$$A = F^{(1)} F^{(2)} \cdots F^{(k)},$$

where $F^{(i)}$ represents an EET; namely, it equals either $E_j(x)$, $E_j^T(x)$, or a positive diagonal matrix. Then $\mathcal{BD}(A)$ can be accurately accumulated using Algorithms 4.2, 4.3, as well as Proposition 4.1 and Algorithm 4.1 from [27].

We will use this approach throughout this section. Say we want to compute $\mathcal{BD}(A)$, where the TN matrix A is obtained from other TN matrices using operations that preserve the total nonnegativity. We will represent A as a product of EETs, which we will then accumulate.

Since any nonnegative bidiagonal matrix is a product of EETs, *any* representation A as a product of nonnegative bidiagonal matrices is a good starting point for performing accurate computations with A. Given any such representation, we can accumulate $\mathcal{BD}(A)$ without loss of accuracy.

5.2. The product of TN matrices. Let F and C be $m \times n$ and $n \times p$ TN matrices such that $m \leq n$ or $n \geq p$. Their product FC is a TN matrix. If $B = \mathcal{BD}(C)$, then from (2.1) we have

$$C = \left(\prod_{i=1}^{n-1} \prod_{j=n-i+1}^{n} E_j(b_{j,i+j-n}) \right) \cdot D \cdot \left(\prod_{1}^{i=p-1} \prod_{p-i+1}^{j=p} E_j^T(b_{i+j-p,j}) \right).$$

Therefore, forming the product FC is equivalent to applying a number of EETs to F.

ALGORITHM 5.1 (product). *Let F and C be $m \times n$ and $n \times p$ TN matrices, respectively, where $m \leq n$ or $n \geq p$. Given $A = \mathcal{BD}(F)$ and $B = \mathcal{BD}(C)$, the following subtraction-free algorithm computes $\mathcal{BD}(FC)$ in $\mathcal{O}(mnp)$ time:*

```
function $A = \texttt{TNProduct}(A,B)$
$[m,n] = \texttt{size}(A)$
$p = \texttt{size}(B,2)$
for $i = 1 : n-1$
    for $j = n-i+1 : \min(n, n+p-i)$
        $A = \texttt{TNAddToPrevious}(A, b_{j,i+j-n}, 1, j)$
    end
end
$A = A(:, 1 : \min(n,p))$
$A = \texttt{TNDiagonalScale}(\mathrm{diag}(B), A^T)^T$
for $i = p-1 : -1 : 1$
    for $j = p : -1 : p-i+1$
        $A = \texttt{TNAddToNext}(A^T, b_{i+j-p,j}, j)^T$
    end
end
```

The function $\texttt{TNAddToPrevious}(A,x,1,i)$ "adds" a multiple x of column i to column $i-1$ and costs at most $6(m+2)$ [27, Algorithm 4.1].

5.3. The re-signed inverse. Let A be an $n \times n$ TN matrix, and let J be a diagonal matrix of alternating 1's and -1's ($J_{ii} = (-1)^{i-1}$, $i = 1, 2, \ldots, n$). The *re-signed inverse* of A,

$$A^* \equiv \left[(-1)^{i+j} a_{ij}\right]^{-1} = (JAJ)^{-1} = JA^{-1}J,$$

is also TN [14]. Using $J^2 = I$, $(E_i(x))^{-1} = E_i(-x)$, $JE_i(-x)J = E_i(x)$, and (2.1),

$$\begin{aligned} A^* &= J \cdot \left(\prod_{i=1}^{n-1} \prod_{j=n-i+1}^{n} E_j^T(-b_{i+j-n,j}) \right) \cdot D^{-1} \cdot \left(\prod_{1}^{i=n-1} \prod_{n-i+1}^{j=n} E_j(-b_{j,i+j-n}) \right) \cdot J \\ &= \left(\prod_{i=1}^{n-1} \prod_{j=n-i+1}^{n} JE_j^T(-b_{i+j-n,j})J \right) \cdot JD^{-1}J \cdot \left(\prod_{1}^{i=n-1} \prod_{n-i+1}^{j=n} JE_j(-b_{j,i+j-n})J \right) \\ &= \left(\prod_{i=1}^{n-1} \prod_{j=n-i+1}^{n} E_j^T(b_{i+j-n,j}) \right) \cdot D^{-1} \cdot \left(\prod_{1}^{i=n-1} \prod_{n-i+1}^{j=n} E_j(b_{j,i+j-n}) \right). \end{aligned}$$

ALGORITHM 5.2 (re-signed inverse). *Let A be a square $n \times n$ TN matrix. Given $B = \mathcal{BD}(A)$, the following subtraction-free algorithm computes $C = \mathcal{BD}(A^*)$ in $\mathcal{O}(n^3)$ time:*

```
function $C = \texttt{TNRSInverse}(B)$
$n = \texttt{size}(B,1)$
$C = I$
for $i = 1 : n-1$
    for $j = n-i+1 : n$
        $C = \texttt{TNAddToNext}(C, b_{j,i+j-n}, j)$
    end
end
$C = \texttt{TNDiagonalScale}((1/b_{11}, \ldots, 1/b_{nn}), C)$
for $i = n-1 : -1 : 1$
    for $j = n : -1 : n-i+1$
        $C = \texttt{TNAddToPrevious}(C^T, b_{i+j-n,j}, 1, j)^T$
    end
end
```

5.4. The converse. If the $m \times n$ matrix $A = [a_{ij}]_{i,j=1}^{m,n}$ is TN, then so is its *converse* [19]

$$A^{\#} \equiv [a_{m+1-i,n+1-j}]_{i,j=1}^{m,n}.$$

Let $B = \mathcal{BD}(A)$, and let $Y_k \equiv [\delta_{k+1-i,j}]_{i,j=1}^{k}$ be the *reverse identity* of size k. Using $Y_k^2 = I$ and $E_i^{\#}(x) = Y_k E_i(x) Y_k = E_{k+2-i}^T(x)$, we obtain

$$\begin{aligned} A^{\#} &= Y_m A Y_n \\ &= Y_m \left(\prod_{i=1}^{m-1} \prod_{j=m-i+1}^{m} E_j(b_{j,i+j-m}) \right) D \left(\prod_{1}^{i=n-1} \prod_{n-i+1}^{j=n} E_j^T(b_{i+j-n,j}) \right) Y_n \\ &= \left(\prod_{i=1}^{m-1} \prod_{j=m-i+1}^{m} Y_m E_j(b_{j,i+j-m}) Y_m \right) Y_m D Y_n \left(\prod_{1}^{i=n-1} \prod_{n-i+1}^{j=n} Y_n E_j^T(b_{i+j-n,j}) Y_n \right) \\ &= \left(\prod_{i=1}^{m-1} \prod_{j=m-i+1}^{m} E_{m+2-j}^T(b_{j,i+j-m}) \right) D^{\#} \left(\prod_{1}^{i=n-1} \prod_{n-i+1}^{j=n} E_{n+2-j}(b_{i+j-n,j}) \right), \end{aligned} \tag{5.1}$$

where $D^{\#} = Y_m D Y_n$ is an $m \times n$ diagonal matrix, $D_{ii}^{\#} = b_{k+1-i,k+1-i}$, $k = \min(m,n)$, $i = 1, 2, \ldots, k$. We compute $\mathcal{BD}(A^{\#})$ as the bidiagonal decomposition of the product of all EETs in (5.1).

ALGORITHM 5.3 (converse). *Given $B = \mathcal{BD}(A)$ of an $m \times n$ TN matrix A, the following subtraction-free algorithm computes $\mathcal{BD}(A^{\#})$ in $\mathcal{O}(mn^2)$ time:*

```
function $C$ = TNConverse($B$)
$[m, n]$ = size($B$)
$C$ = eye($m, n$)
for $i = 1 : m - 1$
    for $j = m - i + 1 : m$
        $C$ = TNAddToNext($A^T, b_{j,i+j-m}, m + 2 - j$)$^T$
    end
end
$e$ = diag($B$)
$C$ = TNDiagonalScale($e(\min(m, n) : -1 : 1), C^T$)$^T$
for $i = n - 1 : -1 : 1$
    for $j = n : -1 : n - i + 1$
        $A$ = TNAddToPrevious($A, b_{i+j-n,j}, 1, n + 2 - j$)
    end
end
```

5.5. QR decomposition. Let A be TN, and let $A = QR$ be its QR decomposition such that R has a positive diagonal. Then R is TN and can be obtained by applying Givens rotations to A. Each Givens rotation preserves the TN structure of A and equals the product of three EETs [27, section 4.3].

ALGORITHM 5.4 (QR decomposition). *Let A be an $m \times n$ TN matrix, and let $A = QR$ be a QR decomposition of A such that $r_{ii} > 0$, $i = 1, 2, \ldots, \min(m, n)$. Given $B = \mathcal{BD}(A)$, the following subtraction-free algorithm computes $\mathcal{BD}(R)$ in $\mathcal{O}(mn^2)$ time:*

```
function $B = \texttt{TNQR}(B)$
$[m,n] = \texttt{size}(B)$
for $i = 1:n$
    for $j = m:-1:i+1$
        $x = b_{ji}$
        $b_{ji} = 0$
        $c = \sqrt{1+x^2}$
        $B = (\texttt{TNAddToPrevious}(B^T, x/c, c, j))^T$
    end
end
```

5.6. QR iteration. Gladwell showed in [20] that if A is TN and symmetric, then one step of QR iteration without pivoting (provided R has a positive diagonal) preserves the TN structure. We will now show how to compute the result of this iteration accurately using algorithms we already have.

Let A be TN and symmetric, and let $A = LDU = QR$ be its LDU and QR decompositions, respectively, with R having a positive diagonal.[2] Let $Q = LD_1U_1$ be the LDU decomposition of Q (Q and A share the L factor).

Let $F = RQ$ be the result of one step of QR iteration performed on A. Then $F = RQ = RLD_1U_1$. Since F is symmetric, it suffices to compute the lower bidiagonal factors and the diagonal factor of $\mathcal{BD}(F)$. Since U_1 is unit upper triangular, it thus suffices to compute $\mathcal{BD}(RLD_1)$. Since the factors are TN, this task is easy. We first use `TNQR` to obtain $\mathcal{BD}(R)$ and then `TNProduct` to obtain $\mathcal{BD}(RLD_1)$. We obtain D_1 by comparing the diagonals of the upper triangular matrices $DU = D_1U_1R$.

5.7. The Schur complement. Let A be an $m \times n$ TN matrix, and let A' be obtained from A after one step of Gaussian elimination. We have $A' = KA$, where

$$
\begin{aligned}
K &= \begin{bmatrix} 1 & & & & \\ -\frac{a_{21}}{a_{11}} & 1 & & & \\ -\frac{a_{32}}{a_{11}} & & 1 & & \\ \vdots & & & \ddots & \\ -\frac{a_{m1}}{a_{11}} & & & & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & \frac{a_{m1}}{a_{m-1,1}} & 1 \end{bmatrix}
\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & \frac{a_{m-1,1}}{a_{m-2,1}} & 1 & \\ & & & & 1 \end{bmatrix}
\cdots
\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & \frac{a_{31}}{a_{21}} & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \\
&\quad \times \begin{bmatrix} 1 & & & & \\ -\frac{a_{21}}{a_{11}} & 1 & & & \\ & -\frac{a_{31}}{a_{21}} & 1 & & \\ & & \ddots & \ddots & \\ & & & -\frac{a_{m1}}{a_{m-1,1}} & 1 \end{bmatrix} \\
&= \prod_{3}^{i=m} E_i(b_{i1}) \times \prod_{i=2}^{m} E_i(-b_{i1})
\end{aligned}
$$

[2]Technically, since A is symmetric, $U = L^T$, but this is unimportant here.

is a product of EETs. Forming the product

$$\left(\prod_{i=2}^{m} E_i(-b_{i1})\right) \cdot A$$

is equivalent to using adjacent rows to zero out the first column of A. It is therefore equivalent to simply setting $b_{i1} = 0,\ i = 2, 3, \ldots, m$ [27, section 4.1].

The multiplications by $E_i(b_{i1}),\ i = m, \ldots, 3$, are performed using Algorithm 4.2.

ALGORITHM 5.5 (Schur complement). *Let A be an $m \times n$ TN matrix, and let A' be obtained from A after one step of Gaussian elimination. Given $B = \mathcal{BD}(A)$, the following subtraction-free algorithm computes $\mathcal{BD}(A')$ and costs $\mathcal{O}(mn)$:*

```
function $B$ = TNSchurComplement($B$)
$m$ = size($B, 1$)
$c = B(:, 1)$
$B(2 : m, 1) = 0$
for $i = 3 : m$
    $B$ = TNAddToNext($B, c_i, i$)
end
```

5.8. A submatrix. Any submatrix C of a TN matrix A is TN. In this section we show how to compute $\mathcal{BD}(C)$, given $\mathcal{BD}(A)$. It suffices to describe how to compute $\mathcal{BD}(C)$ when C is obtained by removing row i from A. We assume that C is TN.

Consider first the case $i = 1$, i.e., C is obtained by removing the first row of A:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \quad C = \begin{bmatrix} a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}.$$

Let

$$\mathcal{BD}(A) = \left\{ \begin{array}{cccc} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{array} \right\}, \quad \mathcal{BD}(C) = \left\{ \begin{array}{cccc} f_{21} & f_{22} & \dots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m1} & f_{m2} & \dots & f_{mn} \end{array} \right\}.$$

Let A' be obtained from A by using adjacent columns to zero out the first row of A above the main diagonal:

$$A' = A \cdot E_n^T(-b_{1n}) \cdot E_{n-1}^T(-b_{1,n-1}) \cdots E_2^T(-b_{12}). \tag{5.2}$$

Then

$$A' = \begin{bmatrix} a'_{11} & 0 & \dots & 0 \\ a'_{21} & a'_{22} & \dots & a'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a'_{m1} & a'_{m2} & \dots & a'_{mn} \end{bmatrix} \quad \text{and} \quad \mathcal{BD}(A') = \left\{ \begin{array}{cccc} b_{11} & 0 & \dots & 0 \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{array} \right\}.$$

Let C' be obtained by removing the first row of A'. From (5.2) we have

$$C' = C \cdot E_n^T(-b_{1n}) \cdot E_{n-1}^T(-b_{1,n-1}) \cdots E_2^T(-b_{12}),$$

which implies

$$C = C' \cdot E_2^T(b_{12}) \cdot E_3^T(b_{13}) \cdots E_n^T(b_{1n}). \tag{5.3}$$

Therefore, it suffices to obtain $\mathcal{BD}(C')$. (Then we will use Algorithm 4.2 to obtain $\mathcal{BD}(C)$ using (5.3).)

Let

$$C' = \begin{bmatrix} a'_{21} & a'_{22} & \dots & a'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a'_{m1} & a'_{m2} & \dots & a'_{mn} \end{bmatrix} \quad \text{and} \quad \mathcal{BD}(C') = \left\{ \begin{matrix} f'_{21} & f'_{22} & \dots & f'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f'_{m1} & f'_{m2} & \dots & f'_{mn} \end{matrix} \right\}.$$

Consider the process of Neville elimination applied to A' and C' to eliminate the entries a'_{jk}, $j \neq k, k+1$, and reduce A' and C' to lower and upper bidiagonal matrices $\bar{A}$ and $\bar{C}$, respectively. The same multipliers will be used in this elimination:

$$f'_{jk} = b_{jk} \quad \text{for} \quad j \neq k, k+1.$$

The matrix

$$\bar{C} = \begin{bmatrix} f'_{21} & & & \\ & f'_{32} & & \\ & & f'_{43} & \\ & & & \ddots \end{bmatrix} \begin{bmatrix} 1 & f'_{22} & & \\ & 1 & f'_{33} & \\ & & 1 & \\ & & & \ddots \end{bmatrix} \tag{5.4}$$

is obtained by removing the first row of

$$\bar{A} = \begin{bmatrix} 1 & & & \\ b_{21} & 1 & & \\ & b_{32} & 1 & \\ & & & \ddots \end{bmatrix} \begin{bmatrix} b_{11} & & & \\ & b_{22} & & \\ & & b_{33} & \\ & & & \ddots \end{bmatrix}. \tag{5.5}$$

By comparing entries in (5.4) and (5.5), we obtain

$$f'_{i+1,i} = b_{i+1,i} b_{ii} \quad \text{and} \quad f'_{i+1,i+1} = \frac{b_{i+1,i+1}}{b_{i+1,i} b_{ii}}.$$

We have obtained the entire $\mathcal{BD}(C')$.

Now consider the general case—we remove the ith row of A to obtain C. In the process of Neville elimination, the same multipliers will be used to eliminate the first row of C that were used to eliminate the first row of A.

We emulate the Neville elimination of the first *column* of C by eliminating the first column of A in a slightly different order. We use adjacent rows to eliminate the entries of the first column of A with the exception of $a_{i+1,1}$. We use row $i-1$ to eliminate $a_{i+1,1}$—the exact same row that would be used to eliminate $a_{i+1,1}$ in C using *adjacent* rows.

This Gaussian-type elimination of rows i and $i+1$ in A can be handled in the same way as in section 5.7. We represent the elimination of rows i and $i+1$ as a sequence of three EETs:

$$\begin{aligned}\begin{bmatrix} 1 & & \\ -\frac{a_i}{a_{i-1}} & 1 & \\ -\frac{a_{i+1}}{a_{i-1}} & & 1 \end{bmatrix} &= \begin{bmatrix} 1 & & \\ -b_{i1} & 1 & \\ -b_{i+1,1}b_{i1} & & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & & \\ & 1 & \\ & b_{i+1,1} & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ -b_{i1} & 1 & \\ & -b_{i+1,1} & 1 \end{bmatrix} \\ &= E_{i+1}(b_{i+1,1})E_i(-b_{i1})E_{i+1}(-b_{i+1,1}).\end{aligned}$$

We then proceed by induction. We eliminate the second row and the second column of A and so on until we have eliminated the first i rows and the first i columns of A. Now we are in familiar territory—we need to remove the first row of the trailing submatrix $A(i : m, i : n)$.

ALGORITHM 5.6 (submatrix). *Let A be an $m \times n$ TN matrix, and let C be obtained by removing the ith row of A. Given $\mathcal{BD}(A)$, the following subtraction-free algorithm computes $\mathcal{BD}(C)$ in $\mathcal{O}(n^2)$ time:*

```
function $B =$ TNSubmatrix$(B, i)$
$[m, n] =$ size$(B)$
if $i < m$
    for $j = 1 : \min(i-1, n)$
        $B(j+1 : m, j+1 : n) =$ TNAddToNext$(B(j+1 : m, j+1 : n), b_{i+1,j}, i-j+1)$
        $b_{i+1,j} = b_{i+1,j} b_{ij}$
    end
    for $j = \min(n, m) + (m > n) : -1 : i+1$
        $b_{j,j-1} = b_{j,j-1} b_{j-1,j-1}$
        if $j \le n$
            $b_{jj} = b_{jj}/b_{j,j-1}$
        end
    end
    for $j = i+1 : n$
        $B(i+1 : m, i : n) =$ TNAddToNext$(B(i+1 : m, i : n)^T, b_{ij}, j-i+1)^T$
    end
end
Remove the $i$th row of $B$
```

6. Generalized Vandermonde matrices. In this section we describe how to easily, accurately, and efficiently compute the bidiagonal decomposition of a TN generalized Vandermonde matrix

$$G \equiv \left[x_i^{j-1+\lambda_{n-j+1}}\right]_{i,j=1}^n$$

with $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \ge 0$, $\lambda_i \in \mathbb{Z}, i = 1, 2, \ldots, n$. The matrix G is well known to be TN when $0 < x_1 < x_2 < \cdots < x_n$ [14, p. 76]. The nodes x_i and the *partition* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ of $|\lambda| = \lambda_1 + \lambda_2 + \cdots + \lambda_n$ are typical parameters used to describe generalized Vandermonde matrices. When $\lambda = (0, \ldots, 0)$, G reduces to the ordinary Vandermonde matrix $V \equiv \left[x_i^{j-1}\right]_{i,j=1}^n$.

There have been a couple of attempts at deriving accurate algorithms for this class of matrices, and both have shortcomings.

In 1977 Van de Vel [34] proposed a subtraction-free algorithm for the LDU decomposition of G. While accuracy was clearly guaranteed, efficiency was not. Recently, motivated by this result and some theoretical arguments [6, section 9.1(2)], Demmel and the current author presented an accurate algorithm for computing $\mathcal{BD}(G)$ [8, 9]. While this algorithm is accurate and efficient (its complexity is bounded by $\mathcal{O}(n^2|\lambda|^{2+\rho}\lambda_1^{3+\rho})$, where ρ is tiny [8, (3.9)]), it requires extended precision arithmetic when computing the Schur function in the intermediate steps [9]. This is a drawback.

With the results of this paper we are finally able to put this issue to rest by presenting a new very simple algorithm for computing $\mathcal{BD}(G)$, which is accurate and efficient—it costs only $\mathcal{O}(n^2\lambda_1)$ (and is thus much more efficient than the algorithm in [8]) and does not require the use of extended precision arithmetic. Once we have $\mathcal{BD}(G)$, we can clearly perform virtually all linear algebra with G at a modest $\mathcal{O}(n^3)$ additional cost.

Our idea is very simple: Start with the rectangular (ordinary) TN Vandermonde matrix

$$F = \left[x_i^{j-1}\right]_{i=1,j=1}^{n,n-1+\lambda_1}.$$

The decomposition $\mathcal{BD}(F)$ is readily available in $\mathcal{O}(n(n+\lambda_1))$ time using the formulas in [27, section 3, (3.6)]. We can then use Algorithm 5.6 to remove the appropriate λ_1 columns of F (at the cost of $\mathcal{O}(n^2)$ per column) to obtain $\mathcal{BD}(G)$. The total cost is nicely bounded by $\mathcal{O}(n^2\lambda_1)$.

7. Numerical experiments.

The algorithms presented in this paper can be used to perform a variety of accurate computations with TN matrices. We performed many tests to confirm their correctness and accuracy. In this section we present two numerical examples which incorporate several techniques for computing with TN matrices and demonstrate the accuracy and significance of our new algorithms.

For our experiments we selected two well-known notoriously ill-conditioned TN matrices—Hilbert and Pascal:

$$H = \left[\frac{1}{i+j-1}\right]_{i,j=1}^{m,n} \quad \text{and} \quad P = \left[\binom{i+j}{i}\right]_{i,j=1}^{n,p}.$$

We selected $m = 20, n = 30$, and $p = 20$, yielding fairly ill-conditioned *rectangular* H and P: $\kappa(H) = 3.3 \cdot 10^{25}$ and $\kappa(P) = 1.2 \cdot 10^{20}$. Both experiments involved the product $T = HP$, which was also severely ill-conditioned: $\kappa(T) = 6 \cdot 10^{45}$.

In our first experiment, we computed the singular values of $T = HP$ using the MATLAB implementations of our accurate algorithms[3]

(7.1) `TNSingularValues(TNProduct(TNCauchyBD(1:m,0:n-1),ones(n,p)))`

and also via the conventional MATLAB call

(7.2) `svd(H*P).`

For verification, we formed H and P, computed their product T, and computed T's singular values in 70-digit decimal floating point arithmetic using the MATLAB function `vpa`. Since $\kappa(T) = 6 \cdot 10^{45}$, `vpa` returned the singular values of T with at least 16 correct decimal digits in each. The results of `vpa` agreed to at least 14 digits with the ones computed using (7.1), confirming the accuracy of our algorithms.

[3]`TNSingularValues` is Algorithm 6.1 from [27], `TNProduct` is Algorithm 5.1 (see section 5); `TNCauchyBD` computes the bidiagonal decomposition of H accurately using the formulas from [27, section 3]; the entries of $\mathcal{BD}(P)$ are all ones, i.e., $\mathcal{BD}(P)$ equals `ones(n,p)`.

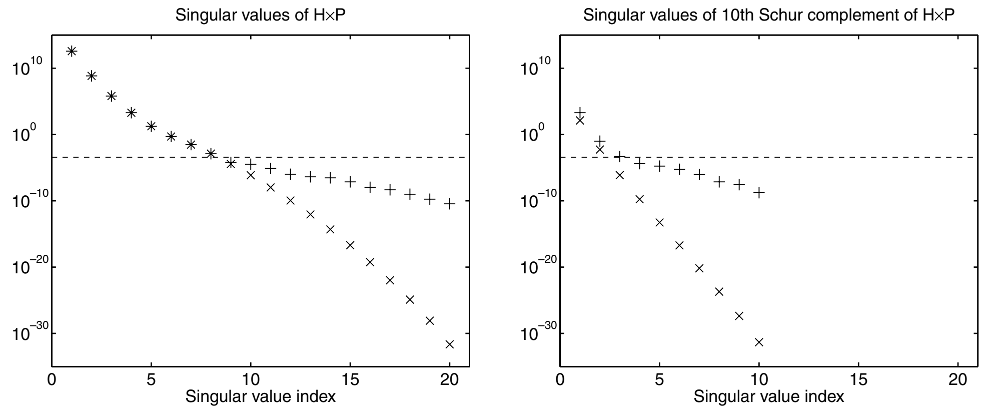

FIG. 7.1. *The singular values of the product T of the 20×30 Hilbert matrix H and the 30×20 Pascal matrix P (left plot) and the 10th Schur complement of T (right plot); "$\times$" = new, accurate algorithms, "$+$" = conventional. The dashed line represents the roundoff threshold, $\|T\| \cdot \varepsilon$.*

In contrast, the conventional singular value algorithms (7.2) in double precision [2] binary floating point arithmetic computed only the largest ones ($\sigma_i > \sigma_1\varepsilon = \|T\|\varepsilon$, where $\varepsilon \approx 10^{-16}$ is the machine precision) with any relative accuracy at all.

The results of this experiment are plotted in Figure 7.1, left.

In our second experiment, we computed the singular values of the 10th Schur complement of T using the same three methods—our new algorithms (in particular, `TNSchurComplement`, Algorithm 5.5) and a conventional MATLAB call, and finally verified the results in extended precision arithmetic. As expected, our new algorithms computed all singular values of the 10th Schur complement of T accurately, while the conventional MATLAB call failed to compute even a single singular value accurately (Figure 7.1, right).

Although this experiment is somewhat artificially contrived, it shows that very simple TN-preserving operations can result in a situation where the conventional matrix algorithms fail to deliver any accuracy at all.

8. Conclusions and open problems.

Using the intrinsic representation of TN matrices as a products of bidiagonal matrices allows for accurate computations with these matrices. The cost is similar to that of the conventional algorithms, but the computations are performed to high *relative* accuracy, as opposed to the high *absolute* accuracy of the conventional algorithms.

The singular (square) totally nonnegative matrices may not have a bidiagonal decomposition, or it may not be unique. Designing new algorithms (or adapting the ones in this paper) to perform accurate computations with these matrices is still an open problem.

The problem of finding algorithms for computing accurate eigenvectors of TN matrices is also open. In particular, such algorithms should guarantee the intrinsic properties of the eigenvector matrix—the jth computed eigenvector should have $j-1$ changes of sign in its entries, and the eigenvector matrix should have an LU decomposition such that L and U^{-1} are TN [14, 17].

The caveat in our algorithms is that every TN matrix must be represented by its bidiagonal decomposition. While every TN matrix intrinsically possesses such a decomposition, and for many classes of structured matrices this decomposition is very easy to obtain accurately (see section 2), there are important TN matrices for which we know of no accurate and efficient way to compute their bidiagonal decompositions. Two such examples are the following:

- the TN generalized Vandermonde matrix $\left[x_i^{y_j}\right]_{i,j=1}^n$, where $0 < x_1 < \cdots < x_n$, $0 < y_1 < y_2 < \cdots < y_n$, and at least one y_i is not an integer;
- the TN matrices appearing in the study of the hypergeometric function of a matrix argument [21].

Acknowledgments. I am deeply indebted to James Demmel for suggesting this problem and for providing motivation and insights during the preparation of this work. I also thank Froilán Dopico and Juan Manuel Peña for the careful reading of the manuscript and for pointing out inconsistencies in its earlier versions.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide, Third Edition*, Software Environ. Tools 9, SIAM, Philadelphia, 1999.

[2] ANSI/IEEE, *IEEE Standard for Binary Floating Point Arithmetic*, New York, Std 754-1985 ed., 1985.

[3] Å. Björck and V. Pereyra, *Solution of Vandermonde systems of equations*, Math. Comp., 24 (1970), pp. 893–903.

[4] T. Boros, T. Kailath, and V. Olshevsky, *A fast parallel Björck-Pereyra-type algorithm for solving Cauchy linear equations*, Linear Algebra Appl., 302/303 (1999), pp. 265–293.

[5] F. Brenti, *Combinatorics and total positivity*, J. Combin. Theory Ser. A, 71 (1995), pp. 175–218.

[6] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač, *Computing the singular value decomposition with high relative accuracy*, Linear Algebra Appl., 299 (1999), pp. 21–80.

[7] J. Demmel and W. Kahan, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 873–912.

[8] J. Demmel and P. Koev, *The accurate and efficient solution of a totally positive generalized Vandermonde linear system*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 142–152.

[9] J. Demmel and P. Koev, *Accurate and efficient evaluation of Schur and Jack functions*, Math. Comp., 75 (2006), pp. 223–239.

[10] S. M. Fallat, *Bidiagonal factorizations of totally nonnegative matrices*, Amer. Math. Monthly, 108 (2001), pp. 697–712.

[11] K. Fernando and B. Parlett, *Accurate singular values and differential qd algorithms*, Numer. Math., 67 (1994), pp. 191–229.

[12] S. Fomin and A. Zelevinsky, *Total positivity: Tests and parametrizations*, Math. Intelligencer, 22 (2000), pp. 23–33.

[13] F. Gantmacher, *The Theory of Matrices*, AMS Chelsea, Providence, RI, 1998.

[14] F. Gantmacher and M. Krein, *Oscillation Matrices and Kernels and Small Vibrations of Mechanical Systems*, revised ed., AMS Chelsea, Providence, RI, 2002.

[15] M. Gasca and C. A. Micchelli, eds., *Total Positivity and Its Applications*, Math. Appl. 359, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[16] M. Gasca and J. M. Peña, *Total positivity and Neville elimination*, Linear Algebra Appl., 165 (1992), pp. 25–44.

[17] M. Gasca and J. M. Peña, *Total positivity, QR factorization, and Neville elimination*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 1132–1140.

[18] M. Gasca and J. M. Peña, *Corner cutting algorithms and totally positive matrices*, in Curves and Surfaces in Geometric Design (Chamonix-Mont-Blanc, 1993), A. K. Peters, Wellesley, MA, 1994, pp. 177–184.

[19] M. Gasca and J. M. Peña, *On factorizations of totally positive matrices*, in Total Positivity and Its Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996, pp. 109–130.

[20] G. M. L. Gladwell, *Total positivity and the QR algorithm*, Linear Algebra Appl., 271 (1998), pp. 257–272.

[21] K. I. Gross and D. S. P. Richards, *Total positivity, spherical series, and hypergeometric functions of matrix argument*, J. Approx. Theory, 59 (1989), pp. 224–246.

[22] N. J. Higham, *Error analysis of the Björck-Pereyra algorithms for solving Vandermonde systems*, Numer. Math., 50 (1987), pp. 613–632.

[23] N. J. Higham, *Stability analysis of algorithms for solving confluent Vandermonde-like systems*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 23–41.

[24] W. Kahan and I. Farkas, *Algorithms* 167-169, Comm. ACM, 6 (1963), pp. 164–165. See also the *Certification*, Comm. ACM, 6(9):523.

[25] S. Karlin, *Total Positivity. Vol.* I, Stanford University Press, Stanford, CA, 1968.

[26] P. Koev, *Algorithms for totally nonnegative matrices*, http://www-math.mit.edu/~plamen.

[27] P. Koev, *Accurate eigenvalues and SVDs of totally nonnegative matrices*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 1–23.

[28] A. Marco and J.-J. Martinez, *A fast and accurate algorithm for solving Bernstein–Vandermonde linear systems*, Linear Algebra Appl., 422 (2007), pp. 616–628.

[29] J. J. Martínez and J. M. Peña, *Factorizations of Cauchy-Vandermonde matrices*, Linear Algebra Appl., 284 (1998), pp. 229–237.

[30] J. J. Martínez and J. M. Peña, *Fast algorithms of Björck-Pereyra type for solving Cauchy-Vandermonde linear systems*, Appl. Numer. Math., 26 (1998), pp. 343–352.

[31] J. J. Martínez and J. M. Peña, *Factorizations of Cauchy-Vandermonde matrices with one multiple pole*, in Recent Research on Pure and Applied Algebra, Nova Science Publishers, Hauppauge, NY, 2003, pp. 85–95.

[32] The MathWorks, Inc., *MATLAB Reference Guide*, Natick, MA, 1992.

[33] H. Oruç and G. M. Phillips, *Explicit factorization of the Vandermonde matrix*, Linear Algebra Appl., 315 (2000), pp. 113–123.

[34] H. Van de Vel, *Numerical treatment of a generalized Vandermonde system of equations*, Linear Algebra Appl., 17 (1977), pp. 149–179.

[35] A. M. Whitney, *A reduction theorem for totally positive matrices*, J. Anal. Math., 2 (1952), pp. 88–92.