SIAM J. MATRIX ANAL. APPL.
Vol. 11, No. 1, pp. 23–41, January 1990

STABILITY ANALYSIS OF ALGORITHMS FOR SOLVING CONFLUENT VANDERMONDE-LIKE SYSTEMS*

NICHOLAS J. HIGHAM†

Abstract. A confluent Vandermonde-like matrix $P(\alpha_0, \alpha_1, \cdots, \alpha_n)$ is a generalisation of the confluent Vandermonde matrix in which the monomials are replaced by arbitrary polynomials. For the case where the polynomials satisfy a three-term recurrence relation algorithms for solving the systems $Px = b$ and $P^Ta = f$ in $O(n^2)$ operations are derived. Forward and backward error analyses that provide bounds for the relative error and the residual of the computed solution are given. The bounds reveal a rich variety of problem-dependent phenomena, including both good and bad stability properties and the possibility of extremely accurate solutions. To combat potential instability, a method is derived for computing a "stable" ordering of the points α_i; it mimics the interchanges performed by Gaussian elimination with partial pivoting, using only $O(n^2)$ operations. The results of extensive numerical tests are summarised, and recommendations are given for how to use the fast algorithms to solve Vandermonde-like systems in a stable manner.

Key words. Vandermonde matrix, orthogonal polynomials, Hermite interpolation, Clenshaw recurrence, forward error analysis, backward error analysis, stability, iterative refinement

AMS(MOS) subject classifications. primary 65F05, 65G05

C. R. classification. G.1.3

1. Introduction.

Let $\{p_k(t)\}_{k=0}^n$ be a set of polynomials, where p_k is of degree k, and let $\alpha_0, \alpha_1, \cdots, \alpha_n$ be real scalars, ordered so that equal points are contiguous, that is,

$$\alpha_i = \alpha_j \quad (i<j) \quad \Rightarrow \quad \alpha_i = \alpha_{i+1} = \cdots = \alpha_j. \tag{1.1}$$

We define the *confluent Vandermonde-like matrix*

$$P = P(\alpha_0, \alpha_1, \cdots, \alpha_n) = [q_0(\alpha_0), q_1(\alpha_1), \cdots, q_n(\alpha_n)] \in \mathbf{R}^{(n+1)\times(n+1)},$$

where the vectors $q_j(t)$ are defined recursively by

$$q_j(t) = \begin{cases} [p_0(t), p_1(t), \cdots, p_n(t)]^T & \text{if } j = 0 \text{ or } \alpha_j \neq \alpha_{j-1}, \\ \dfrac{d}{dt} q_{j-1}(t), & \text{otherwise.} \end{cases}$$

In the case of the monomials, $p_k(t) = t^k$, this definition yields the well-known confluent Vandermonde matrix [9], [4]. When the points α_i are distinct we can write $P = (p_i(\alpha_j))_{i,j=0}^n$, and P is referred to as a nonconfluent Vandermonde-like matrix [12]. For all polynomials and points, P is nonsingular; this follows from the derivation of the algorithms in § 2.

Various applications give rise to confluent or nonconfluent Vandermonde or Vandermonde-like systems

$$Px = b \quad \text{(primal)} \tag{1.2}$$

and

$$P^Ta = f \quad \text{(dual)}. \tag{1.3}$$

* Received by the editors December 7, 1987; accepted for publication (in revised form) March 22, 1989.
† Department of Mathematics, University of Manchester, Manchester M13 9PL, United Kingdom (NA.NHIGHAM@NA-NET.STANFORD.EDU).

Three examples are the construction of quadrature formulae [2], [14], [15], rational Chebyshev approximation [1], and the approximation of linear functionals [3], [22].

For the monomials, with distinct points α_i, efficient algorithms for solving the primal and dual Vandermonde systems are given in [5]. These algorithms have been generalised in two ways: in [4] they are extended to confluent Vandermonde matrices, and in [12] they are extended to nonconfluent Vandermonde-like matrices, under the assumption that the polynomials $p_k(t)$ satisfy a three-term recurrence relation. In § 2 we blend these two extensions, obtaining algorithms for solving (1.2) and (1.3), which include those in [5], [4], [12] as special cases. We also show how to compute the residual vector of the dual system efficiently using a generalisation of the Clenshaw recurrence.

In § 3 we present an error analysis of the algorithms of § 2. The analysis provides bounds for both the forward error and the residual of the computed solutions. It makes no assumptions about the ordering or signs of the points α_i, and thus extends the error analysis in [11].

To interpret the analysis we compare the error bounds with appropriate "ideal" bounds. This leads, in § 4, to pleasing stability results for certain classes of problem, but also reveals grave instabilities in some other cases. The instabilities can be interpreted as indicating that the natural, increasing ordering of the points can be a poor one. In § 5 we derive a technique for computing a more generally appropriate ordering. The method is based on a connection derived between the stability of the fast algorithms and the stability of Gaussian elimination. As a means for restoring stability, the re-ordering approach has several advantages over iterative refinement in single precision, which was used in [12] and [21].

Numerical experiments are presented in § 6. Finally, in § 7 we offer recommendations on the use of the fast algorithms for solving Vandermonde-like systems in a stable manner.

2. Algorithms. Assume that the polynomials $p_k(t)$ satisfy the three-term recurrence relation

$$p_{j+1}(t) = \theta_j(t-\beta_j)p_j(t) - \gamma_j p_{j-1}(t), \qquad j \geqq 1, \tag{2.1a}$$

with

$$p_0(t) = 1, \qquad p_1(t) = \theta_0(t-\beta_0)p_0(t), \tag{2.1b}$$

where $\theta_j \neq 0$ for all j. Algorithms for solving the systems (1.2) and (1.3) can be derived by using a combination of the techniques in [4] and [12]. Denote by $r(i) \geqq 0$ the smallest integer for which $\alpha_i = \alpha_{i-1} = \cdots = \alpha_{r(i)}$. Considering, first, the dual system (1.3), we note that

$$\phi(t) = \sum_{i=0}^{n} a_i p_i(t) \tag{2.2}$$

satisfies

$$\phi^{(i-r(i))}(\alpha_i) = f_i, \qquad 0 \leqq i \leqq n.$$

Thus ϕ is a Hermite interpolating polynomial for the data $\{\alpha_i, f_i\}$, and our task is to obtain its representation in terms of the basis $\{p_i(t)\}_{i=0}^{n}$. As a first step, following [4], we construct the divided difference form of ϕ:

$$\phi(t) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1} (t-\alpha_j). \tag{2.3}$$

The (confluent) divided differences $c_i = f[\alpha_0, \alpha_1, \cdots, \alpha_i]$ may be generated using the recurrence relation [4], [20, p. 55]

(2.4)
$$f[\alpha_{j-k-1}, \cdots, \alpha_j] = \begin{cases} \dfrac{f[\alpha_{j-k}, \cdots, \alpha_j] - f[\alpha_{j-k-1}, \cdots, \alpha_{j-1}]}{\alpha_j - \alpha_{j-k-1}}, & \alpha_j \neq \alpha_{j-k-1}, \\ \dfrac{f_{r(j)+k+1}}{(k+1)!}, & \alpha_j = \alpha_{j-k-1}. \end{cases}$$

Now we need to generate the a_i in (2.2) from the c_i in (2.3). We can use the recurrences in [12], which are unaffected by confluency; these are derived by expanding (2.3) using nested multiplication, and using the recurrence relations (2.1) to express the results as a linear combination of the polynomials p_j.

In the following algorithm Stage I computes the confluent divided differences. We use an implementation of (2.4) from [6, pp. 68–69], in preference to the more complicated version in [4]. Stage II is identical to the corresponding part of the dual algorithm in [12].

ALGORITHM 2.1 (Dual, $P^T a = f$). Given parameters $\{\theta_j, \beta_j, \gamma_j\}_{j=0}^{n-1}$, a vector f, and points $\{\alpha_i\}_{i=0}^{n}$ satisfying (1.1), this algorithm solves the dual system $P^T a = f$.

Stage I: Set $c = f$
For $k = 0$ to $n - 1$
  $clast = c_k$
  For $j = k + 1$ to n
    If $\alpha_j = \alpha_{j-k-1}$ then
      $c_j = c_j/(k + 1)$
    else
      $temp = c_j$
      $c_j = (c_j - clast)/(\alpha_j - \alpha_{j-k-1})$
      $clast = temp$
    endif
  endfor j
endfor k

Stage II: Set $a = c$
$a_{n-1} = a_{n-1} + (\beta_0 - \alpha_{n-1})a_n$
$a_n = a_n/\theta_0$
For $k = n - 2$ to 0 step -1
  $a_k = a_k + (\beta_0 - \alpha_k)a_{k+1} + (\gamma_1/\theta_1)a_{k+2}$
  For $j = 1$ to $n - k - 2$
    $a_{k+j} = a_{k+j}/\theta_{j-1} + (\beta_j - \alpha_k)a_{k+j+1} + (\gamma_{j+1}/\theta_{j+1})a_{k+j+2}$
  endfor j
  $a_{n-1} = a_{n-1}/\theta_{n-k-2} + (\beta_{n-k-1} - \alpha_k)a_n$
  $a_n = a_n/\theta_{n-k-1}$
endfor k □

In the algorithm the vectors c and a have been used for clarity; in fact both can be replaced by f, so that the right-hand side is transformed into the solution without using any extra storage. Assuming that the values γ_j/θ_j are given (note that γ_j appears only in

the terms γ_j/θ_j) the computational cost of Algorithm 2.1 is $n(2n + 1)M$ and at most $n(5n + 3)/2A$, where M denotes a multiplication or division, and A an addition or subtraction.

An algorithm for solving the primal system can be deduced immediately, using the approach of [4], [5], [12]. We will show in § 3 that the dual algorithm effectively multiplies the right-hand side vector f by P^{-T}, employing a factorisation of P^{-T} into the product of $2n$ triangular matrices. Taking the transpose of this product we obtain a representation of P^{-1}, from which it is easy to write an algorithm for computing $x = P^{-1}b$.

ALGORITHM 2.2 (Primal, $Px = b$). Given parameters $\{\theta_j, \beta_j, \gamma_j\}_{j=0}^{n-1}$, a vector b, and points $\{\alpha_i\}_{i=0}^{n}$ satisfying (1.1), this algorithm solves the primal system $Px = b$.

Stage I: Set $d = b$
For $k = 0$ to $n - 2$
For $j = n - k$ to 2 step -1
$d_{k+j} = (\gamma_{j-1}/\theta_{j-1})d_{k+j-2} + (\beta_{j-1} - \alpha_k)d_{k+j-1} + d_{k+j}/\theta_{j-1}$
endfor j
$d_{k+1} = (\beta_0 - \alpha_k)d_k + d_{k+1}/\theta_0$
endfor k
$d_n = (\beta_0 - \alpha_{n-1})d_{n-1} + d_n/\theta_0$

Stage II: Set $x = d$
For $k = n - 1$ to 0 step -1
$xlast = 0$
For $j = n$ to $k + 1$ step -1
If $\alpha_j = \alpha_{j-k-1}$ then
$x_j = x_j/(k + 1)$
else
$temp = x_j/(\alpha_j - \alpha_{j-k-1})$
$x_j = temp - xlast$
$xlast = temp$
endif
endfor j
$x_k = x_k - xlast$
endfor k □

Algorithm 2.2 has, by construction, the same operation count (to within one addition) as Algorithm 2.1. Values of θ_j, β_j, γ_j for some polynomials of interest in Algorithms 2.1 and 2.2 are given in Table 2.1.

For practical use of Algorithms 2.1 and 2.2 it is important to be able to calculate the residual, in order to test that the algorithms have been coded correctly (for example) and, perhaps, to implement iterative refinement (see § 5). Ordinarily, residual computation for linear equations is trivial, but in this context the coefficient matrix is not given explicitly, and computing the residual turns out to be conceptually almost as difficult, and computationally as expensive, as solving the linear system!

To compute the residual for the dual system we need a means for evaluating $\phi(t)$ in (2.2) and its first $k \leqq n$ derivatives, where $k = \max_i(i - r(i))$ is the *order of confluency*. Since the polynomials p_j satisfy a three-term recurrence relation we can use an extension of the Clenshaw recurrence formula. The following algorithm implements the appropriate

TABLE 2.1
Parameters in the three term recurrence (2.1).

Polynomial	θ_j	β_j	γ_j	
Monomials	1	0	0	
Chebyshev	2*	0	1	$*\theta_0 = 1$
Legendre*	$\frac{2j+1}{j+1}$	0	$\frac{j}{j+1}$	$*p_j(1) \equiv 1$
Hermite	2	0	$2j$	
Laguerre	$-\frac{1}{j+1}$	$2j+1$	$\frac{j}{j+1}$	

recurrences, which are given in [18]; we note that an alternative derivation to that in [18] is to differentiate repeatedly the original Clenshaw recurrence and to rescale so as to consign factorial terms to a "clean-up" loop at the end.

ALGORITHM 2.3 (Extended Clenshaw recurrence [18]). This algorithm computes the $k+1$ values $y_j = \phi^{(j)}(x)$, $0 \leqq j \leqq k$, where ϕ is given by (2.2) and $k \leqq n$. It uses a work vector z of order k.

```
Set y_i = z_i = 0        (i = 0, 1, ..., k)
y_0 = a_n
For j = n - 1 to 0 step -1
    temp = y_0
    y_0 = θ_j(x - β_j)y_0 - γ_{j+1}z_0 + a_j
    z_0 = temp
    For i = 1 to min (k, n - j)
        temp = y_i
        y_i = θ_j((x - β_j)y_i + z_{i-1}) - γ_{j+1}z_i
        z_i = temp
    endfor i
endfor j
m = 1
For i = 2 to k
    m = m*i
    y_i = m*y_i
endfor i      □
```

Cost. $3[n + kn - k(k-1)/2](M + A) + 2 \max\{0, k-1\}M$.

Computing the residual using Algorithm 2.3 costs between approximately $3n^2/2(M + A)$ (for full confluency) and $3n^2(M + A)$ (for the nonconfluent case).

The residual for the primal system can be computed in a similar way, using recurrences obtained by differentiating (2.1).

3. Rounding error analysis. In this section we derive bounds for the forward error and the residual of the computed solution obtained from Algorithm 2.1 in floating point arithmetic. Because of the inherent duality between Algorithms 2.1 and 2.2 all the results that we state have obvious counterparts for Algorithm 2.2.

The key to the analysis is the observation that Algorithm 2.1 can be expressed entirely in the language of matrix-vector products (a similar observation drives the analysis

of a related problem in [19]). In Stage I, letting $c^{(k)}$ denote the vector c at the start of the kth iteration of the outer loop, we have

$$c^{(0)} = f, \qquad c^{(k+1)} = L_k c^{(k)}, \quad k = 0, 1, \cdots, n-1. \tag{3.1}$$

We will adopt the convention that the subscripts of all vectors and matrices run from 0 to n. The matrix L_k is lower triangular and agrees with the identity matrix in rows 0 to k. The remaining rows can be described by, for $k+1 \leqq j \leqq n$,

$$e_j^T L_k = \begin{cases} e_j^T/(k+1) & \text{if } \alpha_j = \alpha_{j-k-1}, \\ (e_j^T - e_s^T)/(\alpha_j - \alpha_{j-k-1}) & \text{for some } s<j, \text{ otherwise,} \end{cases}$$

where e_j is column j of the identity matrix. Similarly, Stage II can be expressed as

$$a^{(n)} = c^{(n)}, \qquad a^{(k)} = U_k a^{(k+1)}, \quad k = n-1, n-2, \cdots, 0. \tag{3.2}$$

The matrix U_k is upper triangular, it agrees with the identity matrix in rows 0 to $k-1$ and it has zeros everywhere above the first two superdiagonals.

From (3.1) and (3.2) we see that the overall effect of the Algorithm 2.1 is to evaluate step by step the product

$$a = U_0 \cdots U_{n-1} L_{n-1} \cdots L_0 f \equiv P^{-T} f. \tag{3.3}$$

We adopt the standard model of floating point arithmetic [6, p. 9]:

$$\mathrm{fl}(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \quad |\delta| \leqq u, \quad \operatorname{op} = +, -, *, /, \tag{3.4}$$

where u is the unit roundoff. In line with the general philosophy of rounding error analysis we do not aim for the sharpest possible constants in our bounds, and are thus able to keep the analysis quite short.

THEOREM 3.1. *Let Algorithm* 2.1 *be applied in floating point arithmetic to floating point data* $\{\alpha_i, f_i\}_{i=0}^n$. *Provided that no overflows are encountered the algorithm runs to completion, and the computed solution* $\hat{a}$ *satisfies*

$$|\hat{a} - a| \leqq c(n,u)|U_0| \cdots |U_{n-1}||L_{n-1}| \cdots |L_0||f|,$$

where, with $\mu = (1+u)^4 - 1$, $c(n, u) = (1+\mu)^{2n} - 1 = 8nu + O(u^2)$.

Proof. First, note that Algorithm 2.1 must succeed in the absence of overflow, because division by zero cannot occur.

Because of the form of L_k, straightforward application of the model (3.4) to the components of (3.1) yields

$$\hat{c}^{(k+1)} = D_k L_k \hat{c}^{(k)}, \tag{3.5}$$

where $D_k = \operatorname{diag}(d_i)$, with $d_i = 1$ for $0 \leqq i \leqq k$, and $(1-u)^3 \leqq d_i \leqq (1+u)^3$ for $k+1 \leqq i \leqq n$. Thus

$$|D_k - I| \leqq [(1+u)^3 - 1] I,$$

and hence (3.5) may be written in the form

$$\hat{c}^{(k+1)} = (L_k + \Delta L_k)\hat{c}^{(k)}, \qquad |\Delta L_k| \leqq [(1+u)^3 - 1]|L_k|. \tag{3.6}$$

Turning to (3.2), we can regard the multiplication $a^{(k)} = U_k a^{(k+1)}$ as comprising a sequence of three-term inner products. Analysing these in standard fashion, using (3.4), we arrive at the equation

$$\hat{a}^{(k)} = (U_k + \Delta U_k)\hat{a}^{(k+1)}, \qquad |\Delta U_k| \leqq [(1+u)^4 - 1]|U_k|, \tag{3.7}$$

where we have taken into account the rounding errors in forming $u_{i,i+1}^{(k)} = \beta_j - \alpha_k$ and $u_{i,i+2}^{(k)} = \gamma_{j+1}/\theta_{j+1}$ $(i = k + j)$.

Since $\hat{c}^{(0)} = f$, and $\hat{a} = \hat{a}^{(0)}$, (3.6) and (3.7) imply that

$$\hat{a} = (U_0 + \Delta U_0)\cdots(U_{n-1} + \Delta U_{n-1})(L_{n-1} + \Delta L_{n-1})\cdots(L_0 + \Delta L_0)f, \tag{3.8}$$

where, on weakening (3.6), we have

$$|\Delta U_k| \leqq \mu |U_k|, \quad |\Delta L_k| \leqq \mu |L_k|, \quad \mu = (1+u)^4 - 1.$$

Now we make use of the following perturbation result that is easily proved by induction: *For matrices $X_j + \Delta X_j$, if $|\Delta X_j| \leqq \delta |X_j|$ for all j, then*

$$\left| \prod_{j=0}^{m} (X_j + \Delta X_j) - \prod_{j=0}^{m} X_j \right| \leqq [(1+\delta)^{m+1} - 1] \prod_{j=0}^{m} |X_j|.$$

Applying this result to the difference of (3.8) and (3.3), we obtain the desired bound for the forward error. □

In the course of proving Theorem 3.1 we derived (3.8), a form of backward error result. However, (3.8) is of little intrinsic interest because the perturbations it contains are associated with the matrices U_k and L_k, and not in any exploitable way with the original data $\{\alpha_i, f_i\}$ (and, possibly, $\{\theta_j, \beta_j, \gamma_j\}$). The appropriate way to analyse backward error, as we will explain in § 4.2, is to look at the residual, $r = f - P^T a$ (cf. the similar approach taken in a different context in [7]). Rearranging (3.8),

$$f = (L_0 + \Delta L_0)^{-1}\cdots(L_{n-1} + \Delta L_{n-1})^{-1}(U_{n-1} + \Delta U_{n-1})^{-1}\cdots(U_0 + \Delta U_0)^{-1}\hat{a}. \tag{3.9}$$

From the proof of Theorem 3.1 we can show that

$$(L_k + \Delta L_k)^{-1} = L_k^{-1} + E_k, \qquad |E_k| \leqq [(1-u)^{-3} - 1]|L_k^{-1}|.$$

Strictly, an analogous bound for $(U_k + \Delta U_k)^{-1}$ does not hold, since ΔU_k cannot be expressed in the form of a diagonal matrix times U_k. However, it seems reasonable to make a simplifying assumption that such a bound is valid, say,

$$(U_k + \Delta U_k)^{-1} = U_k^{-1} + F_k, \qquad |F_k| \leqq [(1-u)^{-4} - 1]|U_k^{-1}|. \tag{3.10}$$

Then, writing (3.9) as

$$\begin{aligned} f &= (L_0^{-1} + E_0)\cdots(L_{n-1}^{-1} + E_{n-1})(U_{n-1}^{-1} + F_{n-1})\cdots(U_0^{-1} + F_0)\hat{a} \\ &= P^T\hat{a} + \Bigg(\sum_{k=0}^{n-1} L_0^{-1}\cdots L_{k-1}^{-1} E_k L_{k+1}^{-1}\cdots L_{n-1}^{-1} U_{n-1}^{-1}\cdots U_0^{-1} \\ &\qquad + \sum_{k=0}^{n-1} L_0^{-1}\cdots L_{n-1}^{-1} U_{n-1}^{-1}\cdots U_{k+1}^{-1} F_k U_{k-1}^{-1}\cdots U_0^{-1} \Bigg)\hat{a} + O(u^2), \end{aligned}$$

we obtain the following result.

THEOREM 3.2. *Under the assumption* (3.10), *the residual of the computed solution $\hat{a}$ from Algorithm* 2.1 *is bounded by*

$$|f - P^T\hat{a}| \leqq d_n u |L_0^{-1}|\cdots|L_{n-1}^{-1}||U_{n-1}^{-1}|\cdots|U_0^{-1}||\hat{a}| + O(u^2),$$

with $d_n = 7n$. □

In common with most error analyses the one above uses a profusion of triangle and submultiplicative inequalities, and consequently the bounds will usually be unrealistic

error *estimates*. However, as we will see, they are well able to reveal extremes of behaviour, with respect to accuracy and stability.

4. Implications for stability. Now we pursue the implications of the error analysis. To interpret the forward error bound of Theorem 3.1 and the backward error bound of Theorem 3.2 we need to use two different notions of stability. We consider these separately in §§ 4.1 and 4.2, since there is no simple relation between them and each is of independent interest. We will focus attention mainly on the nonconfluent case, making brief comments about the effects of confluency.

4.1. Weak stability. To interpret the forward error bound

$$|\hat{a}-a| \leqq c(n,u)|U_0|\cdots|U_{n-1}||L_{n-1}|\cdots|L_0||f| \tag{4.1}$$

from Theorem 3.1 we need an "ideal" bound with which to compare it. Following the approach of [11, § 4] we consider the effect of a small, element-wise perturbation in f. If $P^T(a+\delta a) = f + \delta f$ with $|\delta f| \leqq u|f|$, then it is easy to show that

$$|\delta a| \leqq u|P^{-T}||f|, \tag{4.2}$$

and that equality is attained for suitable choice of δf. This prompts the informal definition that an algorithm for solving $P^T a = f$ in floating point arithmetic is *weakly stable* if the error in the computed solution is not much larger, in some appropriate measure, than the upper bound in (4.2). A useful way to interpret the definition is that if the machine right-hand side vector is inexact, then a weakly stable algorithm solves the machine problem to as good an accuracy as the data warrants.

By comparing (4.1) and (4.2) we see that Algorithm 2.1 is certainly weakly stable if

$$|U_0|\cdots|U_{n-1}||L_{n-1}|\cdots|L_0| \leqq b_n|P^{-T}| = b_n|U_0\cdots U_{n-1}L_{n-1}\cdots L_0| \tag{4.3}$$

for some small constant $b_n \geqq 1$. This condition requires that there be little subtractive cancellation in the product $U_0\cdots U_{n-1}L_{n-1}\cdots L_0$. Suppose the points are distinct and consider the case $n = 3$. We have

$$\begin{aligned}
P^{-T} &= U_0U_1U_2L_2L_1L_0 \\
&\equiv \begin{bmatrix} 1 & \beta_0-\alpha_0 & \gamma_1/\theta_1 & 0 \\ & \theta_0^{-1} & \beta_1-\alpha_0 & \gamma_2/\theta_2 \\ & & \theta_1^{-1} & \beta_2-\alpha_0 \\ & & & \theta_2^{-1} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ & 1 & \beta_0-\alpha_1 & \gamma_1/\theta_1 \\ & & \theta_0^{-1} & \beta_1-\alpha_1 \\ & & & \theta_1^{-1} \end{bmatrix} \\
&\times \begin{bmatrix} 1 & 0 & 0 & 0 \\ & 1 & 0 & 0 \\ & & 1 & \beta_0-\alpha_2 \\ & & & \theta_0^{-1} \end{bmatrix} \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & -1/(\alpha_3-\alpha_0) & 1/(\alpha_3-\alpha_0) \end{bmatrix} \\
&\times \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & -1/(\alpha_2-\alpha_0) & 1/(\alpha_2-\alpha_0) & \\ 0 & 0 & -1/(\alpha_3-\alpha_1) & 1/(\alpha_3-\alpha_1) \end{bmatrix} \\
&\times \begin{bmatrix} 1 & & & \\ -1/(\alpha_1-\alpha_0) & 1/(\alpha_1-\alpha_0) & & \\ 0 & -1/(\alpha_2-\alpha_1) & 1/(\alpha_2-\alpha_1) & \\ 0 & 0 & -1/(\alpha_3-\alpha_2) & 1/(\alpha_3-\alpha_2) \end{bmatrix}.
\end{aligned} \tag{4.4}$$

There is no subtractive cancellation in this product as long as each matrix has the alternating sign pattern defined, for $A = (a_{ij})$, by $(-1)^{i+j}a_{ij} \geq 0$. This sign pattern holds for the matrices L_i if the points α_i are arranged in increasing order. The matrices U_i have the required sign pattern provided that (in general)

$$\theta_i > 0, \qquad \gamma_i \geq 0 \quad \text{for all } i, \quad \text{and} \quad \beta_i - \alpha_k \leq 0 \quad \text{for all } i+k \leq n-1. \tag{4.5}$$

Hence we obtain the following result, where we weaken the last condition to $\beta_i = 0$ and $\alpha_i \geq 0$ for all i since $\beta_i = 0$ holds for most of the commonly occurring polynomials.

COROLLARY 4.1. *If* $0 \leq \alpha_0 < \alpha_1 < \cdots < \alpha_n$, *and* $\theta_i > 0$, $\beta_i = 0$ *and* $\gamma_i \geq 0$ *for all* i, *then*

$$|\hat{a} - a| \leq c(n,u)|P^{-T}||f|,$$

where $c(n, u)$ *is defined in Theorem* 3.1, *and hence, under these conditions, Algorithm* 2.1 *is weakly stable.* □

Corollary 4.1 is stated without proof in [12, § 3]. In the special case of the monomials ($\theta_i = 1$, $\beta_i = \gamma_i = 0$) Corollary 4.1 is essentially the same as the main result of [11, Thm. 2.3]. As shown in [11], the bound in the corollary can imply high relative accuracy even when P^{-T} is large. To see this, note that under the conditions of the corollary P^{-T} has the alternating sign pattern, since each of its factors does; thus if $(-1)^i f_i \geq 0$ then $|P^{-T}||f| = |P^{-T}f| = |a|$, and the corollary implies that $\hat{a}$ is accurate essentially to full machine precision.

The nonnegativity condition on the points α_i in Corollary 4.1 is rather restrictive, since points of both signs are likely to occur in practice. Suppose, then, that we alter the conditions of Corollary 4.1 to allow that

$$\alpha_0 < \cdots < \alpha_m < 0 \leq \alpha_{m+1} < \cdots < \alpha_n, \qquad 0 \leq m \leq n-1. \tag{4.6}$$

The matrices L_i retain the alternating sign pattern, as do $U_{m+1}, \cdots, U_{n-1}$. But $U_0, \cdots, U_m$ lose the sign property, and so there is subtractive cancellation within the product $U_0 \cdots U_{n-1} L_{n-1} \cdots L_0$. It is possible to derive an a priori bound for the effect of this cancellation. For example, we have $U_i \geq 0$ for $0 \leq i \leq m$, and so

$$\begin{aligned} |U_0| \cdots |U_{n-1}||L_{n-1}| \cdots |L_0| &= U_0 \cdots U_m |U_{m+1} \cdots U_{n-1} L_{n-1} \cdots L_0| \\ &= B|B^{-1}BU_{m+1} \cdots U_{n-1} L_{n-1} \cdots L_0| \\ &= B|B^{-1}P^{-T}| \\ &\leq B|B^{-1}||P^{-T}|, \end{aligned} \tag{4.7}$$

where $B = U_0 \cdots U_m$. However, in our experience this inequality is quite weak, and to obtain a manageable bound for the term $B|B^{-1}|$ would produce a substantial further weakening. Therefore we adopt an empirical approach.

For various distributions of distinct points $\alpha_i \in [-1, 1]$, ordered according to (4.6), we evaluated for the monomials, and for the Chebyshev polynomials $T_k(t)$, the ratio

$$q_n = \frac{\| |U_0| \cdots |U_{n-1}||L_{n-1}| \cdots |L_0| \|_\infty}{\|P^{-T}\|_\infty} \geq 1. \tag{4.8}$$

This quantity is a norm-wise analogue of b_n in (4.3); we have taken norms because for points satisfying (4.6) inequality (4.3) can fail to hold for any b_n, since P^{-T} can have a zero element while the lower bound matrix in (4.3) has a nonzero in the same position. Note that q_n can be interpreted as a measure of the sensitivity of the factorisation $P^{-T} = U_0 \cdots U_{n-1} L_{n-1} \cdots L_0$ to perturbations in the factors. Loosely, for a particular

problem we would expect Algorithm 2.1 to perform in a weakly stable manner only if q_n is not too large compared to one.

Values of q_n, together with the condition number $\kappa_\infty(P^T) = \|P^T\|_\infty \|P^{-T}\|_\infty$, are presented for two representative point distributions in Figs. 4.1 and 4.2. The q_n values for the monomials are reasonably small, but suggest some potential instability. More seriously, the results indicate severe instability of Algorithm 2.1 for the Chebyshev polynomials; for example, with $n = 30$ and α_i the extrema of T_n, there is a potential loss of up to 14 figures in solving an almost perfectly conditioned linear system (cf. problem (6.3)). Instability of this magnitude was diagnosed in [12], and a heuristic explanation is given there. The present analysis reveals the source of the problem: the matrix factorisation at the heart of Algorithm 2.1 is, in some cases, an unstable one, in the sense that the product is unduly sensitive to small perturbations in the factors.

If the order of confluency k is positive, and the points are in increasing order, then the alternating sign condition fails to hold for at least one of $L_0, \cdots, L_{k-1}$. A result similar to Corollary 4.1 can be obtained using the technique employed in (4.7). For example, if $k = 1$ then the bound in Corollary 4.1 can be replaced by

$$|\hat{a} - a| \leqq c(n,u)|P^{-T}||M||f|,$$

where $M = |L_0^{-1}||L_0|$ is unit lower triangular and satisfies $|m_{ij}| \leqq 2$.

4.2. Backward stability. We turn now to the residual bound in Theorem 3.2:

$$|f - P^T\hat{a}| \leqq d_n u |L_0^{-1}| \cdots |L_{n-1}^{-1}||U_{n-1}^{-1}| \cdots |U_0^{-1}||\hat{a}| + O(u^2). \tag{4.9}$$

For comparison, if $\tilde{a}$ agrees with a to working precision (e.g., $\tilde{a} = \mathrm{fl}(a)$) then

$$a = \tilde{a} + \delta\tilde{a}, \qquad |\delta\tilde{a}| \leqq u|\tilde{a}|,$$

and so

$$|f - P^T\tilde{a}| = |P^T\delta\tilde{a}| \leqq u|P^T||\tilde{a}|. \tag{4.10}$$

We take (4.10), and the norm-wise version

$$\|f - P^T\tilde{a}\|_\infty \leqq u\|P^T\|_\infty\|\tilde{a}\|_\infty, \tag{4.11}$$

as our model bounds for the residual vector. Connections with the usual notion of backward error are that (4.10) is true if and only if, for some E [16], [17],

$$(P^T + E)\tilde{a} = f, \qquad |E| \leqq u|P^T|, \tag{4.12}$$

and (4.11) implies

$$(P^T + F)\tilde{a} = f, \quad \|F\|_\infty \leqq n^{1/2}u\|P^T\|_\infty \quad (F = (f - P^T\tilde{a})\tilde{a}^T/\tilde{a}^T\tilde{a}). \tag{4.13}$$

Thus (4.10) and (4.11) are equivalent to the condition that $\tilde{a}$ is the solution of a linear system obtained from $P^Ta = f$ by slightly perturbing P^T, in the element-wise sense in (4.12), or the norm-wise sense in (4.13). Note, however, that these perturbed matrices are not, in general, Vandermonde-like matrices.

For the monomials, with distinct, nonnegative points arranged in increasing order, the matrices L_i and U_i are bidiagonal with the alternating sign pattern, as we have seen in § 4.1. Thus $L_i^{-1} \geqq 0$ and $U_i^{-1} \geqq 0$, and since $P^T = L_0^{-1}\cdots L_{n-1}^{-1}U_{n-1}^{-1}\cdots U_0^{-1}$, we obtain from (4.9) the following pleasing backward stability result.

COROLLARY 4.2. *Let* $0 \leqq \alpha_0 < \alpha_1 < \cdots < \alpha_n$, *and consider Algorithm* 2.1 *for the monomials. Under the assumption* (3.10), *the computed solution* $\hat{a}$ *satisfies*

$$|f - P^T\hat{a}| \leqq d_n u |P^T||\hat{a}| + O(u^2),$$

with $d_n = 7n$. □

To investigate the general case (permitting confluency) it is useful to approximate the matrix product in (4.9) by its lower bound in

$$|L|\,|U| := |L_0^{-1}\cdots L_{n-1}^{-1}|\,|U_{n-1}^{-1}\cdots U_0^{-1}| \leqq |L_0^{-1}|\cdots|L_{n-1}^{-1}|\,|U_{n-1}^{-1}|\cdots|U_0^{-1}|,$$

where $P^T = LU$ is an unnormalised LU factorisation. In so doing we make the residual bound smaller and so we are still able to draw conclusions from a large value for the bound. The significance of the approximation is that $|L|\,|U|$ is the matrix that appears in the backward error analysis of Gaussian elimination. For example, from [8] the LU factors $\hat{L}$ and $\hat{U}$ computed by Gaussian elimination without pivoting on $A \in \mathbf{R}^{n\times n}$ satisfy

$$\hat{L}\hat{U} = A + E, \qquad |E| \leqq \frac{nu}{1-nu}\,|\hat{L}|\,|\hat{U}|. \tag{4.14}$$

Using our approximation in the bound (4.9), we obtain

$$|f - P^T\hat{a}| \leqq d_n u\,|L|\,|U|\,|\hat{a}| + O(u^2) \qquad (P^T = LU). \tag{4.15}$$

The similarity of (4.14) and (4.15) suggests that the backward stability of Algorithm 2.1 is related to that of Gaussian elimination without pivoting on P^T. (Note that $|L|\,|U| = |LD|\,|D^{-1}U|$ for any diagonal D, so the normalisation of our LU factorisation is unimportant.) For the same polynomials and points as in Figs. 4.1 and 4.2, Figs. 4.3 and 4.4 show values of

$$g_n = \frac{\||L|\,|U|\|_\infty}{\|P^T\|_\infty} \geqq 1 \qquad (P^T = LU). \tag{4.16}$$

Again, the results predict serious instability of Algorithm 2.1 for the Chebyshev polynomials, and, to a somewhat lesser extent, for the monomials.

5. Preventing and curing instability. Although the increasing ordering for the points α_i yields the favourable stability results in Corollaries 4.1 and 4.2, this ordering is not universally appropriate for Algorithm 2.1, as evidenced by the instability for the Chebyshev

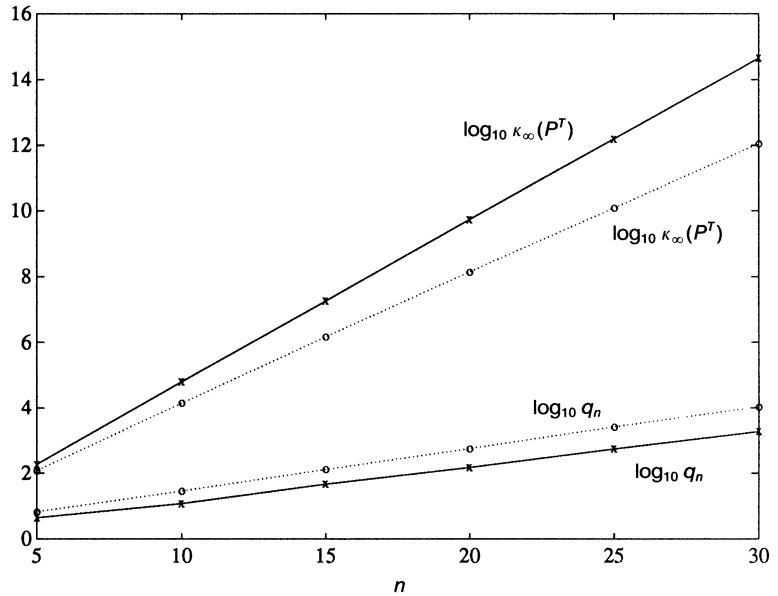

FIG. 4.1. *Monomials.* ×: $\alpha_i = -1 + 2i/n$; ○: $\alpha_{n-i} = \cos(i\pi/n)$ (*extrema of* T_n).

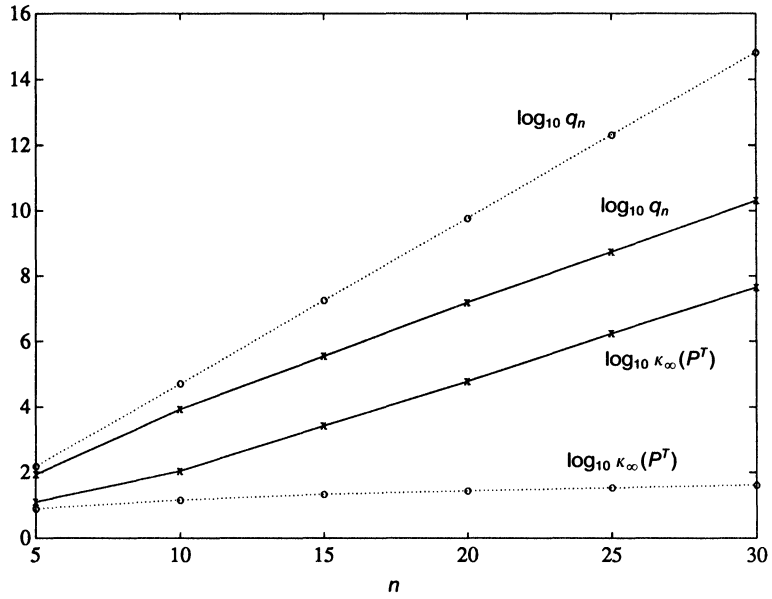

FIG. 4.2. *Chebyshev polynomials.* ×: $\alpha_i = -1 + 2i/n$; ○: $\alpha_{n-i} = \cos(i\pi/n)$ (*extrema of* T_n).

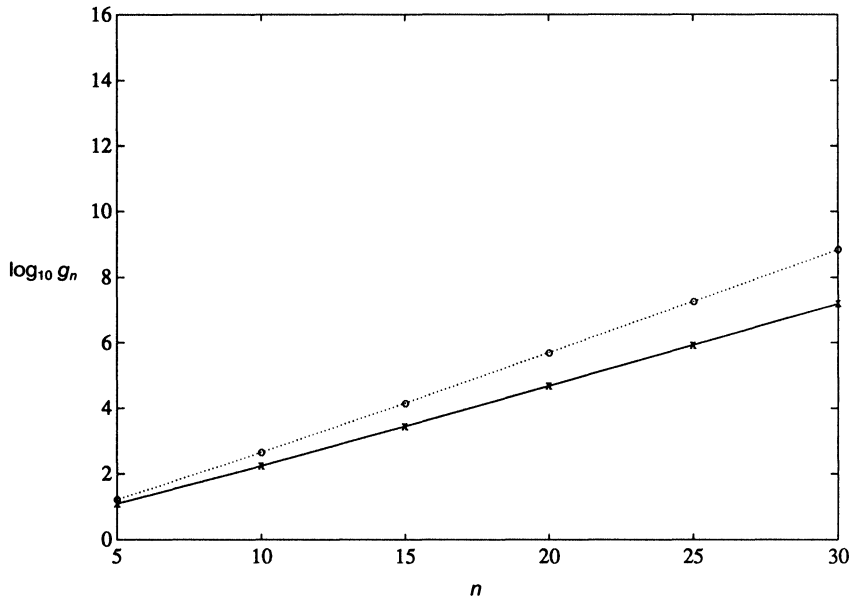

FIG. 4.3. *Monomials.* ×: $\alpha_i = -1 + 2i/n$; ○: $\alpha_{n-i} = \cos(i\pi/n)$ (*extrema of* T_n).

polynomials when there are points of both signs. How, then, in general, can we construct a "good" ordering of the points?

Consider the nonconfluent case. We suggest the following approach that exploits the connection with Gaussian elimination exposed in (4.14) and (4.15). The bound (4.15) suggests that to make Algorithm 2.1 backward stable the points should be ordered

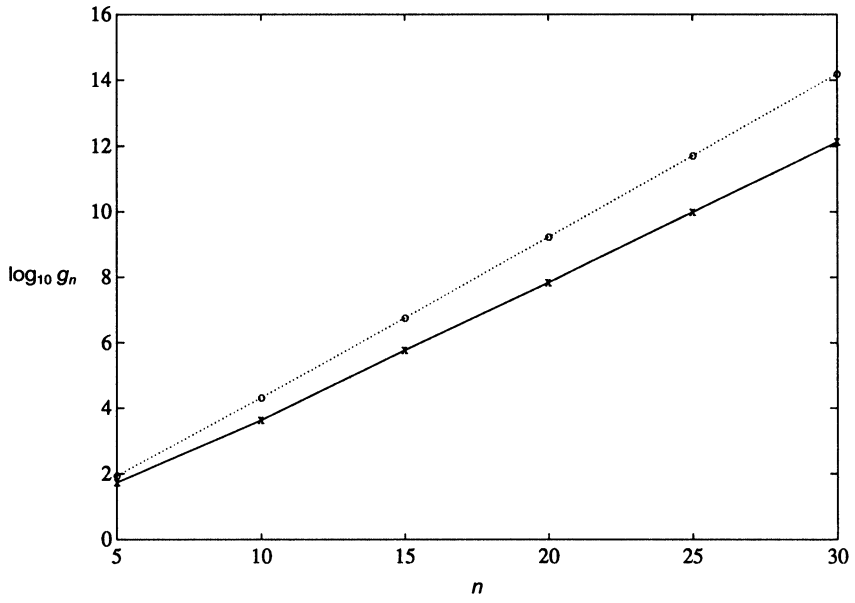

FIG. 4.4. *Chebyshev polynomials*. ×: $\alpha_i = -1 + 2i/n$; ○: $\alpha_{n-i} = \cos(i\pi/n)$ (*extrema of* T_n).

so that g_n in (4.16) is reasonably small. But re-ordering the points is equivalent to permuting the rows of P^T, and as is well known, Gaussian elimination with partial pivoting is a very successful way to obtain a row permutation that keeps g_n small. Now we make the crucial observation that the permutation that would be produced by Gaussian elimination with partial pivoting on P^T can be computed in $O(n^2)$ operations, *without actually performing the elimination*. To see this, recall that $P^T = L_0^{-1} \cdots L_{n-1}^{-1} U_{n-1}^{-1} \cdots U_0^{-1} \equiv LU$, and so if we take L unit lower triangular then (cf. the inverse of (4.4))

$$u_{ii} = h_i \prod_{j=0}^{i-1} (\alpha_i - \alpha_j),$$

where h_i depends only on the θ_i. At the kth stage of Gaussian elimination on P^T the partial pivoting strategy interchanges rows k and r, where $|u_{rk}| = \max_{s \geq k} |u_{sk}|$. Because of the equivalence between interchanges among the rows of P^T and among the points α_i, it follows that r is characterised by

$$\left| \prod_{j=0}^{k-1} (\alpha_r - \alpha_j) \right| = \max_{s \geq k} \left| \prod_{j=0}^{k-1} (\alpha_s - \alpha_j) \right|.$$

This relation forms the basis for the next algorithm.

ALGORITHM 5.1. Given distinct points $\alpha_0, \alpha_1, \cdots, \alpha_n$, this algorithm re-orders the points according to the same permutation that would be produced by Gaussian elimination with partial pivoting applied to $P^T(\alpha_0, \alpha_1, \cdots, \alpha_n)$ (but see below). The permutation is recorded in the vector p.

Swap (α_0, α_j) where $\alpha_j = \min_{i \geq 0} \alpha_i$, $\quad p_0 = j$
Swap (α_1, α_j) where $\alpha_j = \max_{i \geq 1} \alpha_i$, $\quad p_1 = j$
$\pi_i = \alpha_i - \alpha_0 \quad (i = 2, 3, \cdots, n)$

For $k = 2$ to $n - 1$
  $\pi_i = \pi_i * (\alpha_i - \alpha_{k-1})$ $(i = k, \cdots, n)$
  Find j where $\pi_j = \max_{i \geq k} |\pi_i|$
  Swap (α_k, α_j); Swap (π_k, π_j), $p_k = j$
endfor k

Cost. Approximately $n^2/2$ multiplications and comparisons.

In fact, Algorithm 5.1 does slightly more than imitate partial pivoting since it chooses α_0 and α_1, rather than just α_1, to maximise the $(1, 1)$ pivot $\alpha_1 - \alpha_0$. This has the desirable effect of making the output of the algorithm independent of the initial ordering of the points.

If we apply the heuristic that $g_n \approx 1$ for Gaussian elimination with partial pivoting, then from (4.15) we obtain for the ordering of Algorithm 5.1 the approximate residual bound

$$\|f - P^T\hat{a}\|_\infty \leqq d_n u \|P^T\|_\infty \|\hat{a}\|_\infty + O(u^2).$$

Thus, under the several assumptions leading to (4.15), the ordering of Algorithm 5.1 renders Algorithm 2.1 (and similarly Algorithm 2.2) backward stable.

We note that Algorithm 5.1 never produces the increasing ordering, since it sets $\alpha_1 := \max_i \alpha_i$. It is also interesting to note that Algorithm 5.1 is invariant under the linear transformation of the points $\alpha_i := \mu\alpha_i + \lambda$.

An alternative approach to achieving backward stability is to take an arbitrary ordering of the points and to follow Algorithm 2.1 with one step of iterative refinement in single precision. This approach, advocated for general linear equation solvers in [13], was used successfully with the nonconfluent version of Algorithm 2.1, with Chebyshev polynomials, in [12]. However, we have no rigorous forward error bounds or residual bounds for Algorithm 2.1 combined with iterative refinement.

In terms of computational cost the re-ordering strategy is preferable to iterative refinement, since it requires only $5n^2/2$ multiplications in total, compared to the $7n^2$ multiplications required for two invocations of Algorithm 2.1 and a residual vector computation. Moreover, in some applications a sequence of problems with the same, or slightly changed, sequence of points may arise, in which case the re-ordering strategy need be applied only once for the whole sequence.

In the confluent case Algorithm 5.1 can be applied to the *distinct* subset of the points, with groups of equal points interchanged block-wise (since condition (1.1) must be maintained). Note, however, that in this form the algorithm no longer mimics the partial pivoting interchanges, and so the theoretical support is weaker.

6. Numerical experiments. We have carried out a wide variety of numerical experiments to test the analysis of §§ 3–5, and to gain further insight into the behaviour of Algorithm 2.1; we present detailed results for a subset of the tests in this section. The tests were done using Borland Turbo Basic on a PC-AT compatible machine. Turbo Basic uses IEEE-standard single and double precision arithmetic, for which the unit roundoffs are $u_{sp} = 2^{-23} \approx 1.19 \times 10^{-7}$ and $u_{dp} = 2^{-52} \approx 2.22 \times 10^{-16}$, respectively.

We solved each test problem in single precision using each of the following four schemes, which we will refer to by the mnemonics indicated.

(1) Alg: Algorithm 2.1.
(2) Ord: Algorithm 2.1 preceded by Algorithm 5.1.
(3) Sir: Algorithm 2.1 followed by one step of iterative refinement with the residual computed in single precision using Algorithm 2.3.

(4) Gepp: Gaussian elimination with partial pivoting, where P^T is formed in double precision using repeated calls to Algorithm 2.3 (with $x = \alpha_i$, and $a = e_j$ in (2.2)).

In all our test problems the points are in increasing order (of course this is irrelevant for Ord and Gepp). For each computed solution $\hat{a}$ we formed the norm-wise relative error

$$\text{ERR} = \frac{\|\hat{a} - a\|_\infty}{u_{\text{sp}} \|a\|_\infty}$$

and the relative residual

$$\text{RES} = \frac{\|f - P^T\hat{a}\|_\infty}{u_{\text{sp}} \|P^T\|_\infty \|\hat{a}\|_\infty}.$$

Here, $a := \hat{a}_{\text{dp}}$ is the solution computed by Algorithm 2.1 in double precision, and the residual $f - P^T\hat{a}$ is computed using Algorithm 2.3 in double precision. The order n was restricted to ensure that $\hat{a}_{\text{dp}}$ was correct to single precision, thus ensuring a correct value for ERR. Note that ERR and RES are scaled to be "independent of the machine precision"; thus both should be compared with 1 when assessing the accuracy of a computed solution or the size of its residual.

Two further quantities computed were the model bound for ERR, from (4.2),

$$w_n = \frac{\| \, |P^{-T}| \, |f| \, \|_\infty}{\|a\|_\infty},$$

and g_n in (4.16) (for the original, increasing ordering of the points).

The first problem,

$$\text{(6.1)} \qquad \text{Chebyshev polynomials} \quad \alpha_i = \frac{i}{n}, \quad f_i \sim \text{Unif}[-1, 1],$$

illustrates Corollary 4.1 (Unif denotes the uniform random number distribution); see Table 6.1. The excellent accuracy of Algorithm 2.1 is forecast by the corollary since, as is clear from the results, $\| \, |P^{-T}| \, |f| \, \|_\infty \approx \|a\|_\infty$ (a is a large-normed solution). Interestingly, the favourable forward error properties are seen to be lost in the process of iterative refinement, as has been observed in [12].

Next, we consider the monomials on problems with points of both signs. We tried a variety of problems, aiming to generate the instability that the analysis of § 4 predicts may occur for the monomials. In most problems, including all those from [5] and [11], Algorithm 2.1 performed in both a weakly stable and a backward stable manner, yielding

TABLE 6.1
Results for problem (6.1). *All values except n are logs to base* 10.

			ERR				RES					
n	$\kappa_\infty(P^T)$	$\|a\|_\infty$	Alg	Ord	Sir	Gepp	Alg	Ord	Sir	Gepp	w_n	g_n
10	8.6	6.5	−0.2	0.8	5.8	6.7	−1.6	−1.4	−1.5	−1.3	0.9	2.1
15	13.1	11.4	−0.3	0.4	10.3	6.9	−1.3	−1.6	−1.6	−1.3	0.1	3.5
20	17.6	13.9	1.0	1.6	14.7	6.9	−1.4	−1.8	−1.5	−1.7	2.1	5.2
25	22.1	19.6	0.2	0.8	19.3	6.9	−1.7	−1.6	−1.6	−0.9	0.8	6.4

TABLE 6.2
Results for problem (6.2). *All values except n are logs to base* 10.

			ERR				RES					
n	$\kappa_\infty(P^T)$	$\|a\|_\infty$	Alg	Ord	Sir	Gepp	Alg	Ord	Sir	Gepp	w_n	g_n
10	4.8	0.0	1.1	1.8	0.9	−0.3	−0.2	−1.5	−1.7	−3.6	1.3	2.2
15	7.3	0.0	2.2	3.9	2.0	1.2	1.6	−1.1	−1.2	−2.9	2.3	3.4
20	9.7	0.0	3.8	6.4	2.8	1.9	3.1	−1.5	−1.9	−3.1	3.3	4.8
25	12.2	0.0	5.1	9.2	5.5	3.4	4.4	−2.1	−0.9	−3.0	4.4	5.2
30	14.7	0.0	6.1	11.6	9.3	4.4	5.4	−1.8	1.2	−2.9	5.5	6.1

TABLE 6.3
Results for problem (6.3). *All values except n are logs to base* 10.

			ERR				RES					
n	$\kappa_\infty(P^T)$	$\|a\|_\infty$	Alg	Ord	Sir	Gepp	Alg	Ord	Sir	Gepp	w_n	g_n
10	1.0	0.0	3.8	1.1	0.3	0.2	3.5	0.8	0.0	−0.2	0.4	4.2
15	1.2	0.0	6.5	1.0	0.3	0.5	5.9	0.3	0.1	−0.1	0.4	6.6
20	1.3	0.0	8.8	1.7	2.3	0.5	6.4	1.2	1.8	−0.1	0.4	9.1
25	1.4	0.0	10.9	2.1	6.5	1.9	6.5	1.4	5.8	0.1	0.5	10.3

ERR $\leqq w_n$, and RES = $O(1)$. On examining the error analysis we selected the problem

$$\text{monomials} \quad \alpha_i = -1 + \frac{2i}{n}, \quad f = P^T e_n, \tag{6.2}$$

reasoning that $a = e_n$ might "pick out" large elements in the matrix product in (4.9). The results, summarised in Table 6.2, do indeed display instability, principally in the residual, and they match well the predictions of the analysis, as can be seen by comparing the values of RES (for Alg) and g_n.

The problem

$$\text{Chebyshev polynomials} \quad \alpha_{n-i} = \cos\left(\frac{(i+\frac{1}{2})\pi}{n+1}\right), \quad f = P^T e, \tag{6.3}$$

in which the points are the zeros of T_{n+1}, illustrates the instability of Algorithm 2.1 for the Chebyshev polynomials when there are points of both signs; the results are in Table 6.3. The re-ordering strategy successfully stabilises Algorithm 2.1, as does iterative refinement except at $n = 25$ (at this value even using double precision to compute the residual brought no further improvement). Note that because P is well conditioned [10], a small residual implies a small forward error in this problem.

Finally, we present two confluent problems. In these the order of confluency is four and the distinct points $\{\lambda_i\}_{i=0}^{d}$ occur in groups of successive sizes 4, 3, 2, 1, 4, 3, $\cdots$, where the obvious pattern repeats. The two problems are:

(6.4a) monomials
(6.4b) Chebyshev polynomials

$$\Big\} \lambda_{d-i} = \cos\left(\frac{i\pi}{d}\right), \quad i = 0, 1, \cdots, d, \quad f = e_n.$$

In Table 6.4 we see that both iterative refinement and the re-ordering approach behave very unstably on (6.4a) in the sense of weak stability; in our experience this instability

TABLE 6.4
Results for problem (6.4a). *All values except n are logs to base* 10.

			ERR				RES					
n	$\kappa_\infty(P^T)$	$\|a\|_\infty$	Alg	Ord	Sir	Gepp	Alg	Ord	Sir	Gepp	w_n	g_n
9	6.4	−0.3	0.3	3.7	0.3	3.5	−1.1	−0.9	−1.0	−1.2	0.0	1.0
19	12.3	2.3	1.0	6.4	6.1	6.9	−1.7	−1.8	−1.7	−1.5	0.0	3.0
29	17.4	5.7	2.8	10.6	10.9	6.9	−0.6	−2.3	0.4	−1.8	0.0	4.8

TABLE 6.5
Results for problem (6.4b). *All values except n are logs to base* 10.

			ERR				RES					
n	$\kappa_\infty(P^T)$	$\|a\|_\infty$	Alg	Ord	Sir	Gepp	Alg	Ord	Sir	Gepp	w_n	g_n
9	6.6	−0.6	0.5	2.5	0.5	1.8	−3.7	−2.8	−2.5	−2.5	0.0	1.3
19	9.4	−0.9	4.9	4.2	2.1	4.6	−0.8	−2.4	−3.6	−2.3	0.0	4.1
29	11.3	−1.1	9.9	8.2	9.8	5.7	0.7	−0.8	1.0	−2.3	0.0	5.8

is unusual for the latter scheme. Table 6.5 demonstrates clearly that weak stability is not implied by backward stability.

The complete set of test results contain several more features worth noting.

(1) The results for confluent problems were similar in most respects to those for nonconfluent ones; the behaviour of Algorithm 2.1 seems to be minimally affected by confluency. Test results for the Legendre polynomials were very similar in almost every respect to those for the Chebyshev polynomials.

(2) The growth quantity g_n for Gaussian elimination without pivoting is sometimes many orders of magnitude bigger than RES for Alg, but approximate equality can be attained, as in problem (6.2). This behaviour confirms our expectations—see the comment at the end of § 3.

(3) For the monomials our experience is that the forward error from Alg is usually similar to, or smaller than, the forward error from Ord.

(4) Unlike in the tests of [12], in which $u_{sp} \approx 10^{-15}$, we found that iterative refinement in single precision does not always yield a small residual (see Table 6.3, for example). This does not appear to be due to errors in computing the single precision residual via Algorithm 2.3, but seems to indicate that in order to guarantee the success of iterative refinement in single precision a certain level of precision is required relative to the degree of instability (indeed this is implied by the results in [13]).

(5) All our tests support the following heuristic, for which theoretical backing is easily given:

> The computed solution $\hat{x}$ from Gaussian elimination with partial pivoting applied to a linear system $Ax = b$ usually satisfies $\|\hat{x}\|_\infty \leqq u^{-1}\|b\|_\infty / \|A\|_\infty$, where u is the unit roundoff.

Thus, although Gaussian elimination with partial pivoting is guaranteed to produce a small residual, it is unable to solve accurately Vandermonde problems with a very large solution, such as problem (6.1). (Indeed, merely forming the machine matrix $\mathrm{fl}(P^T)$ may be enough to force $\|a\|_\infty \leqq u^{-1}\|f\|_\infty / \|P^T\|_\infty$ for the machine problem!)

7. Conclusions. To conclude, we offer some brief guidelines on the numerical solution of Vandermonde and Vandermonde-like systems. First, we caution that construction of algorithms that involve the solution of a Vandermonde-like system is not generally to be recommended. The tendency for Vandermonde matrices to be extremely ill-conditioned may render such an approach inherently unstable, in the sense that the "ideal" forward error bound (4.2) is unacceptably large; furthermore, as n increases the solution components may soon exceed the largest representable machine number, producing overflow. Despite these problems we have seen that many Vandermonde systems can be solved to surprisingly high accuracy using Algorithms 2.1 and 2.2. A useful rule of thumb is that it is those Vandermonde systems with a large-normed solution—one that reflects the size of P^{-1}—that are solved to high accuracy.

Our experience shows that of the four solution methods considered in § 6 (Alg, Ord, Sir, Gepp), none consistently produces the smallest forward error or the smallest relative residual. Nevertheless, the error analysis and the test results point to some clear recommendations for the choice of solution method. Recall that Alg denotes Algorithm 2.1 (or Algorithm 2.2) with the points arranged in increasing order, and Ord denotes Algorithm 2.1 (or Algorithm 2.2) preceded by Algorithm 5.1.

Monomials. Nonnegative points: Use Alg. In the nonconfluent case Corollaries 4.1 and 4.2 guarantee both weak and backward stability.

Points of both signs: (i) Use Alg. This usually behaves in a weakly stable and a backward stable manner. (ii) If it is vital to obtain a small residual use Ord, perhaps after first trying Alg. Note, however, that the forward error for Ord is usually no smaller, and sometimes larger, than that for Alg (see Tables 6.2 and 6.4).

Other polynomials. Nonnegative points: Use Alg. In the nonconfluent case Corollary 4.1 guarantees weak stability if $\theta_i > 0$, $\beta_i = 0$, and $\gamma_i \geqq 0$ in (2.1), as for the Chebyshev, Legendre, and Hermite polynomials.

Points of both signs: Use Ord (Alg is unstable).

If the points are all nonpositive then in both cases Alg should be used with the points in *decreasing* order (appropriate analogues of Corollaries 4.1 and 4.2 can be derived for this situation).

Acknowledgments. I am grateful to Professor Charles Clenshaw for pointing out reference [18], and to Des Higham for valuable comments on the manuscript.

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] C. T. H. BAKER AND M. S. DERAKHSHAN, *Fast generation of quadrature rules with some special properties*, in Numerical Integration: Recent Developments, Software and Applications, P. Keast and G. Fairweather, eds., D. Reidel, Dordrecht, the Netherlands, 1987, pp. 53–60.

[3] C. BALLESTER AND V. PEREYRA, *On the construction of discrete approximations to linear differential expressions*, Math. Comp., 21 (1967), pp. 297–302.

[4] Å. BJÖRCK AND T. ELFVING, *Algorithms for confluent Vandermonde systems*, Numer. Math., 21 (1973), pp. 130–137.

[5] Å. BJÖRCK AND V. PEREYRA, *Solution of Vandermonde systems of equations*, Math. Comp., 24 (1970), pp. 893–903.

[6] S. D. CONTE AND C. DE BOOR, *Elementary Numerical Analysis*, 3rd ed., McGraw-Hill, Tokyo, 1980.

[7] G. CYBENKO, *The numerical stability of the Levinson–Durbin algorithm for Toeplitz systems of equations*, SIAM J. Sci. Statist. Comput., 1 (1980), pp. 303–319.

[8] C. DE BOOR AND A. PINKUS, *Backward error analysis for totally positive linear systems*, Numer. Math., 27 (1977), pp. 485–490.

[9] W. GAUTSCHI, *On inverses of Vandermonde and confluent Vandermonde matrices*, Numer. Math., 4 (1962), pp. 117–123.

[10] ———, *The condition of Vandermonde-like matrices involving orthogonal polynomials*, Linear Algebra Appl., 52/53 (1983), pp. 293–300.

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

[12] ———, *Fast solution of Vandermonde-like systems involving orthogonal polynomials*, IMA J. Numer. Anal., 8 (1988), pp. 473–486.

[13] M. JANKOWSKI AND H. WOŹNIAKOWSKI, *Iterative refinement implies numerical stability*, BIT, 17 (1977), pp. 303–311.

[14] J. KAUTSKY AND S. ELHAY, *Calculation of the weights of interpolatory quadratures*, Numer. Math., 40 (1982), pp. 407–422.

[15] J. N. LYNESS, *Some quadrature rules for finite trigonometric and related integrals*, in Numerical Integration: Recent Developments, Software and Applications, P. Keast and G. Fairweather, eds., D. Reidel, Dordrecht, the Netherlands, 1987, pp. 17–33.

[16] 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.

[17] R. D. SKEEL, *Scaling for numerical stability in Gaussian elimination*, J. Assoc. Comput. Mach., 26 (1979), pp. 494–526.

[18] F. J. SMITH, *An algorithm for summing orthogonal polynomial series and their derivatives with applications to curve-fitting and interpolation*, Math. Comp., 19 (1965), pp. 33–36.

[19] G. W. STEWART, *Error analysis of the algorithm for shifting the zeros of a polynomial by synthetic division*, Math. Comp., 25 (1971), pp. 135–139.

[20] J. STOER AND R. BULIRSCH, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.

[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 Rev., 8 (1966), pp. 277–301.