SIAM J. MATRIX ANAL. APPL.
Vol. 27, No. 1, pp. 142–152

THE ACCURATE AND EFFICIENT SOLUTION OF A TOTALLY POSITIVE GENERALIZED VANDERMONDE LINEAR SYSTEM*

JAMES DEMMEL† AND PLAMEN KOEV‡

Abstract. Vandermonde, Cauchy, and Cauchy–Vandermonde totally positive linear systems can be solved extremely accurately in $O(n^2)$ time using Björck–Pereyra-type methods. We prove that Björck–Pereyra-type methods exist not only for the above linear systems but also for *any* totally positive linear system as long as the initial minors (i.e., contiguous minors that include the first row or column) can be computed accurately. Using this result we design a new $O(n^2)$ Björck–Pereyra-type method for solving generalized Vandermonde systems of equations by using a new algorithm for computing the Schur function. We present explicit formulas for the entries of the bidiagonal decomposition, the LDU decomposition, and the inverse of a totally positive generalized Vandermonde matrix, as well as algorithms for computing these entries to high relative accuracy.

Key words. high relative accuracy, generalized Vandermonde matrix, total positivity, bidiagonal decomposition, Schur function

AMS subject classifications. 65F05, 15A23

DOI. 10.1137/S0895479804440335

1. Introduction.

We consider the class of *totally positive* (TP) matrices, i.e., matrices with all minors positive, and in particular the TP *generalized Vandermonde matrices* defined as

$$G = \begin{bmatrix} x_1^{a_1} & x_1^{a_2} & \dots & x_1^{a_n} \\ x_2^{a_1} & x_2^{a_2} & \dots & x_2^{a_n} \\ & & \ddots & \\ x_n^{a_1} & x_n^{a_2} & \dots & x_n^{a_n} \end{bmatrix}, \tag{1.1}$$

where $0 \le a_1 < a_2 < \cdots < a_n$ are integers and $0 < x_1 < x_2 < \cdots < x_n$ [12, p. 76].

We define the *partition* λ associated with G as the nonincreasing sequence of nonnegative integers

$$\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n) = (a_n - (n-1), a_{n-1} - (n-2), \dots, a_1).$$

In terms of λ_i,

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

Throughout this paper G will denote this particular TP generalized Vandermonde matrix and $V = [x_i^{j-1}]_{1 \le i,j \le n}$ will denote the ordinary Vandermonde matrix. The latter corresponds to partition $\lambda = (0)$.

Our goal is to perform accurate and efficient matrix computations with G—we consider bidiagonal decomposition, linear equation solving, triangular decomposition, and inversion. Our inspiration comes from other algorithms for Vandermonde,

*Received by the editors January 29, 2004; accepted for publication (in revised form) by H. J. Woerdeman November 19, 2004; published electronically July 13, 2005.
http://www.siam.org/journals/simax/27-1/44033.html

†Computer Science Division and Mathematics Department, University of California, Berkeley, CA 94720 (demmel@cs.berkeley.edu).

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

Cauchy, and Cauchy–Vandermonde matrices [4, 18, 25] which use a method introduced by Björck and Pereyra [3] to

1. solve $n \times n$ linear systems $Ax = b$ in $O(n^2)$ time, and
2. be arbitrarily more accurate than the usual condition number $\kappa(A) \equiv \|A\|_2 \cdot \|A^{-1}\|_2$ would suggest.

Such accurate computation is justified, because in fact the *structured* condition number of these matrices, i.e., the sensitivity of the solution with respect to the parameters defining the matrices, can be arbitrarily smaller than $\kappa(A)$. So these accurate computations do in fact compute the answer as accurately as the data deserve.

We extend these earlier results as follows.

First, we observe that accurate and efficient Björck–Pereyra-type methods exist for *any* TP matrix that permits accurate and efficient evaluation of its *initial* minors. Initial minors are minors of contiguous sets of rows and columns and include at least the first row and/or the first column [16]. This is based on an earlier result of Gasca and Peña: the entries of the bidiagonal factors of a matrix inverse arising from *Neville elimination* are products and quotients of initial minors and so can be computed accurately and efficiently if these minors can be [16, 25].

Next, we must consider the initial minors of generalized Vandermonde matrices. It is well known that

$$\det G = s_\lambda(x_1, x_2, \ldots, x_n) \cdot \det V = s_\lambda(x_1, x_2, \ldots, x_n) \cdot \prod_{i>j}(x_i - x_j) \tag{1.2}$$

is the product of the *Schur function* $s_\lambda(x_1, x_2, \ldots, x_n)$ and $\det V = \prod_{i>j}(x_i - x_j)$ [23]. The other minors of G have analogous forms.

Our second and most important contribution is the application of our new fast and accurate algorithm for evaluating Schur functions [7] to accurately compute all the needed minors of G in order to obtain

- an accurate bidiagonal decomposition of G^{-1} (section 3);
- a new Björck–Pereyra-type algorithm for solving $Gy = b$ (section 4);
- an accurate G^{-1} (section 5);
- an accurate LDU decomposition of G (section 6).

These computations can be performed in time that grows like a low order polynomial in n and $|\lambda| \equiv \sum_i \lambda_i$.

The accuracy of our algorithms is due to the fact that they can suffer strictly bounded or no subtractive cancellation (depending on the choice of algorithm from [7] for computing the Schur function accurately—see section 3). This means all rounding errors accumulate slowly, and the final computed values are quite accurate, yielding very small error bounds (section 7).

In contrast, previous algorithms for generalized Vandermonde matrices may be arbitrarily less accurate than ours, or may require arbitrarily high precision arithmetic to achieve the same accuracy. We compare the accuracy of our new Björck–Pereyra-type algorithm to that of the traditional algorithms in section 8.

In what follows we will use MATLAB [26] notation for vectors and minors of a matrix. For example, the submatrix of A consisting of rows i through j and columns k through l will be denoted as $A(i:j, k:l)$ or, equivalently, $A_{i:j,k:l}$. The submatrix of A consisting of rows i through j and columns $k, k+1, \ldots, l$ and s will be denoted as $A(i:j, [k:l, s])$. Similarly, $x_{1:n}$ will mean the vector $(x_1, x_2, \ldots, x_n)$, etc.

2. Related work. Björck and Pereyra proposed an $O(n^2)$ method for solving a Vandermonde linear system $Vy = b$ in their landmark paper [3]. Their idea was to

use Newton interpolation, which written in matrix form is simply a decomposition

$$V^{-1} = U^{(1)}U^{(2)}\dots U^{(n-1)}L^{(n-1)}L^{(n-2)}\dots L^{(1)} \tag{2.1}$$

of V^{-1} as a product of upper ($U^{(i)}$) and lower ($L^{(i)}$) bidiagonal matrices. Björck and Pereyra noticed that their method sometimes produced solutions with surprising accuracy despite the ill conditioning of V. Higham [17, 18] showed that when the nodes x_i are positive and sorted (i.e., when V is TP) no subtractive cancellation occurs in forming the solution if b has an alternating sign pattern.[1] This is a consequence of the fact that in floating point arithmetic the relative accuracy is preserved when multiplying, dividing, or adding quantities with the same sign, or forming $x_i \pm x_j$, where x_i and x_j are exact initial data. In fact all entries of $L^{(i)}$ and $U^{(i)}$ are either 0, 1, $-x_i$, or $1/(x_i - x_j)$, $i > j$ [18].

Björck–Pereyra-type methods were also derived for Cauchy [4] and Cauchy–Vandermonde matrices [25].

Explicit formulas for the entries of the LU decomposition of an ordinary Vandermonde matrix were obtained in [28] and for the entries of the LU decomposition and the inverse of a generalized Vandermonde matrix in [29].

The total positivity of a matrix A is not revealed by its entries—one would need to verify that the n^2 initial minors are positive to conclude that all 4^n minors are and thus A is TP [10]. Yet to design structure-exploiting algorithms, we first look for a structure-revealing representation. It turns out that the TP structure is revealed by the entries of the bidiagonal decomposition of A (or equivalently the bidiagonal decomposition of A^{-1}). This bidiagonal decomposition is obtained from a process called *Neville elimination* [13, 15, 16], which reduces A to upper triangular form using only *adjacent* rows for elimination (unlike Gaussian elimination, which uses a fixed pivot row). The entries of the bidiagonal factors in this decomposition intrinsically parameterize the set of TP matrices [2, 8, 9, 10, 16]. They determine the entries of the inverse [17], the LU decomposition, the solution (if the right-hand side has an alternating sign pattern) [17], the eigenvalues, and the SVDs [22] to high relative accuracy.

The numerical solution of a generalized Vandermonde linear system was discussed in [29]. In that paper Van de Vel shows how to compute an LU decomposition of a TP generalized Vandermonde matrix G without suffering any subtractive cancellation, thus computing each entry of L and U to high relative accuracy. He says, "The Björck–Pereyra modification is essentially based on Newton's formula for polynomial interpolation, and hence is not applicable, since we work with a set of powers[2] $j_1, \dots, j_n$, which are not 'in line'." We demonstrate that Björck–Pereyra-type methods are in fact applicable to G. To derive those, one must use the bidiagonal decomposition of G^{-1} and not Newton's method. Since the Björck–Pereyra-type methods yield more accurate solutions than LU-based algorithms (see section 7), this paper represents a substantial advance over the results in [29].

3. Bidiagonal decomposition of G^{-1}. In this section we present explicit formulas for the decomposition of G^{-1} as the product of bidiagonal matrices and show how the entries of the latter can be computed efficiently to high relative accuracy. This is perhaps the most important decomposition of G from a numerical point of view:

[1] The preservation of the relative accuracy in the process of Newton interpolation with positive and increasing interpolation nodes was, however, first noticed by Kahan and Farkas in 1963 [19].

[2] We call these $a_1, \dots, a_n$ in (1.1).

- It reveals the total positivity of G [16].
- It yields a Björck–Pereyra-type algorithm for the solution to the TP generalized Vandermonde linear system $Gy = b$ (section 4).
- It allows for the computation of the eigenvalues and SVDs of G to high relative accuracy in $O(n^3)$ additional time [22].

We start with a result due to Gasca and Peña [13, Lemma 2.6(1)].

PROPOSITION 3.1. *If A is TP, then it can be uniquely factored as*

$$A^{-1} = U^{(1)}U^{(2)}\dots U^{(n-1)}D^{-1}L^{(n-1)}L^{(n-2)}\dots L^{(1)}, \tag{3.1}$$

where $L^{(k)}$ and $U^{(k)}$, $k = 1, 2, \dots, n$, are unit lower and upper bidiagonal matrices, respectively, such that $L^{(k)}_{i+1,i} = U^{(k)}_{i,i+1} = 0$ for $i < k$, and D is diagonal. Also

$$L^{(k)}_{i+1,i} = -\frac{\det A_{i-k+2:i+1,1:k}}{\det A_{i-k+2:i,1:k-1}} \cdot \frac{\det A_{i-k+1:i-1,1:k-1}}{\det A_{i-k+1:i,1:k}} \quad \textit{for} \quad i \geq k; \tag{3.2}$$

$$U^{(k)}_{i,i+1} = -\frac{\det A_{1:k,i-k+2:i+1}}{\det A_{1:k-1,i-k+2:i}} \cdot \frac{\det A_{1:k-1,i-k+1:i-1}}{\det A_{1:k,i-k+1:i}} \quad \textit{for} \quad i \geq k; \tag{3.3}$$

$$D_{ii} = \frac{\det A_{1:i,1:i}}{\det A_{1:i-1,1:i-1}}. \tag{3.4}$$

We now apply Proposition 3.1 to G. Let $i < j, k < l \in \{1, 2, \dots, n\}$ be such that $j - i = l - k$. Then from (1.2)

$$\begin{aligned}
\det(G_{i:j,k:l}) &= \det\bigl(x_s^{r-1+\lambda_{n-r+1}}\bigr)_{i\leq s\leq j, k\leq r\leq l} \\
&= \det\bigl(x_s^{r-1+\lambda_{n-r+1}}\bigr)_{i\leq s\leq j, 1\leq r\leq l-k+1} \cdot \prod_{s=i}^{j} x_s^{k-1} \\
&= \det(V_{i:j,1:l-k+1}) \cdot s_{(\lambda_{n-l+1},\lambda_{n-l+2},\dots,\lambda_{n-k+1})}(x_{i:j}) \cdot \prod_{s=i}^{j} x_s^{k-1}.
\end{aligned}$$

Now for the bidiagonal decomposition (3.1) of G^{-1} we have, from (3.2), (3.3), and (3.4),

$$\begin{aligned}
L^{(k)}_{i+1,i} &= -\frac{\det(V_{i-k+2:i+1,1:k}) \cdot s_{(\lambda_{n-k+1:n})}(x_{i-k+2:i+1})}{\det(V_{i-k+2:i,1:k-1}) \cdot s_{(\lambda_{n-k+2:n})}(x_{i-k+2:i})} \\
&\quad \times \frac{\det(V_{i-k+1:i-1,1:k-1}) \cdot s_{(\lambda_{n-k+2:n})}(x_{i-k+1:i-1})}{\det(V_{i-k+1:i,1:k}) \cdot s_{(\lambda_{n-k+1:n})}(x_{i-k+1:i})} \\
&= -\frac{s_{(\lambda_{n-k+1:n})}(x_{i-k+2:i+1})}{s_{(\lambda_{n-k+2:n})}(x_{i-k+2:i})} \cdot \frac{s_{(\lambda_{n-k+2:n})}(x_{i-k+1:i-1})}{s_{(\lambda_{n-k+1:n})}(x_{i-k+1:i})} \\
&\quad \times \prod_{j=i-k+1}^{i-1} \frac{x_{i+1} - x_{j+1}}{x_i - x_j},
\end{aligned} \tag{3.5}$$

$$D_{ii}^{-1} = \frac{\det(G_{1:i-1,1:i-1})}{\det(G_{1:i,1:i})} = \frac{s_{(\lambda_{n-i+2:n})}(x_{1:i-1})}{s_{(\lambda_{n-i+1:n})}(x_{1:i})} \cdot \prod_{j=1}^{i-1} \frac{1}{x_i - x_j}, \tag{3.6}$$

$$
\begin{aligned}
U^{(k)}_{i,i+1} &= -\frac{\det(V_{1:k,1:k})\cdot s_{(\lambda_{n-i:n-i+k-1})}(x_{1:k})\prod_{s=1}^{k} x_s^{i-k+1}}{\det(V_{1:k-1,1:k-1})\cdot s_{(\lambda_{n-i+1:n-i+k-1})}(x_{1:k-1})\prod_{s=1}^{k-1} x_s^{i-k+1}} \\
&\quad\times \frac{\det(V_{1:k-1,1:k-1})\cdot s_{(\lambda_{n-i+2:n-i+k})}(x_{1:k-1})\prod_{s=1}^{k-1} x_s^{i-k}}{\det(V_{1:k,1:k})\cdot s_{(\lambda_{n-i+1:n-i+k})}(x_{1:k})\prod_{s=1}^{k} x_s^{i-k}} \\
&= -x_k\cdot\frac{s_{(\lambda_{n-i:n-i+k-1})}(x_{1:k})}{s_{(\lambda_{n-i+1:n-i+k-1})}(x_{1:k-1})}\frac{s_{(\lambda_{n-i+2:n-i+k})}(x_{1:k-1})}{s_{(\lambda_{n-i+1:n-i+k})}(x_{1:k})}. \qquad (3.7)
\end{aligned}
$$

Example 3.1. Let $\lambda = (1)$ and $n = 4$. Then $s_{(1)}(x_1, x_2, \ldots, x_i) = x_1 + x_2 + \cdots + x_i$ and the decomposition (3.1) of G^{-1} is

$$
\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^4 \\ 1 & x_2 & x_2^2 & x_2^4 \\ 1 & x_3 & x_3^2 & x_3^4 \\ 1 & x_4 & x_4^2 & x_4^4 \end{bmatrix}^{-1}
$$

$$
= \begin{bmatrix} 1 & -x_1 & & \\ & 1 & -x_1 & \\ & & 1 & -x_1 \\ & & & 1 \end{bmatrix}\begin{bmatrix} 1 & & & \\ & 1 & -x_2 & \\ & & 1 & -x_2 \\ & & & 1 \end{bmatrix}\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & -x_3(x_1+x_2+x_3) \\ & & & 1 \end{bmatrix}
$$

$$
\times \operatorname{diag}\left(1, x_2 - x_1, (x_3 - x_2)(x_3 - x_1), (x_4 - x_3)(x_4 - x_2)(x_4 - x_1)\sum_{i=1}^{4} x_i\right)^{-1}
$$

$$
\times \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -\frac{(x_4-x_3)(x_4-x_2)}{(x_3-x_1)(x_3-x_2)} & 1 \end{bmatrix}\begin{bmatrix} 1 & & & \\ & 1 & & \\ & -\frac{x_3-x_2}{x_2-x_1} & 1 & \\ & & -\frac{x_4-x_3}{x_3-x_2} & 1 \end{bmatrix}\begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & -1 & 1 & \\ & & -1 & 1 \end{bmatrix}.
$$

The Schur function can be computed to high relative accuracy as described in [7]. The expressions (3.5)–(3.7) involve only multiplications, divisions, and/or forming $x_i - x_j$, and thus are computed to high relative accuracy.

ALGORITHM 3.1 (bidiagonal decomposition of G^{-1}). *Given a partition λ and a vector $x = (x_1, x_2, \ldots, x_n)$ the following algorithm computes the bidiagonal decomposition of G^{-1} to high relative accuracy. All Schur functions are computed to high relative accuracy using the techniques from* [7].

$d = \texttt{ones}(n)$
for $i = 2 : n$
  for $j = i - 1 : -1 : 1$
    $d_{ij} = d_{i,j+1} \cdot (x_i - x_j)$
for $k = 1 : n - 1$
  for $i = n - 1 : -1 : k$
    $L^{(k)}_{i+1,i} = -\frac{s_{(\lambda_{n-k+1:n})}(x_{i-k+2:i+1})}{s_{(\lambda_{n-k+2:n})}(x_{i-k+2:i})} \cdot \frac{s_{(\lambda_{n-k+2:n})}(x_{i-k+1:i-1})}{s_{(\lambda_{n-k+1:n})}(x_{i-k+1:i})} \cdot \frac{d_{i+1,i-k+2}}{d_{i,i-k+1}}$
for $i = 1 : n \quad D^{-1}_{ii} = \frac{s_{(\lambda_{n-i+2:n})}(x_{1:i-1})}{s_{(\lambda_{n-i+1:n})}(x_{1:i})} \cdot \frac{1}{d_{i1}}$
for $k = n - 1 : -1 : 1$
  for $i = k : n - 1$
    $U^{(k)}_{i,i+1} = -x_k \cdot \frac{s_{(\lambda_{n-i:n-i+k-1})}(x_{1:k})}{s_{(\lambda_{n-i+1:n-i+k-1})}(x_{1:k-1})} \cdot \frac{s_{(\lambda_{n-i+2:n-i+k})}(x_{1:k-1})}{s_{(\lambda_{n-i+1:n-i+k})}(x_{1:k})}$

Algorithm 3.1 implements (3.5)–(3.7) straightforwardly. For efficiency we precompute $d_{ij} = (x_i - x_{i-1})(x_i - x_{i-2}) \dots (x_i - x_j)$ for all $i > j$ at the cost of n^2 floating point operations.

Let ϵ denote machine precision. We now briefly discuss the two algorithms from [7] for computing the Schur function to high relative accuracy.

- The first algorithm evaluates the *dual Jacobi–Trudi determinant* [23, (3.5), p. 41]

$$s_\mu = \det\bigl(e_{\mu'_i - i + j}\bigr)_{1 \le i,j \le \mu_1} \tag{3.8}$$

  in $(|\mu|\mu_1 \log \mu_1 + \log \frac{1}{\epsilon})$-digit arithmetic, then rounds the result back to $\log \frac{1}{\epsilon}$ digits, i.e., to precision ϵ. Here e_k are the elementary symmetric polynomials and μ' is the conjugate partition ($\mu'_i = \#(\mu_j \ge i)$). The amount of subtractive cancellation in evaluating (3.8) is bounded, and by using just $|\mu|\mu_1 \log \mu$ extra digits, s_μ is computed to high relative accuracy. The cost of computing $s_\mu(x_{i:j})$ is $O(((j-i)|\mu| + \mu_1^3)(|\mu|\mu_1)^{1+\rho})$, where ρ is tiny and accounts for certain logarithmic functions. Of this cost $O((j-i)|\mu|(|\mu|\mu_1)^{1+\rho})$ is for computing the e_i and the rest is for computing the determinant in (3.8), all done in $(|\mu|\mu_1 \log \mu_1 + \log \frac{1}{\epsilon})$-digit arithmetic. The functions $e_k(x_{i:j})$, $1 \le k \le |\lambda|$, $1 \le i \le j \le n$, need only be computed once. Since all Schur functions $s_\mu(x_{i:j})$ in Algorithm 3.1 satisfy $j - i < n$ and $\mu \le \lambda$, the overall cost of Algorithm 3.1 is bounded by

$$O\bigl(n^2 + n^2|\lambda|(|\lambda|\lambda_1)^{1+\rho} + n^2\lambda_1^3(|\lambda|\lambda_1)^{1+\rho}\bigr) \le O\bigl(n^2|\lambda|^{2+\rho}\lambda_1^{3+\rho}\bigr). \tag{3.9}$$

- The second algorithm uses the recursive formula [23, (5.9), p. 72]

$$s_\lambda(x_{1:n}) = \sum_{\mu \le \lambda} s_\mu(x_{1:n-1}) \cdot x_n^{|\lambda| - |\mu|}, \tag{3.10}$$

  which involves only additions of positive quantities and multiplications, and is therefore always accurate. There is no need to use extra precision. In the process of computing $s_\lambda(x_{1:n})$ using (3.10) one ends up computing also $s_\nu(x_{1:m})$ for all $\nu \le \lambda$ and $m \le n$. In practice we compute $s_\lambda(x_{i:n})$ for $i = 1, 2, \dots, n-1$ and obtain *all* Schur functions that appear in (3.5)–(3.7) (see [21, Algorithm 4.4.1, p. 69] for an implementation). The complexity of Algorithm 3.1 can then be bounded by

$$O\left(n^2 e^{5.2|\lambda|^{1/2}}\right).$$

  This bound is asymptotically worse than (3.9), but this approach does not require the use of extended precision.

4. Björck–Pereyra-type method for solving $Gy = b$. We now present our main algorithm for solving a generalized Vandermonde linear system.

ALGORITHM 4.1 (Björck–Pereyra-type method for solving $Gy = b$). *Given the entries of the bidiagonal factors* (3.1) *of G, the solution to $Gy = b$ can be computed in $O(n^2)$ time by forming*

$$y = G^{-1}b = U^{(1)}U^{(2)} \dots U^{(n-1)}D^{-1}L^{(n-1)}L^{(n-2)} \dots L^{(1)}b$$

from right to left:

```
for k = 1 : n − 1
    for i = n − 1 : −1 : k
        b_{i+1} = b_{i+1} − b_i · L^{(k)}_{i+1,i}
for i = 1 : n   b_i = b_i · D^{−1}_{ii}
for k = n − 1 : −1 : 1
    for i = k : n − 1
        b_i = b_i − b_{i+1} · U^{(k)}_{i,i+1}
```

When b has an alternating sign pattern ($\operatorname{sign}(b_i) = (-1)^i$ or $\operatorname{sign}(b_i) = -(-1)^i$) then Algorithm 4.1 is subtraction-free, so that y is very accurate. Algorithm 4.1 is the natural generalization of the Björck–Pereyra algorithm [3] for ordinary Vandermonde matrices. A MATLAB implementation of Algorithm 4.1 is available from [20].

5. The inverse of G. In this section we present explicit expressions for the entries of G^{-1} and describe two methods for computing these entries to high relative accuracy.

The $(n-1) \times (n-1)$ minor of G, obtained by removing row k and column l, is a generalized Vandermonde determinant. We use Cramer's rule to obtain explicit formulas for the entries of G^{-1}:

$$\begin{aligned}
\left(G^{-1}\right)_{lk} &= (-1)^{k+l} \frac{\det\left(x_i^{j-1+\lambda_{n-j+1}}\right)_{1\le i,j\le n; i\ne k, j\ne l}}{\det G} \\
&= (-1)^{k+l} \frac{s_{\lambda^{(k,l)}}(x_1,\ldots,\hat{x}_k,\ldots,x_n)\cdot \det V(x_1,\ldots,\hat{x}_k,\ldots,x_n)}{s_\lambda(x_1,\ldots,x_n)\cdot \det V} \\
&= (-1)^{n-l} \frac{s_{\lambda^{(k,l)}}(x_1,\ldots,\hat{x}_k,\ldots,x_n)}{s_\lambda(x_1,\ldots,x_n)} \cdot \prod_{i=1, i\ne k}^{n} \frac{1}{x_k - x_i}, \qquad (5.1)
\end{aligned}$$

where $\lambda^{(k,l)} = (\lambda_1 + 1, \lambda_2 + 1, \ldots, \lambda_{n-l} + 1, \lambda_{n-l+2}, \ldots, \lambda_n)$. The "hats" indicate the omission of the symbols they cover. The formula (5.1) was also obtained in [29], where the Schur function is referred to as a "D-determinant."

Using (5.1) we can compute G^{-1} to high relative accuracy componentwise by employing the techniques from [7] for computing the Schur function. The last factor in (5.1), $\prod_{i=1,i\ne k}^{n} \frac{1}{x_k - x_i}$, is computed to high relative accuracy.

Alternatively, G^{-1} can be computed by multiplying out (3.1). Because of the checkerboard sign pattern of the factors in (3.1), no subtractive cancellation can occur in this multiplication, obtaining each entry of G^{-1} to high relative accuracy.

Both methods scale cubically with n and as a low order polynomial in $|\lambda|$.

6. The LDU decomposition of G. In this section we present explicit formulas for the entries of the LDU decomposition of G resulting from Gaussian elimination with no pivoting. We also present two methods for computing these entries to high relative accuracy.

Let $G = LDU$, where L and U are unit lower and upper triangular matrices, respectively. Then [11]

$$D_{ii} = \frac{\det G_{1:i,1:i}}{\det G_{1:i-1,1:i-1}}; \quad U_{ij} = \frac{\det G_{1:i,[1:i-1,j]}}{\det G_{1:i,1:i}}; \quad \text{and} \quad L_{ij} = \frac{\det G_{[1:j-1,i],1:j}}{\det G_{1:j,1:j}}.$$

All minors in the above expressions are generalized Vandermonde determinants; thus

$$\det G_{1:i,1:i} = (\det V_{1:i,1:i}) \cdot s_{(\lambda_{n-i+1},\ldots,\lambda_p)}(x_{1:i});$$
$$\det G_{[1:i-1,j],1:i} = (\det V_{[1:i-1,j],1:i}) \cdot s_{(\lambda_{n-i+1},\ldots,\lambda_p)}(x_{[1:i-1,j]});$$
$$\det G_{1:i,[1:i-1,j]} = (\det V_{1:i,1:i}) \cdot s_{(\lambda_{n-j+1}+j-i,\lambda_{n-i+2},\ldots,\lambda_p)}(x_{1:i}).$$

Therefore

$$D_{ii} = \frac{s_{(\lambda_{n-i+1},\ldots,\lambda_p)}(x_{1:i})}{s_{(\lambda_{n-i+2},\ldots,\lambda_p)}(x_{1:i-1})} \prod_{j=1}^{i-1}(x_i - x_j); \tag{6.1}$$

$$U_{ij} = \frac{s_{(\lambda_{n-j+1}+j-i,\lambda_{n-i+2},\ldots,\lambda_p)}(x_{1:i})}{s_{(\lambda_{n-i+1},\ldots,\lambda_p)}(x_{1:i})}, \quad i < j; \tag{6.2}$$

$$L_{ij} = \frac{s_{(\lambda_{n-j+1},\ldots,\lambda_p)}(x_{[1:j-1,i]})}{s_{(\lambda_{n-j+1},\ldots,\lambda_p)}(x_{1:j})} \prod_{k=1}^{j-1} \frac{x_i - x_k}{x_j - x_k}, \quad i > j. \tag{6.3}$$

The formulas (6.1)–(6.3) can be used to compute each entry of L, D, and U to high relative accuracy if the techniques from [7] are used for computing the Schur function. The total cost will be $O(n^2)$, where the constant hidden in the big-O notation grows like a low order polynomial in $|\lambda|$ and can be found in [7].

Alternatively, the entries of L, D, and U can be computed starting with (3.1). For instance,

$$L = (L^{(1)})^{-1}(L^{(2)})^{-1} \ldots (L^{(n-1)})^{-1}.$$

Since $L^{(i)}$ is bidiagonal, each multiplication by $(L^{(i)})^{-1}$ can be performed using only additions and multiplications of positive numbers in $O(n^2)$ time. The total cost of that approach is $O(n^3)$, in addition to computing (3.1), and yields each entry of L, D, and U to high relative accuracy.

The formulas (6.1)–(6.3) were also obtained in [29] along with a subtraction-free procedure for computing the entries of L, D, and U to high relative accuracy. No complexity analysis was performed in [29].

Our interest in the LDU decomposition of TP matrices stems from the fact that an accurate LDU decomposition of a TP matrix yields a solution to a TP linear system with small componentwise relative backward error [5]. Our Björck–Pereyra-type method from section 3 goes a step further by producing a solution that also has a small *forward* error when the right-hand side has an alternating sign pattern.

We consider only the LDU decomposition from Gaussian elimination with no pivoting, since pivoting is not needed to ensure stability when solving a TP linear system [14] (see also parts 3 and 4 of Proposition 7.2).

7. Perturbation theory and error analysis. We will now show that the entries of the bidiagonal decomposition (3.1) of G^{-1}, each entry of L, D, and U in the LDU decomposition of G, and each entry of G^{-1} may be very well conditioned functions of the data x_i despite the fact that $\kappa(G)$ may be very large.

THEOREM 7.1. *Let $\hat{x}_i = x_i(1+\delta_i)$, where $|\delta_i| < \delta < 1/|\lambda|$, be small perturbations of $x_i > 0$. Let z be any nonzero entry of $L^{(k)}$, D^{-1}, or $U^{(k)}$ in the bidiagonal decomposition of G^{-1}, any entry of G^{-1}, or any entry of L, D, or U in the LDU decomposition of G. Let $\hat{z}$ be the corresponding entry in the appropriate decomposition or inverse of $\hat{G}$, where $\hat{G}$ is obtained by replacing x_i by $\hat{x}_i$ in G. Let* $\mathrm{rel\_gap}_x$ *be the smallest relative gap between any pair of x_i's:*

$$\mathrm{rel\_gap}_x \equiv \min_{i \neq j} \frac{|x_i - x_j|}{|x_i| + |x_j|}.$$

Then

$$|\hat{z} - z| \leq \frac{(4|\lambda| + 2n/\text{rel\_gap}_x)\,\delta}{1 - (4|\lambda| + 2n/\text{rel\_gap}_x)\,\delta}|z|.$$

Proof. The Schur function $s_\mu(x_{i:j})$, $1 \leq i \leq j \leq n$, $\mu \leq \lambda$, is a polynomial of degree $|\mu| \leq |\lambda|$ with positive coefficients. Therefore [7]

$$|s_\mu(\hat{x}_{1:i}) - s_\mu(x_{1:i})| \leq \frac{\delta|\mu|}{1 - \delta|\mu|} s_\mu(x_{1:i}), \tag{7.1}$$

which, combined with formulas (3.5)–(3.7), (5.1), and (6.1)–(6.3), implies the desired result. ∎

Therefore the quantity

$$\kappa^{\text{struct}}(G) = 4|\lambda| + 2n/\text{rel\_gap}_x \tag{7.2}$$

is the correct *structured condition number* of G, which can be arbitrarily smaller than $\kappa(G)$.

We now turn our attention to the accuracy of the computed solution.

PROPOSITION 7.2 (see [1, 4, 18]). *Let the relative error in every entry of $L^{(i)}$, $U^{(i)}$, D^{-1} be bounded by $\xi \ll 1$, and let $\hat{y}$ be the solution of $Gy = b$ computed in floating point arithmetic with machine precision ϵ as explained in section 4. Then the following error bounds are valid:*

1. $|y - \hat{y}| \leq O(\epsilon + \xi)|G^{-1}||b|$.
2. *If b has an alternating sign pattern (i.e., $(-1)^i b_i$ have the same sign for all $i = 1, 2, \ldots, n$), then $\hat{y}$ is computed with a small componentwise relative forward error:*

$$|y - \hat{y}| \leq O(\epsilon + \xi)|y|.$$

3. *The componentwise residual is small:*

$$|b - G\hat{y}| \leq O(\epsilon + \xi)|G||\hat{y}| + O((\epsilon + \xi)^2).$$

4. *The componentwise relative backward error* [6, p. 37], [27] *is also small, i.e., $\hat{G}$ such that $\hat{G}\hat{y} = b$ can be taken to satisfy*

$$|G - \hat{G}| \leq O(\epsilon + \xi)|G| + O((\epsilon + \xi)^2).$$

A proof is available in [1, 4, 18]; we just mention some details. Since G is TP, the bidiagonal factors in (3.1) have positive diagonals and nonpositive offdiagonals. Thus the product of all bidiagonals may be formed, obtaining G^{-1} without suffering any subtractive cancellation, proving part 1. If the right-hand side has an alternating sign pattern, then the solution is also formed without subtractive cancellation, proving part 2. Parts 3 and 4 are equivalent [27] and follow from the fact that the inverse of a small relative componentwise perturbation of a bidiagonal matrix is a small relative componentwise perturbation of the inverse of the bidiagonal matrix.

8. Numerical experiments. We performed extensive numerical experiments and confirmed the correctness and cost of our algorithms. We selected two of those tests and report them below.

TABLE 8.1

Solving a 12×12 generalized Vandermonde linear system, $\lambda = (6, 3, 2, 1, 1)$. The digits that disagree with the results from Mathematica are underlined.

Nodes x_i	*Mathematica*	Algorithm 3.1	Traditional
2.0594766805425224	-1.055583330089385 0e12	-1.055583330089384 8e12	-1.0 7084960321674 12e12
2.0940348689727863	4.1382711534659341e12	4.1382711534659 16e12	4.1 9823427844711 91e12
2.1568348650743552	-7.1458326298047432e12	-7.1458326298047 402e12	-7.2 4957491512918 46e12
2.3449660139029209	7.073689709677744e12	7.073689709677 725e12	7.1 7658604767132 62e12
2.3472440864992130	-4.3450312586851128e12	-4.3450312586851128e12	-4.4 0836088884471 39e12
2.4163531061463885	1.6416125390612351e12	1.6416125390612 349e12	1.6 6558742744586 79e12
2.6563511105968578	-3.2574038855981763e11	-3.2574038855981 757e11	-3.3 0507323452876 65e11
2.7184145910794202	1.5677076891114130e10	1.5677076891114 128e10	1.5 9074447181127 51e10
2.7423051335090571	-2.9091221498536110e09	-2.9091221498536110e09	-2.9 5195966317788 03e09
2.8840219496751516	4.3989708280861162e07	4.3989708280861 54e07	4.4 6401996532131 06e07
2.9550986332751350	-5.8977191662033915e05	-5.8977191662033915e05	-5.9 8530192932528 68e05
2.9582142929074067	1.1347682330511120e02	1.1347682330511 122e02	1.1 5176415913470 63e02

We tested Algorithm 3.1 as follows. We chose a partition $\lambda = (6, 3, 2, 1, 1)$, $n = 12$, and uniformly distributed random nodes $x_i \in [2, 3]$, which we then sorted in increasing order, and a right-hand side

$$b = (-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1)^T$$

with alternating signs. This is equivalent to interpolating the data b with a polynomial of the form

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^8 + a_8 x^9 + a_9 x^{11} + a_{10} x^{13} + a_{11} x^{17}$$

at the nodes x_i, $i = 1, 2, \ldots, 12$. This resulted in a TP generalized Vandermonde matrix G that was quite ill conditioned: $\kappa(G) \approx 3.7 \cdot 10^{18}$. The choice of a right-hand side with alternating signs guarantees that the solution will be computed with no subtractive cancellation with Algorithm 3.1.

We computed the solution to $Gz = b$ using Algorithm 3.1 and also using 100-digit arithmetic in *Mathematica* [30] and got the same result to 14 digits, a loss of at most two digits of accuracy in each entry. Note that we generated the random numbers x_i in MATLAB, output them as decimal numbers to a file, read them into *Mathematica*, converted them to 100 decimal digit big floats, and then formed the explicit matrix which we then solved with. In other words, we more than likely made some errors in the 16th place just in forming the matrix in *Mathematica*. The smallest relative gap between the x_i's is $\text{rel\_gap}_x \approx 10^{-3}$ and so the structured condition number (7.2) is only $\kappa^{\text{struct}} \approx 10^4$. Thus the problem is well conditioned, and we could expect the solution computed by *Mathematica* and the one computed by Algorithm 3.1 to differ in at most four digits in the last place. The results in Table 8.1 confirmed our claims that the problem is well conditioned, even though $\kappa(G) \approx 3.7 \cdot 10^{18}$ and the traditional algorithm computed a solution that was only good in the first digit. Since Algorithm 3.1 computes the Schur function at every step, this test also confirms the correctness of the algorithm for computing the Schur function from [7].

Acknowledgments. The authors would like to thank Jose-Javier Martínez for pointing out to us the relevance of the paper [24] and especially Nick Higham for suggesting the reference [8] and for several helpful discussions in the preparation of this paper.

REFERENCES

[1] P. Alonso, M. Gasca, and J. M. Peña, *Backward error analysis of Neville elimination*, Appl. Numer. Math., 23 (1997), pp. 193–204.

[2] A. Berenstein, S. Fomin, and A. Zelevinsky, *Parametrizations of canonical bases and totally positive matrices*, Adv. Math., 122 (1996), pp. 49–149.

[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] C. de Boor and A. Pinkus, *Backward error analysis for totally positive linear systems*, Numer. Math., 27 (1977), pp. 485–490.

[6] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[7] J. Demmel and P. Koev, *Accurate and efficient evaluation of Schur and Jack functions*, Math. Comp., to appear.

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

[9] S. Fomin and A. Zelevinsky, *Double Bruhat cells and total positivity*, J. Amer. Math. Soc., 12 (1999), pp. 335–380.

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

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

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

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

[14] M. Gasca and J. M. Peña, *Scaled pivoting in Gauss and Neville elimination for totally positive systems*, Appl. Numer. Math., 13 (1993), pp. 345–355.

[15] M. Gasca and J. M. Peña, *A matricial description of Neville elimination with applications to total positivity*, Linear Algebra Appl., 202 (1994), pp. 33–53.

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

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

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

[19] W. Kahan and I. Farkas, *Algorithms* 167–169, Comm. ACM, 6 (1963), pp. 164–165; see also the *certification*, Comm. ACM, 6 (1963), p. 523.

[20] P. Koev, http://www-math.mit.edu/~plamen.

[21] P. Koev, *Accurate and Efficient Computations with Structured Matrices*, Ph.D. dissertation, University of California, Berkeley, CA, 2002.

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

[23] I. G. Macdonald, *Symmetric Functions and Hall Polynomials*, 2nd ed., Oxford University Press, New York, 1995.

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

[25] 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.

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

[27] W. Oettli and W. Prager, *Compatibility of approximate solution of linear equations with given error bounds for coefficients and right hand sides*, Numer. Math., 6 (1964), pp. 405–409.

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

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

[30] S. Wolfram, *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley, Redwood City, CA, 1988.