BIT 33 (1993), 473–484.

FAST INVERSION OF VANDERMONDE-LIKE MATRICES INVOLVING ORTHOGONAL POLYNOMIALS‡

D. CALVETTI* and L. REICHEL†

* *Department of Pure and Applied Mathematics, Stevens Institute of Technology, Hoboken, NJ 07030.*
† *Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242.*

Dedicated to Gene H. Golub on his 60th birthday

Abstract.

Let $\{q\}_{j=0}^{n-1}$ be a family of polynomials that satisfy a three-term recurrence relation and let $\{t_k\}_{k=1}^n$ be a set of distinct nodes. Define the Vandermonde-like matrix $W_n = [w_{jk}]_{k,j=1}^n$, $w_{jk} = q_{j-1}(t_k)$. We describe a fast algorithm for computing the elements of the inverse of W_n in $O(n^2)$ arithmetic operations. Our algorithm generalizes a scheme presented by Traub [22] for fast inversion of Vandermonde matrices. Numerical examples show that our scheme often yields higher accuracy than the LINPACK subroutine SGEDI for inverting a general matrix. SGEDI uses Gaussian elimination with partial pivoting and requires $O(n^3)$ arithmetic operations.

AMS(MOS) Subject classification: 65F05, 65D05, 65D30.

Key words: Vandermonde matrix, inverse, fast algorithm, Leja ordering.

1. Introduction.

Let $\{t_k\}_{k=1}^n$ be a set of real distinct nodes and let $\{q_j\}_{j=0}^{n-1}$ be a family of polynomials that satisfy a three-term recurrence relation

$$q_{j+1}(t) = \theta_j(t - \beta_j)q_j(t) - \gamma_j q_{j-1}(t), \qquad j = 0, 1, \ldots, n-2, \tag{1.1}$$

with

$$q_0(t) = \delta, \qquad q_{-1}(t) = 0, \tag{1.2}$$

where $\delta \neq 0$ and $\theta_j > 0$ for all j. Define the $n \times n$ Vandermonde-like matrix

‡ Research supported by NSF grant DMS-9002884.
Received January 1992. Revised December 1992.

$$W_n = \begin{bmatrix} q_0(t_1) & q_0(t_2) & \cdots & q_0(t_n) \\ q_1(t_1) & q_1(t_2) & \cdots & q_1(t_n) \\ \vdots & \vdots & & \vdots \\ q_{n-1}(t_1) & q_{n-1}(t_2) & \cdots & q_{n-1}(t_n) \end{bmatrix}. \tag{1.3}$$

The associated linear systems of equations $W_n \boldsymbol{a} = \boldsymbol{b}$ and $W_n^T \boldsymbol{a} = \boldsymbol{b}$, where $\boldsymbol{b}$ is a given right hand side vector, appear naturally and have to be solved in many applications, such as when approximating functionals or when approximating functions by polynomials or rational functions; see [1, 22]. If $q_j(t) = t^j$, then the matrix W_n reduces to a Vandermonde matrix.

Many representations of the inverse of a Vandermonde matrix are available. A representation of the elements in terms of symmetric functions is described in [18]. Algorithms for fast solution of linear systems of equations with Vandermonde matrices or their transpose are presented in [3, 6, 12, 13, 21]. These algorithms determine various representations of the inverse of a Vandermonde matrix, e.g., as products of triangular matrices or as a sum of such products. Traub [22] describes an algorithm for rapidly computing the elements of the inverse of a Vandermonde matrix.

Representations of Vandermonde-like matrices W_n in factored form are described by Higham [15, 16] and Verde-Star [23]. In [15, 16] algorithms for the factorization of Vandermonde-like matrices are described, and [23] presents a factorization that can be used to develop an algorithm for computing the entries of the inverse of a Vandermonde-like matrix.

Vandermonde matrices with real nodes t_k are often quite ill-conditioned; see Gautschi and Inglese [10, 11]. On the other hand, Vandermonde-like matrices, in which the polynomials q_j are chosen to be orthogonal polynomials for an interval $[a, b]$ containing the nodes, can be much better conditioned; see Gautschi [9]. This has generated interest in developing fast solution methods for linear systems of equations with Vandermonde-like matrices (1.3); see [5, 15, 16, 20].

The present paper presents a new fast algorithm for computing the elements of the inverse of an $n \times n$ Vandermonde-like matrix W_n. Our scheme requires only $O(n^2)$ arithmetic operations. If the structure of W_n is ignored and W_n^{-1} is computed by first factoring W_n by Gaussian elimination with partial pivoting and thereafter solving $W_n X_n = I$ by using the computed factorization, then $O(n^3)$ arithmetic operations are required. Moreover, our fast algorithm generally yields higher accuracy than the slower scheme based on Gaussian elimination. Our algorithm is derived by generalizing the scheme for computing the inverse of a Vandermonde matrix described by Traub [22].

The exploitation of structure is a well-known theme in numerical linear algebra. However, fast solvers for linear systems of equations that use the structure of the matrix to reduce the operation count can yield less accurate approximate solutions than structure-ignoring slower solvers, such as Gaussian elimination with partial pivoting; see [4] for a discussion. Our interest in the new scheme for computing the

entries of W_n^{-1} stems from that it is faster than Gaussian elimination with partial pivoting and numerical experiments suggest that it also is more accurate.

We remark that Higham [14] demonstrated that the fast Vandermonde solver by Björck and Pereyra [3] also can be very accurate. More precisely, Higham [14] showed that if the signs of the entries of the right hand side vector alternate and if the nodes t_j are all of one sign, then the accuracy of the computed solution is independent of the condition number. Computed examples show that our algorithm for the computation of W_n^{-1} yields higher accuracy than Gaussian elimination with partial pivoting also when the nodes t_j are not all of the same sign.

It may be convenient to apply our scheme when many Vandermonde-like systems have to be solved with the same matrix and different right hand side vectors. After having determined W_n^{-1} the solution of each linear system of equation requires only one matrix-vector product. The matrix-vector products can be computed efficiently by using level 2 BLAS. If several right hand side vectors are known simultaneously, then several solutions can be determined at the same time by computing a matrix-matrix product, which can be carried out efficiently by using level 3 BLAS; see [2] for a description of the BLAS.

The paper is organized as follows. Section 2 derives the formulas required and Section 3 describes our computational scheme. The accuracy of the computed inverse depends strongly on the ordering of the nodes t_k. An ordering scheme is proposed in Section 4. Computed examples in Section 5 compare our fast algorithm with the LINPACK [7] subroutine SGEDI. This subroutine computes the inverse of a general $n \times n$ nonsingular matrix by first computing its LU-factorization by Gaussian elimination with partial pivoting and requires $O(n^3)$ arithmetic operations. The computed examples illustrate that our fast algorithm generally yields a more accurate approximation of the inverse than SGEDI when the nodes are ordered suitably. Numerical properties of SGEDI are discussed in [8].

2. The inverse of a Vandermonde-like matrix.

Let the polynomials $q_j(t)$ satisfy the three-term recurrence relation (1.1)–(1.2) and let the matrix W_n be defined by (1.3). Introduce the polynomials

$$p_k(t) = \prod_{j=1}^{k} (t - t_j), \qquad 0 \le k \le n, \tag{2.1}$$

where $p_0(t) = 1$. Consider the divided difference

$$p_n[t, u] = \frac{p_n(t) - p_n(u)}{t - u}, \tag{2.2}$$

and define the set of polynomials $\{r_j\}_{j=0}^{n-1}$ by

$$p_n[t,u] = \sum_{j=0}^{n-1} q_j(t) r_{n-j-1}(u). \tag{2.3}$$

We will see in Section 3 that each of the polynomials r_j is of degree j. Since the nodes t_k are distinct, we have $p_n[t_k, t_l] = 0$ for $k \neq l$, and from (2.3) it follows that

$$\sum_{j=0}^{n-1} q_j(t_k) r_{n-j-1}(t_l) = 0. \tag{2.4}$$

Letting t converge to u in (2.3) yields

$$p_n'(u) = \sum_{j=0}^{n-1} q_j(u) r_{n-j-1}(u). \tag{2.5}$$

Thus, equations (2.4)–(2.5) yield the orthonormality relations

$$\sum_{j=0}^{n-1} \frac{q_j(t_k) r_{n-j-1}(t_l)}{p_n'(t_l)} = \delta_{kl}, \qquad 1 \leq k, l \leq n, \tag{2.6}$$

where δ_{kl} denotes the Kronecker δ-function. The polynomial values $q_j(t_k), 0 \leq j < n$, are the entries of the kth column of W_n. Therefore, it follows from (2.6) that

$$\hat{w}_{lj} = \frac{r_{n-j}(t_l)}{p_n'(t_l)} \tag{2.7}$$

are the elements of the inverse of W_n, i.e., $W_n^{-1} = [\hat{w}_{lj}]_{l,j=1}^n$. The following section presents a recursive scheme for evaluating the polynomials r_j at the nodes.

3. Computation of the inverse.

We describe an algorithm for computing the elements of W_n^{-1}. Express the polynomials p_k defined by (2.1) in terms of the orthogonal polynomials q_j, i.e.,

$$p_k(t) = \sum_{j=0}^{k} \rho_{kj} q_j(t). \tag{3.1}$$

We wish to determine the coefficients ρ_{nj}, $0 \leq j \leq n$, and proceed as follows. Substituting (3.1) into the recursion formula $p_{k+1}(t) = (t - t_{k+1}) p_k(t)$, and applying

$$t q_j(t) = \frac{1}{\theta_j}(q_{j+1}(t) + \gamma_j q_{j-1}(t)) + \beta_j q_j(t), \tag{3.2}$$

yields

$$\begin{aligned} \sum_{j=0}^{k+1} \rho_{k+1,j} q_j(t) = {} & \sum_{j=1}^{k+1} \rho_{k,j-1} \frac{1}{\theta_{j-1}} q_j(t) + \sum_{j=0}^{k} \rho_{kj}(\beta_j - t_{k+1}) q_j(t) \\ & + \sum_{j=0}^{k-1} \rho_{k,j+1} \frac{\gamma_{j+1}}{\theta_{j+1}} q_j(t), \end{aligned} \tag{3.3}$$

which gives the recurrence relations

$$\rho_{k+1,j} = \frac{1}{\theta_{j-1}}\rho_{k,j-1} + (\beta_j - t_{k+1})\rho_{kj} + \frac{\gamma_{j+1}}{\theta_{j+1}}\rho_{k,j+1}, \qquad 0 \le j \le k, \tag{3.4}$$

$$\rho_{k+1,k+1} = \frac{1}{\theta_k}\rho_{kk},$$

where $\theta_{-1} = 1$, $\rho_{k,-1} = 0$ and $\rho_{kj} = 0$ for $j > k$.

Having computed the coefficients ρ_{nj}, $0 \le j \le n$, from (3.4), we can determine the values $r_j(t_k)$ recursively. From (2.2)–(2.3) it follows that

$$p_n(t) = (t - t_k)\sum_{j=0}^{n-1} q_j(t) r_{n\to j-1}(t_k), \qquad 1 \le k \le n. \tag{3.5}$$

Substitution of (3.1)–(3.2) into (3.5) yields

$$\sum_{j=0}^{n} \rho_{nj} q_j(t) = \sum_{j=1}^{n} \frac{1}{\theta_{j-1}} q_j(t) r_{n-j}(t_k) + \sum_{j=0}^{n-1} (\beta_j - t_k) q_j(t) r_{n-j-1}(t_k)$$
$$+ \sum_{j=0}^{n-2} \frac{\gamma_{j+1}}{\theta_{j+1}} q_j(t) r_{n-j-2}(t_k),$$

which gives the recurrence relations

$$r_0(t_k) = \theta_{n-1}\rho_{nn}, \tag{3.6}$$

$$r_{n-j}(t_k) = \theta_{j-1}\left(\rho_{nj} - (\beta_j - t_k) r_{n-j-1}(t_k) - \frac{\gamma_{j+1}}{\theta_{j+1}} r_{n-j-2}(t_k)\right), \qquad 1 \le j < n,$$

where $r_l = 0$ for $l < 0$. In particular, (3.6) shows that $r_j(t)$ is a polynomial of degree j. We use formulas (3.6) to evaluate the polynomials r_j at the nodes. The denominators in (2.7) are determined by evaluating the derivative of the right hand side in (2.1). This gives rise to the following algorithm.

Algorithm 3.1. (*Inversion of Vandermonde-like matrix*)

Input: n, δ, $\{\theta_j\}_{j=0}^{n}$, $\{\beta_j\}_{j=0}^{n-1}$, $\{\gamma_j\}_{j=0}^{n}$, $\{t_j\}_{j=1}^{n}$;
Output: $W_n^{-1} = \{\hat{w}_{kl}\}_{k,l=1}^{n}$;
$\rho_0 := 1/\delta$;
for $k := 1, 2, \ldots, n$ **do** $\rho_k := 0$;
for $j := 1, 2, \ldots, n-1$ **do begin**

$\hat{\rho} := (\beta_0 - t_j)\rho_0 + \frac{\gamma_1}{\theta_1}\rho_1$;
for $k := 1, 2, \ldots, j$ **do begin**

$\hat{\rho}_k := \rho_{k-1}/\theta_{k-1} + (\beta_k - t_j)\rho_k + \frac{\gamma_{k+1}}{\theta_{k+1}}\rho_{k+1}$; $\rho_{k-1} := \hat{\rho}_{k-1}$;
end k;

if $j < n-1$ **then** $\rho_j := \hat{\rho}_j$ **else begin** $\rho_n := \rho_j/\theta_j$; $\rho_j := \hat{\rho}_j$; **end**;
end j;
for $k := 1, 2, \ldots, n$ **do begin**
$r_0(t_k) := \theta_{n-1}\rho_n$;
$r_1(t_k) := \theta_{n-2}(\rho_{n-1} - (\beta_{n-1} - t_k)r_0)$;
for $j := n-2, n-3, \ldots, 1$ **do begin**

$$r_{n-j}(t_k) := \theta_{j-1}(\rho_j - (\beta_j - t_k)r_{n-j-1}(t_k) - \frac{\gamma_{j+1}}{\theta_{j+1}} r_{n-j-2}(t_k));$$

end j;
$p'_n := 1$;
for $j := 1, 2, \ldots, n$ **do if** $j \neq k$ **then** $p'_n := p'_n(t_k - t_j)$;
for $j := 1, 2, \ldots, n$ **do** $\hat{w}_{kj} := r_{n-j}(t_k)/p'_n$;
end k; ■

We remark that the coefficients $\hat{\rho}_k$ and ρ_k can share the same storage locations for almost all values k, and, therefore, the $\hat{\rho}_k$ do not have to be stored in a separate $(n+1)$-vector. The function values $r_{j-1}(t_k)$ can be saved in the storage locations for $\hat{w}_{kj}$. The algorithm requires $12n^2 + O(n)$ arithmetic operations $(+, -, \times, /)$, if the quantities γ_j/θ_j, $1 \leq j \leq n$, are computed only once and stored. The elements of W_n^{-1} are independent of the coefficients θ_j, β_j and γ_j for $j \geq n-1$, but round-off errors introduced during the computations can make the computed entries depend on θ_{n-1}, β_{n-1} and γ_{n-1}. In the computed examples of Section 5 we select these coefficients so that q_n belongs to the same family of orthogonal polynomials as the q_j, $1 \leq j < n$.

4. Ordering of the nodes.

The accuracy achieved with Algorithm 3.1 depends on the ordering of the nodes. In our computed examples of Section 5 we consider two orderings: i) the t_k are ordered monotonically, and ii) the t_k are ordered so that

$$|t_1| = \max_{1 \leq k \leq n} |t_k|, \tag{4.1}$$

$$\prod_{j=1}^{k-1} |t_k - t_j| = \max_{k \leq l \leq n} \prod_{j=1}^{k-1} |t_l - t_j|, \qquad 2 \leq k \leq n.$$

We refer to the ordering (4.1) as *Leja ordering* of the nodes, because of the connection with Leja's work on the approximation of analytic functions by interpolating polynomials [17]. This ordering has also been used by Higham [16] in fast solvers for Vandermonde-like systems based on the Björck-Pereyra [3] algorithm. In that context the ordering (4.1) corresponds to partial pivoting. An application of Leja

ordering to the Newton interpolation formula is discussed in [19]. The ordering of an arbitrary set of n real distinct nodes $\{t_k\}_{k=1}^n$ so as to satisfy (4.1) can be carried out in less than n^2 arithmetic operations.

Numerous computed examples show that Leja ordered nodes generally yield higher accuracy than monotonically ordered nodes; see Section 5 for some illustrative examples. We present a heuristic motivation for using the ordering (4.1). The ordering of the nodes only affects the computation of the coefficients ρ_{kj} for $k < n$ and $0 \leq j < k$. It follows from (3.1) that ρ_{kk} is the reciprocal value of the leading coefficient of $q_k(t)$, and, therefore, is independent of the ordering of the nodes. Substitution of t_j, $1 \leq j \leq k$, into (3.1) yields the linear Vandermonde-like system of equations

$$W_k^T \boldsymbol{r}_k = -\rho_{kk} \boldsymbol{q}_k, \tag{4.2}$$

for $\boldsymbol{r}_k = [\rho_{k0}, \rho_{k1}, \ldots, \rho_{k,k-1}]$, where $\boldsymbol{q}_k = [q_k(t_1), q_k(t_2), \ldots, q_k(t_k)]^T$. We would like to choose the nodes t_j, $1 \leq j \leq k$, so that the solution vector is of small norm. This would reduce the risk of loss of significant digits due to subtraction of large quantities of nearly equal magnitudes and of the same sign during the computation of the coefficients $\rho_{k+1,j}$, $0 \leq j \leq k$, from the ρ_{kj}, $0 \leq j \leq k$ by (3.6). Introduce the matrices W_{kj}^T, $1 \leq j \leq k$, in which column j of W_k^T is replaced by $-\rho_{kk}\boldsymbol{q}_k$. Cramer's rule yields

$$\rho_{kj} = \frac{\det W_{kj}}{\det W_k}, \qquad 0 \leq j < k. \tag{4.3}$$

Assume that the coefficients $\{\rho_{k-1,j}\}_{j=0}^{k-1}$ already have been computed, and therefore the nodes $\{t_j\}_{j=1}^{k-1}$ already have been selected. Let T_n denote the set of n nodes that determine W_n. We now would like to choose the node $t_k \in T_n \setminus \{t_j\}_{j=1}^{k-1}$ so that the coefficients $\{\rho_{kj}\}_{j=0}^{k}$ are of small magnitude. Formula (4.3) suggests that a good way to choose the node t_k is to maximize the magnitude of the denominator. This can be accomplished by factoring $W_k = L_k V_k$, where $L_k = [l_{jm}]_{j,m=1}^k$ is a lower triangular matrix, with diagonal elements $l_{11} = \delta$ and $l_{jj} = \delta \prod_{m=0}^{j-2} \theta_m$ for $j > 1$, and $V_k = [v_{jm}]_{j,m=1}^k$, $v_{jm} = t_m^{j-1}$, is a Vandermonde matrix. Thus,

$$\det W_k = \delta^k \prod_{j=1}^{k-1} \theta_{k-j-1}^j \prod_{1 \leq l < m \leq k} (t_m - t_l).$$

Regarding $|\det W_k|$ as a function of $t = t_k$ yields that

$$t \to |\det W_k| = c_k \prod_{m=1}^{k-1} |t - t_m|,$$

for some constant c_k independent of t. Therefore, Leja ordering is equivalent to ordering the nodes so as to maximize the magnitude of the determinants of leading principal submatrices of W_n over the set of nodes not already chosen.

5. Computed examples.

We present some experiments that illustrate the numerical behavior of Algorithm 3.1 when the nodes are ordered monotonically or so as to satisfy (4.1). The computations were carried out on an HP 9000-720 workstation using single and double precision arithmetic, i.e., with about 7 and 15 significant difits, respectively. The orthogonal polynomials q_j used in our computed examples are Chebyshev polynomials

$$T_j(t) = \cos(j \arccos(t/2))$$

for the interval $[-2, 2]$. We compare the accuracy in the computed inverse determined by our algorithm with the two ordering of the nodes mentioned and in the inverse computed using Gaussian elimination with partial pivoting (GEPP) for the following nodes $\{t_k\}_{k=1}^n$:

$$\text{(5.1)} \qquad \text{equidistant on } [-2,2]: \qquad t_k = -2 + 4\frac{k-1}{n-1},$$

$$\text{(5.2)} \qquad \text{zeros of } T_n(t): \qquad t_k = 2\cos\left(\pi\frac{2k-1}{2n}\right),$$

$$\text{(5.3)} \qquad \text{clustered on } [-2,2]: \qquad t_k = -2 + 4\left(\frac{k-1}{n-1}\right)^2.$$

Note that roughly half of the "clustered" nodes (5.3) lie is the interval $[-2, -1]$. We select the interval $[-2, 2]$ because this yields a scaling that makes underflow and overflow when forming the products (4.1) less likely than if we would choose a longer or shorter interval. The advantages of the interval $[-2, 2]$ follow from that it has capacity 1; see [19] for details.

For each monotonic ordering (M) and Leja ordering (L) of the nodes we compute the inverse W_n^{-1} of the Vandermonde-like matrix W_n in three different ways in single precision arithmetic: i) by our fast algorithm (FASTINV), ii) by Gaussian elimination with partial pivoting applied to W_n (GEPP), iii) by Gaussian elimination with partial pivoting applied to the transpose W_n^T (GEPPT). The computation of the inverse using Gaussian elimination with partial pivoting was performed using the LINPACK subroutine SGEDI; see [7]. While the accuracy of the inverse computed by our algorithm is sensitive to the ordering of the nodes, the accuracy of the inverses computed using GEPP and GEPPT is essentially unaffected by changing the order of the nodes.

In order to compare the accuracy of the inverse W_n^{-1} computed by the different schemes mentioned, we computed the Frobenius norm of the difference of the inverse computed in single precision and the inverse computed using GEPP in double precision. In addition, we computed the relative left residuals

Table 1. *Equidistant nodes on* $[-2,2]$*: monotonic ordering.*

k	FASTINV(M)	GEPP(M)	GEPPT(M)	rlr	rrr	cond_F
4	2.17E-7	8.03E-8	4.95E-8	9.01E-8	8.00E-8	4.51
8	6.80E-6	5.30E-7	4.77E-7	9.76E-7	6.14E-7	1.46E+1
12	5.59E-3	9.36E-6	2.75E-6	1.22E-4	2.16E-5	1.07E+2
16	1.29	8.56E-4	6.75E-4	2.86-E3	2.37E-4	1.18E+3
20	7.94E+1	4.96E-2	4.96E-2	1.51E-2	5.25E-4	1.49E+4

$$\text{rlr} = \frac{\|W_n^{-1}W_n - I\|_F}{\|W_n\|_F \|W_n^{-1}\|_F} \tag{5.4}$$

and the relative right residuals

$$\text{rrr} = \frac{\|W_n W_n^1 - I\|_F}{\|W_n\|_F \|W_n^{-1}\|_F} \tag{5.5}$$

for the approximate inverses W_n^{-1} computed by the different schemes. The relative left and right residuals for the inverse computed by Gaussian elimination with partial pivoting appied to the matrices W_n and W_n^T were consistently very small, of the order of 10^{-8}, independently of the Frobenius norm of the error in the computed inverse. The relative left and right residuals for the inverse computed by our algorithm, on the other hand, varies with the norm of the errror in the computed inverse, as can be seen in the columms labelled rlr and rrr of Tables 1–6. The Frobenius norm condition numbers of the matrices W_n are shown in the Tables 1, 3 and 5 in the column labelled cond_F. We remark that since the Frobenius norm is invariant under unitary transformations, and since reordering of the nodes corresponds to a permutation of columns of W_n, the computed condition number is independent of the ordering of the nodes t_j.

EXAMPLE 5.1. We use equidistant nodes (5.1) introduced in monotonically increasing order and Leja order. The condition number of the matrix W_n grows rapidly with n for this choice of nodes; see Table 1. When the nodes are ordered monotonically, the error in the computation of the inverse by the fast algorithm grows quickly with the size of the matrix, as it shows in Table 1. Notice that in this case the norm of the error for the fast algorithms is up to three orders of magnitude larger than the norm of the error for GEPP and GEPPT. However, Table 2 shows that the fast inversion algorithm yields higher accuracy than GEPP and GEPPT, in fact up to two digits, when the nodes are ordered so as to satisfy (4.1). The sensitivity of the accuracy of the inverse computed by our fast algorithm to the ordering of the nodes is more noticeable as the size of the matrix increases. A difference of 5 digits in the accuracy of the inverse can be observed when $n = 16$ simply by changing the ordering of the nodes.

Table 2. *Equidistant nodes on* $[-2,2]$ *Leja ordering.*

k	FASTINV(L)	GEPP(L)	GEPPT(L)	rlr	rrr
4	8.66E-8	6.32E-8	5.93E-8	4.04E-8	3.02E-8
8	4.38E-7	3.20E-7	2.79E-7	6.99E-8	4.45E-8
12	5.62E-6	1.04E-5	2.00E-6	1.61E-7	1.09E-7
16	3.92E-5	1.05E-3	6.74E-4	1.28E-7	6.60E-8
20	6.96E-4	2.89E-2	4.92E-2	1.85E-7	1.39E-7

Table 3. *The nodes are zeros of Chebyshev polynomials on* $[-2,2]$: *monotonic ordering.*

k	FASTINV(M)	GEPP(M)	GEPPT(M)	rlr	rrr	cond_F
4	3.11E-7	8.19E-8	1.32E-7	1.10E-7	1.07E-7	4.18
8	3.40E-5	2.05E-7	1.97E-7	8.87E-6	8.30E-6	8.22
12	5.95E-4	3.59E-7	3.36E-7	1.35E-4	1.20E-4	1.22E+1
16	5.56E-2	3.79E-7	3.51E-7	1.03E-2	9.68E-3	1.62E+1
20	3.48E+1	5.76E-7	5.12E-7	2.30E-1	2.18E-1	2.02E+1

Table 4. *The nodes are zeros of Chebyshev polynomials on* $[-2,2]$: *Leja ordering.*

k	FASTINV(L)	GEPP(L)	GEPPT(L)	rlr	rrr
4	9.00E-8	1.340E-7	1.33E-7	3.84E-8	3.97E-8
8	3.51E-7	1.96E-7	2.42E-7	8.84E-8	7.94E-8
12	7.22E-7	3.48E-7	3.39E-7	1.35E-7	1.35E-7
16	9.23E-7	3.70E-7	3.81E-7	1.77E-7	1.67E-7
20	1.27E-6	4.70E-7	4.79E-7	2.17E-7	2.15E-7

Table 5. *The nodes are clustered on* $[-2,2]$: *monotonic ordering.*

k	FASTINV(M)	GEPP(M)	GEPPT(M)	rlr	rrr	cond_F
4	3.13E-8	2.39E-7	2.64E-7	2.57E-8	2.53E-8	6.70
8	2.04E-4	2.64E-4	1.33E-4	1.23E-6	4.30E-7	4.32E+2
12	2.78	1.54	8.50E-1	1.35E-4	1.12E-5	7.08E+4

Table 6. *The nodes are clustered on* $[-2,2]$: *Leja ordering.*

k	FASTINV(M)	GEPP(M)	GEPPT(M)	rlr	rrr
4	3.13E-8	2.73E-7	2.48E-7	2.57E-8	3.37E-8
8	3.54E-5	5.33E-4	1.33E-4	2.31E-7	6.41E-8
12	3.97E-3	1.33	8.50E-1	1.93E-7	9.74E-8

EXAMPLE 5.2. In this example the nodes are the zeros (5.2) of the Chebyshev polynomial $T_n(t)$, ordered so as to satisfy (4.1), or in monotonically decreasing order. We remark that, unlike in Example 5.1, the Vandermonde-like matrices W_n which result are fairly well-conditioned, as it is shown in Table 3. The accuracy of the inverse computed by Algorithm 3.1 with Leja ordering of the nodes is essentially the same as the accuracy of the inverse computed by GEPP and GEPPT; see Table 4. As in Example 5.1, monotonic ordering of the nodes causes Algorithm 3.1 to yield poor accuracy except for very small values of n, as is illustrated by Table 3.

EXAMPLE 5.3. In this example we use the set of clustered nodes defined in (5.3), introduced in either monotonically increasing order or Leja order. Table 5 shows that the matrices W_n obtained are very ill-conditioned. Note that for this set of nodes the accuracy of the inverse computed by our algorithm with monotonic ordering of the nodes is essentially the same as the accuracy of the inverse computed by GEPP, while GEPPT yields a more accurate inverse, as shown in Table 5. Table 6 shows that the inverse computed by our algorithm with Leja ordering of the nodes is more accurate than the inverse computed by GEPP or GEPPT.

The examples above are typical for our computational experience with a wide variety of Vandermonde-like matrices. Our numerical experiments suggest that, in general, our fast algorithm used in conjunction with Leja ordering of the nodes yields at least as high accuracy in the computed inverse of a Vandermonde-like matrix as Gaussian elimination with partial pivoting (GEPP and GEPPT). However, Gaussian elimination (GEPP and GEPPT) sometimes yields slightly smaller residual errors (5.4) and (5.5).

REFERENCES

1. M. Almacany, C. B. Dunham and J. Williams, *Discrete Chebyshev approximation by interpolating rationals*, IMA J.Numer. Anal., 4 (1984), pp. 467–477.
2. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, 1992.
3. Å. Björck and V. Pereyra, *Solution of Vandermonde systems of equations*, Math. Comp., 24 (1970), pp. 893–903.
4. J. R. Bunch, *The weak and strong stability of algorithms in numerical linear algebra*, Linear Algebra Appl., 88/89 (1988), pp. 49–66.
5. D. Calvetti and L. Reichel, *A Chebyshev-Vandermonde solver*, Linear Algebra Appl., 172 (1992), pp. 219–229.
6. J. Chun and T. Kailath, *Displacement structure for Hankel, Vandermonde and related (derived) matrices*, Linear Algebra Appl., 151 (1991), pp. 199–227.
7. J. J. Donngarra, J. R. Bunch, C. B. Moler and G. W. Stewart, *LINPACK Users' Guide*, SIAM Philadelhia, 1979.
8. J. Du Croz and N. J. Higham, *Stability of methods for matrix inversion*, IMA J. Numer. Anal., 12 (1992), pp. 1–19.
9. W. Gautschi, *The condition of Vandermonde-like matrices involving orthogonal polynomials*, Linear Algebra Appl., 52/53 (1983), pp. 293–300.

10. W. Gautschi, *How (un)stable are Vandermonde systems?*, in Asymptotic and Computational Analysis, ed. R. Wong, Lecture Notes in Pure and Applied Mathematics, v. 124, Dekker, New York, 1990, pp. 193–210.
11. W. Gautschi and G. Inglese, *Lower bounds for the condition number of Vandermonde matrices*, Numer. Math., 52 (1988), pp. 241–250.
12. I. Gohberg, T. Kailath and I. Koltracht, *Efficient solution of linear systems of equations with recursive structure*, Linear Algebra Appl., 80 (1986) pp. 81–113.
13. G. Heinig and K. Rost, *Algebraic Methods for Toeplitz-like Matrices and Operators*, Birkhäuser, Basel, 1984.
14. N. J. Higham, *Error analysis of the Björck-Pereyra algorithms for solving Vandermonde system*, Numer. Math., 50 (1987), pp. 613–632.
15. N. J. Higham, *Fast solution of Vandermonde-like systems involving orthogonal polynomials*, IMA J. Numer. Anal., 8 (1988), pp. 473–486.
16. N. J. Higham, *Stability analysis of algorithms for solving confluent Vandermonde-like systems*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 23–41.
17. F. Leja, *Sur certaines suites liées aux ensemble plans et leur application à la representation conforme*, Ann. Polon. Math., 4 (1957), pp. 8–13.
18. N. Macon and A. Spitzbart, *Inverses of Vandermonde matrices*, Amer. Math. Monthly, 65 (1958), pp. 95–100.
19. L. Reichel, *Newton interpolation at Leja points*, BIT, 30 (1990), pp. 332–346.
20. L. Reichel and G. Opfer, *Chebyshev-Vandermonde systems*, Math. Comp., 57 (1991), pp. 703–721.
21. W. P. Tang and G. H. Golub, *The block decomposition of a Vandermonde matrix and its applications*, BIT, 21 (1981), pp. 505–517.
22. J. F. Traub, *Associated polynomials and uniform methods for the solution of linear problems*, SIAM Review, 8 (1966), pp. 277–301.
23. L. Verde-Star, *Inverses of generalized Vandermonde matrices*, J. Math. Anal. Appl., 131 (1988), pp. 341–353.