IMA Journal of Numerical Analysis (1988) **8**, 473–486

Fast Solution of Vandermonde-Like Systems Involving Orthogonal Polynomials

NICHOLAS J. HIGHAM

Department of Mathematics, University of Manchester, Manchester M13 9PL

Dedicated to Professor Leslie Fox on the occasion of his seventieth birthday

[Received 26 August 1986]

Consider the $(n+1)\times(n+1)$ Vandermonde-like matrix $P=[p_{i-1}(\alpha_{j-1})]$, where the polynomials $p_0(x),\ldots,p_n(x)$ satisfy a three-term recurrence relation. We develop algorithms for solving the primal and dual systems, $P\boldsymbol{x}=\boldsymbol{b}$ and $P^{\mathrm{T}}\boldsymbol{a}=\boldsymbol{f}$ respectively, in $O(n^2)$ arithmetic operations and $O(n)$ elements of storage. These algorithms generalize those of Björck & Pereyra which apply to the monomial case $p_i(x)\equiv x^i$. When the $p_i(x)$ are the Chebyshev polynomials, the algorithms are shown to be numerically unstable. However, it is found empirically that the addition of just one step of iterative refinement is, in single precision, enough to make the algorithms numerically stable.

1. Introduction

GIVEN a set of polynomials $\{p_i(x)\}_{i=0}^n$, where $p_i(x)$ has degree i, and a set of distinct scalars $\alpha_0,\ldots,\alpha_n$, one can define the matrix of order $n+1$:

$$P=P(\alpha_0,\ldots,\alpha_n)=\begin{bmatrix} p_0(\alpha_0) & p_0(\alpha_1)\cdots p_0(\alpha_n)\\ p_1(\alpha_0) & p_1(\alpha_1)\cdots p_1(\alpha_n)\\ \vdots & \vdots \qquad \vdots\\ p_n(\alpha_0) & p_n(\alpha_1)\cdots p_n(\alpha_n)\end{bmatrix}, \tag{1.1}$$

and the associated linear systems

$$P\boldsymbol{x}=\boldsymbol{b}\quad\text{(primal)}, \tag{1.2}$$

$$P^{\mathrm{T}}\boldsymbol{a}=\boldsymbol{f}\quad\text{(dual)}. \tag{1.3}$$

For the monomials $p_i(x)\equiv x^i$, the matrix P is the well-known Vandermonde matrix; the corresponding Vandermonde systems (1.2) and (1.3) arise in a variety of applications [1, 2, 11, 14]. We are interested here in polynomials p_i that satisfy a three-term recurrence relation, and particularly in orthogonal polynomials. A particular application is the solution of certain Chebyshev approximation problems [1, 6], in which (1.2) and (1.3) must be solved with p_i the Chebyshev polynomial of degree i.

The standard method for solving dense systems of linear equations, Gaussian elimination, requires $O(n^3)$ arithmetic operations and $O(n^2)$ elements of storage when applied to (1.2) or (1.3). For the particular case of the monomials, several authors have developed methods that solve the primal and dual Vandermonde systems in only $O(n^2)$ operations and $O(n)$ elements of storage [4, 13]. In Section

2 we derive algorithms which apply to the more general case where the polynomials p_i satisfy a three-term recurrence relation. These algorithms are direct generalizations of those for the monomial case given in [4], and they inherit the property of requiring only $O(n^2)$ operations and $O(n)$ elements of storage.

In Section 3, we examine the numerical stability of the algorithms of Section 2. For the Chebyshev polynomials, the algorithms are shown to be unstable in general, although satisfactory error bounds hold for a certain class of problems.

In Section 4, we consider the use of iterative refinement in single-precision arithmetic as a means of stabilizing the algorithms. Numerical experiments are presented which indicate that just one step of iterative refinement is sufficient to achieve numerical stability.

In the final section, we make recommendations concerning the use of the algorithms of Section 2.

2. Algorithms

Let the polynomials $p_i(x)$ satisfy the three-term recurrence relation

$$p_{j+1}(x) = \theta_j(x-\beta_j)p_j(x) - \gamma_j p_{j-1}(x) \quad (j \geq 1), \tag{2.1}$$

with

$$p_0(x) = 1, \tag{2.2}$$

$$p_1(x) = \theta_0(x-\beta_0)p_0(x), \tag{2.3}$$

where $\theta_j \neq 0$ for all j. Assume the points α_i are distinct and consider the dual system (1.3). Defining

$$\phi(x) = \sum_{i=0}^{n} a_i p_i(x)$$

it follows from (1.3) that ϕ is the unique polynomial of degree n that interpolates to the values f_i at the points α_i. Thus, solving (1.3) is equivalent to obtaining the interpolating polynomial ϕ in terms of the basis $(p_0, \ldots, p_n)$. This representation can be obtained in two stages.

Following [4] we obtain, in the first stage, the divided-difference representation of ϕ:

$$\phi(x) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1} (x-\alpha_j), \qquad c_i = f(\alpha_0, \ldots, \alpha_i), \tag{2.4}$$

where the empty product is defined to be unity. The divided differences c_i may be computed using the standard scheme based on the relation [4]

$$f(\alpha_{j-k-1}, \ldots, \alpha_j) = \frac{f(\alpha_{j-k}, \ldots, \alpha_j) - f(\alpha_{j-k-1}, \ldots, \alpha_{j-1})}{\alpha_j - \alpha_{j-k-1}}.$$

In the second stage, the solution components a_i are generated from the divided differences c_i. Following [4], we define

$$q_n(x) = c_n,$$
$$q_k(x) = (x - \alpha_k)q_{k+1}(x) + c_k \quad (k = n-1, n-2, \ldots, 0), \tag{2.5}$$

from which $q_0(x) = \phi(x)$ (see (2.4)). At this point, we depart from [4] by expressing q_k in terms of the polynomials p_i, rather than the monomials. Let

$$q_k(x) = \sum_{j=0}^{n-k} a_{k+j}^{(k)} p_j(x). \tag{2.6}$$

To obtain recurrences for the coefficients $a_j^{(k)}$, we expand the right-hand side of (2.5), giving

$$q_k(x) = (x - \alpha_k) \sum_{j=0}^{n-k-1} a_{k+j+1}^{(k+1)} p_j(x) + c_k.$$

Using the relations, from (2.1)–(2.3),

$$xp_0(x) = \frac{1}{\theta_0} p_1(x) + \beta_0,$$
$$xp_j(x) = \frac{1}{\theta_j}[p_{j+1}(x) + \gamma_j p_{j-1}(x)] + \beta_j p_j(x) \quad (j \geq 1),$$

we obtain, for $0 \leq k \leq n-2$,

$$\begin{aligned} q_k(x) &= a_{k+1}^{(k+1)}\left(\frac{1}{\theta_0} p_1(x) + \beta_0\right) + \sum_{j=1}^{n-k-1} a_{k+j+1}^{(k+1)}\left(\frac{1}{\theta_j}[p_{j+1}(x) + \gamma_j p_{j-1}(x)] + \beta_j p_j(x)\right) \\ &\qquad - \alpha_k \sum_{j=0}^{n-k-1} a_{k+j+1}^{(k+1)} p_j(x) + c_k \\ &= c_k + (\beta_0 - \alpha_k)a_{k+1}^{(k+1)} + \frac{\gamma_1}{\theta_1} a_{k+2}^{(k+1)} \\ &\quad + \sum_{j=1}^{n-k-2}\left(\frac{1}{\theta_{j-1}} a_{k+j}^{(k+1)} + (\beta_j - \alpha_k)a_{k+j+1}^{(k+1)} + \frac{\gamma_{j+1}}{\theta_{j+1}} a_{k+j+2}^{(k+1)}\right)p_j(x) \\ &\quad + \left(\frac{1}{\theta_{n-k-2}} a_{n-1}^{(k+1)} + (\beta_{n-k-1} - \alpha_k)a_n^{(k+1)}\right)p_{n-k-1}(x) + \frac{1}{\theta_{n-k-1}} a_n^{(k+1)} p_{n-k}(x), \end{aligned} \tag{2.7}$$

in which the empty summation is defined to be zero. For the special case $k = n-1$, we have

$$q_{n-1}(x) = c_{n-1} + (\beta_0 - \alpha_{n-1})a_n^{(n)} + \frac{1}{\theta_0} a_n^{(n)} p_1(x). \tag{2.8}$$

Recurrences for the coefficients $a_j^{(k)}$ $(j = k, \ldots, n)$ in terms of $a_j^{(k+1)}$ $(j = k+1, \ldots, n)$, follow immediately by comparing (2.6) with (2.7) and (2.8).

In detail, the two-stage algorithm is as follows. Here we adopt the convention that, if a value $z_j^{(k)}$ (say) is not defined formally, then $z_j^{(k)} = z_j^{(k-1)}$ (for stage I) or $z_j^{(k)} = z_j^{(k+1)}$ (for stage II).

Algorithm 1. Dual ($P^{\mathrm{T}}\boldsymbol{a}=\boldsymbol{f}$).

```
Stage I   $c_j^{(0)}=f_j \quad (j=0,\ldots,n)$.
          For $k=0$ to $n-1$:
              for $j=n$ to $k+1$ (step $-1$):
                  $c_j^{(k+1)}=(c_j^{(k)}-c_{j-1}^{(k)})/(\alpha_j-\alpha_{j-k-1})$.
Stage II
    $a_j^{(n)}=c_j^{(n)} \quad (j=0,\ldots,n)$,
    $a_{n-1}^{(n-1)}=a_{n-1}^{(n)}+(\beta_0-\alpha_{n-1})a_n^{(n)}$,
    $a_n^{(n-1)}=a_n^{(n)}/\theta_0$.
    For $k=n-2$ to $0$ (step $-1$):
        $a_k^{(k)}=a_k^{(k+1)}+(\beta_0-\alpha_k)a_{k+1}^{(k+1)}+\gamma_1 a_{k+2}^{(k+1)}/\theta_1$;
        for $j=1$ to $n-2-k$:
            $a_{k+j}^{(k)}=a_{k+j}^{(k+1)}/\theta_{j-1}+(\beta_j-\alpha_k)a_{k+j+1}^{(k+1)}+\gamma_{j+1}a_{k+j+2}^{(k+1)}/\theta_{j+1}$;
        $a_{n-1}^{(k)}=a_{n-1}^{(k+1)}/\theta_{n-k-2}+(\beta_{n-k-1}-\alpha_k)a_n^{(k+1)}$,
        $a_n^{(k)}=a_n^{(k+1)}/\theta_{n-k-1}$.
```

The solution is $a_j=a_j^{(0)}$ $(j=0,\ldots,n)$. □

Following the approach used for the (monomial) Vandermonde matrix in [4], we derive an algorithm for solving the primal system (1.2) by expressing Algorithm 1 in matrix–vector notation. Observe that, in stage I of Algorithm 1, the vectors $\boldsymbol{c}^{(k)}=[c_0^{(k)},\ldots,c_n^{(k)}]^{\mathrm{T}}$ are related according to

$$\boldsymbol{c}^{(k+1)}=L_k\boldsymbol{c}^{(k)} \quad (k=0,\ldots,n-1)$$

where L_k is a certain lower bidiagonal matrix (see [4] for the details). Similarly, for the vectors $\boldsymbol{a}^{(k)}=[a_0^{(k)},\ldots,a_n^{(k)}]^{\mathrm{T}}$ in stage II,

$$\boldsymbol{a}^{(k)}=U_k\boldsymbol{a}^{(k+1)} \quad (k=n-1,\ldots,0),$$

where U_k is upper triangular with nonzeros on the diagonal and the first two superdiagonals only.

Thus

$$P^{-\mathrm{T}}\boldsymbol{f}=\boldsymbol{a}=\boldsymbol{a}^{(0)}=U_0\cdots U_{n-1}\boldsymbol{c}^{(n)}=U_0\cdots U_{n-1}L_{n-1}\cdots L_0\boldsymbol{f},$$

so that $P^{-\mathrm{T}}=(U_0\cdots U_{n-1})(L_{n-1}\cdots L_0)\equiv UL$ (which is a UL decomposition of $P^{-\mathrm{T}}$, in factored form). Hence, the solution to the primal system is given by

$$\boldsymbol{x}=P^{-1}\boldsymbol{b}=L_0^{\mathrm{T}}\cdots L_{n-1}^{\mathrm{T}}U_{n-1}^{\mathrm{T}}\cdots U_0^{\mathrm{T}}\boldsymbol{b}.$$

Writing out this equation in detail, we obtain the following algorithm.

Algorithm 2. Primal ($P\boldsymbol{x}=\boldsymbol{b}$).

```
Stage I   $d_j^{(0)}=b_j \quad (j=0,\ldots,n)$.
```

```
For $k = 0$ to $n - 2$:
    $d_{k+1}^{(k+1)} = (\beta_0 - \alpha_k)d_k^{(k)} + d_{k+1}^{(k)}/\theta_0$;
    for $j = 2$ to $n - k$:
        $d_{k+j}^{(k+1)} = \gamma_{j-1}d_{k+j-2}^{(k)}/\theta_{j-1} + (\beta_{j-1} - \alpha_k)d_{k+j-1}^{(k)} + d_{k+j}^{(k)}/\theta_{j-1}$.
$d_n^{(n)} = (\beta_0 - \alpha_{n-1})d_{n-1}^{(n-1)} + d_n^{(n-1)}/\theta_0$.
```

Stage II

```
$x_j^{(n)} = d_j^{(n)}$   $(j = 0, \ldots, n)$.
For $k = n - 1$ to 0 (step $-1$):
    $x_k^{(k+\frac{1}{2})} = x_k^{(k)}$;
    for $j = k + 1$ to $n$:
        $x_j^{(k+\frac{1}{2})} = x_j^{(k+1)}/(\alpha_j - \alpha_{j-k-1})$;
    for $j = k + 1$ to $n$:
        $x_j^{(k)} = x_j^{(k+\frac{1}{2})} - x_{j+1}^{(k+\frac{1}{2})}$;
    $x_n^{(k)} = x_n^{(k+\frac{1}{2})}$.
```

The solution is $x_j = x_j^{(0)}$ $(j = 0, \ldots, n)$. □

Polynomials of interest in Algorithms 1 and 2 include the following.

Monomials: $$\theta_j \equiv 1, \quad \beta_j = \gamma_j \equiv 0. \tag{2.9}$$

Chebyshev: $$\theta_0 = 1, \quad \beta_0 = 0; \qquad \theta_j = 2, \quad \beta_j = 0, \quad \gamma_j = 1 \qquad (j \geq 1). \tag{2.10}$$

Legendre (with the normalization $p_j(1) = 1$ for all j):

$$\theta_j = (2j + 1)/(j + 1), \quad \beta_j = 0, \quad \gamma_j = j/(j + 1). \tag{2.11}$$

Hermite: $$\theta_j = 2, \quad \beta_j = 0, \quad \gamma_j = 2j. \tag{2.12}$$

Laguerre: $$\theta_j = -1/(j + 1), \quad \beta_j = 2j + 1, \quad \gamma_j = j/(j + 1). \tag{2.13}$$

Of course, for the monomials, Algorithms 1 and 2 reduce to the algorithms in [4].

The operation counts for Algorithms 1 and 2 are the same, by construction. Notice that, in both algorithms, γ_j appears only in the terms γ_j/θ_j. Assuming that the values γ_j/θ_j are given, then the operation count is $n(2n + 1)M + \frac{1}{2}n(5n + 3)$ A, where A denotes an addition or subtraction and M is a multiplication or division.

Note that the algorithms can be implemented in such a way that the right-hand-side vector is transformed into the solution vector, without using any extra storage.

3. Numerical stability

In this section, we examine the behaviour of Algorithms 1 and 2 in finite-precision arithmetic. Consider Algorithm 1. It is easy to show, inductively, that

$$a_j^{(0)} = \sum_{i=j}^{\min\{j+2k,n\}} \mu_{ji}^{(k)} a_i^{(k)} \quad (1 \leq k \leq n,\ 0 \leq j \leq n), \tag{3.1}$$

where the coefficients $\mu_{ji}^{(k)}$ depend on the points α_i and on the parameters θ_i, β_i, γ_i. Let $\hat{a}_i^{(k)} \approx a_i^{(k)}$ denote the values computed in finite-precision arithmetic with unit roundoff u [8, p. 32]. Corresponding to (3.1), we have

$$\hat{a}_j^{(0)} = \sum_{i=j}^{\min\{j+2k,n\}} \mu_{ji}^{(k)} \hat{a}_i^{(k)} + R_j^{(k)} \quad (1 \leq k \leq n,\ 0 \leq j \leq n), \tag{3.2}$$

where $R_j^{(k)} = O(u)$, whose precise form is unimportant, collects together terms from a forward rounding-error analysis. Write

$$\hat{a}_i^{(k)} = a_i^{(k)}(1 + \varepsilon_i^{(k)}),$$

where, almost certainly, the relative error $\varepsilon_i^{(k)}$ will satisfy $|\varepsilon_i^{(k)}| \geq u$. Subtracting (3.1) from (3.2) gives

$$\hat{a}_j^{(0)} - a_j^{(0)} = \sum_{i=j}^{\min\{j+2k,n\}} \mu_{ji}^{(k)} a_i^{(k)} \varepsilon_i^{(k)} + R_j^{(k)} \quad (1 \leq k \leq n,\ 0 \leq j \leq n). \tag{3.3}$$

Thus, unless severe cancellation takes place in the expressions (3.3), it must hold that

$$|\hat{a}_j^{(0)} - a_j^{(0)}| \gtrsim u \max_{1 \leq k \leq n} \max_{j \leq i \leq \min\{j+2k,n\}} |\mu_{ji}^{(k)} a_i^{(k)}|. \tag{3.4}$$

It follows that, if any of the intermediate quantities $|a_i^{(k)}|$ in Algorithm 1 is large relative to some final $|a_j^{(0)}|$ $(i \geq j)$, or to $\|\boldsymbol{a}\|$, then there is the possibility of large relative errors $|\hat{a}_j^{(0)} - a_j^{(0)}|/|a_j^{(0)}|$ and $\|\hat{\boldsymbol{a}} - \boldsymbol{a}\|/\|\boldsymbol{a}\|$ respectively. The following example shows that large relative errors can indeed be observed, even on a very well conditioned problem.

Consider the Chebyshev polynomial dual system defined by

$$\left.\begin{aligned} p_i(x) &= \mathrm{T}_i(x) = \cos(i \cos^{-1} x) \\ \alpha_i &= \cos(i\pi/n), \qquad f_i = (-1)^i \end{aligned}\right\} \quad (i = 0, \ldots, n). \tag{3.5}$$

The points α_i are the extrema of $\mathrm{T}_n(x)$, and it can be shown that

(a) $\kappa_2(P) \leq 2$,

(b) $a_n^{(n)} = f(\alpha_0, \ldots, \alpha_n) = 2^{n-1}$ [5, p. 237],

(c) $a = [0, 0, \ldots, 0, 1]^{\mathrm{T}}$,

where $\kappa_2(P) = \|P\|_2 \|P^{-1}\|_2$ is the condition number in the 2-norm.

The heuristic analysis above suggests that, for this example, the relative error $\|\hat{\boldsymbol{a}} - \boldsymbol{a}\|_2/\|\boldsymbol{a}\|_2$ will be of order at least $2^{n-1}u$. We tested this prediction numerically, using the computing environment to be described in Section 4 ($u \approx 10^{-15}$). The results are summarized in Table 3.1. For $n = 20$ and $n = 30$, the relative errors observed are actually somewhat larger than is predicted by (3.4) (on the assumption that $|\mu_{ji}^{(k)}| = O(1)$). Since the coefficient matrix P in this example is very well-conditioned, any stable algorithm for the solution, such as

TABLE 3.1
Results for problem (3.5) ($\|\boldsymbol{a}\|_2 = 1$)

n	$a_n^{(n)}$	$u \max_{i,k} \lvert a_i^{(k)}\rvert$	$\|\hat{\boldsymbol{a}} - \boldsymbol{a}\|_2/\|\boldsymbol{a}\|_2$
5	1·6 E1	1·3 E −13	2·3 E −13
10	5·1 E2	1·6 E −11	7·5 E −11
20	5·2 E5	2·1 E −7	1·7 E −5
30	5·4 E8	2·4 E −3	3·6

Gaussian elimination with partial pivoting, would yield a computed solution accurate (in the norm sense) almost to full machine precision. In contrast, for $n = 30$, Algorithm 1 provides a computed solution with no correct significant figures.

We conclude that Algorithm 1 is numerically unstable, for the Chebyshev polynomials at least; the same conclusion applies to Algorithm 2 because of the underlying duality. The situation is unclear for other choices of the parameters θ_j, β_j, and γ_j in Algorithms 1 and 2, unless specific examples of instability are produced, or a detailed error analysis is performed—one which preferably would apply to arbitrary θ_j, β_j, and γ_j.

A forward error analysis of the monomial versions of Algorithms 1 and 2 is presented in [9], for the case where the points α_i satisfy

$$0 \leq \alpha_0 < \alpha_1 < \cdots < \alpha_n. \tag{3.6}$$

We state without proof the following extension of Theorem 2.3 in [9].

If the conditions (3.6) and

$$\beta_j = 0, \quad \theta_j > 0, \quad \gamma_j \geq 0 \qquad \text{for all } j \tag{3.7}$$

hold, then the computed solutions $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{x}}$ from Algorithms 1 and 2 respectively satisfy the bounds

$$|\hat{\boldsymbol{a}} - \boldsymbol{a}| \leq q_n u \, |P^{-\mathrm{T}}| \, |\boldsymbol{f}| + O(u^2), \tag{3.8}$$

$$|\hat{\boldsymbol{x}} - \boldsymbol{x}| \leq q_n u \, |P^{-1}| \, |\boldsymbol{b}| + O(u^2), \tag{3.9}$$

where q_n is a small constant and $|X|$ denotes the vector or matrix whose elements are respectively the absolute values of the elements of the vector or matrix X.

It follows (see [9: Section 4]) that, when conditions (3.6) and (3.7) are satisfied, Algorithms 1 and 2 introduce no more uncertainty into the numerical solution than was already present if the machine right-hand-side vector were subject to relative errors of the order of the unit roundoff. Thus, Algorithms 1 and 2 are guaranteed to perform satisfactorily, being numerically stable in a weak but relevant sense, whenever conditions (3.6) and (3.7) are satisfied. We note, however, that the nonnegativity condition in (3.6) is likely to be violated in problems involving orthogonal polynomials whose natural range includes part of the negative real axis (such as the polynomials in (2.10)–(2.12)); an example of a problem to which the analysis is not applicable is (3.5). Therefore we cannot draw from this analysis any overall conclusions about the stability of Algorithms 1 and 2.

In the next section we consider a practical approach to alleviating the potential instabilities in Algorithms 1 and 2.

4. Iterative refinement in single precision

A standard way to improve the performance of a linear-equation solver is to combine it with iterative refinement. Given the computed solution $\hat{\boldsymbol{x}}$ to $A\boldsymbol{x} = \boldsymbol{b}$, the refinement procedure takes the form

```
$x \leftarrow \hat{x}$
Repeat
1. Compute $r := b - Ax$.
2. Solve $Ay = r$.
3. Update $x \leftarrow x + y$.
```

This technique is perhaps most commonly applied to Gaussian elimination with partial pivoting [8: p. 74]. Here, the refinement stage is relatively inexpensive compared to the initial factorization stage; and it is well known that, if residuals are computed in double precision, then iterative refinement will converge to a solution accurate to full machine precision, as long as A is not too ill conditioned [8: p. 75].

Iterative refinement can also be used with Algorithms 1 and 2. However, in this case, steps 1 and 2 of the refinement procedure both require approximately the same amount of work as computation of the original solution. Further, the residual vector cannot be computed in a straightforward and efficient manner using double-precision accumulation of inner products, since one would first have to compute the elements of P in (1.1) explicitly (because they are not part of the problem data). Therefore, for Algorithms 1 and 2, there is little point in using iterative refinement with residuals computed in double precision; one may as well apply the basic algorithm just once, using double-precision arithmetic throughout.

We are led to consider using iterative refinement with residuals computed in single precision, an approach which has been used successfully by several authors [3, 10, 12, 13]. Although iterative refinement in single precision clearly cannot be expected to produce solutions accurate to full machine precision, it can improve greatly the stability properties of the method to which it is applied.

Jankowski & Wozniakowski [10] carry out a detailed error analysis of iterative refinement in single precision. Their analysis applies to any method for solving linear systems $A\boldsymbol{x} = \boldsymbol{b}$ that satisfies the following condition: if A is not too ill conditioned, then the computed solution $\hat{\boldsymbol{x}}$ has relative error bounded in some norm by $q < 1$, that is,

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}\| / \|\boldsymbol{x}\| \leq q < 1. \tag{4.1}$$

It is shown in [10] that any such method combined with iterative refinement in single precision is numerically stable whenever $q\kappa(A) < 1$, where $\kappa(A) = \|A\| \, \|A^{-1}\|$ is the condition number. By *numerically stable* is meant that the computed solution $\bar{x}$ satisfies

$$\|b - A\bar{x}\| \leq c_n u \, \|A\| \, \|x\|, \tag{4.2}$$

where c_n is a constant depending only on n, the order of A.

Remark. Our terminology differs from that in [10], where a method is called *well-behaved* if (4.2) is satisfied, and *numerically stable* if a weaker condition is satisfied.

It is easy to show [8: p. 16] that (4.2) implies the existence of a matrix E and a constant c_n' for which

$$(A + E)\bar{x} = b \quad \text{and} \quad \|E\| \leq c_n' u \, \|A\|, \tag{4.3}$$

assuming that $c_n u\kappa(A) < \frac{1}{2}$. Similarly, (4.3) implies (4.2) if $c_n' u\kappa(A) < \frac{1}{2}$. Thus a numerically stable method is one for which the computed solution is the true solution to a 'nearby problem'.

Note that the following definition of stability is equivalent to (4.2) when (4.1) is satisfied:

$$\|b - A\bar{x}\| \leq c_n'' u \, \|A\| \, \|\bar{x}\| \quad \text{for some constant } c_n''. \tag{4.4}$$

We prefer to work with (4.4) because, for the 2-norm, it is unconditionally equivalent to (4.3) (with $c_n' = c_n''$) [8: p. 16].

Unfortunately, for the Chebyshev polynomials at least, Algorithms 1 and 2 do not satisfy condition (4.1). This is shown by the example of Section 3, where A is very well conditioned, but where, for $n = 30$, we must take $q > 1$ (see Table 3.1). Although the results of [10] are thus not strictly applicable here, they do provide motivation for investigating the behaviour of iterative refinement in single precision experimentally.

We have carried out some experiments with the Chebyshev-polynomial version of Algorithm 1, using one step of iterative refinement. The computations were performed in Fortran 77 on a CDC computer with unit roundoff $u = 2^{-48} \approx 3.55 \times 10^{-15}$. The residuals

$$r_i = f_i - \sum_{j=0}^{n} \mathrm{T}_j(\alpha_i)a_j \quad (i = 0, \ldots, n)$$

were computed in single precision using the nested-multiplication algorithm for orthogonal polynomials [5: p. 257]. In the general case, computation of r in this way costs approximately $3n^2(M + A)$ (recall that Algorithm 1 requires approximately $2n^2M + \frac{5}{2}n^2A$).

For each computed solution, we evaluated the quantities

$$\text{ERR} = \frac{\|\bar{a} - a\|_2}{u \, \|a\|_2}, \qquad \text{RES} = \frac{\|f - T^{\mathrm{T}}\bar{a}\|_2}{u \, \|\bar{a}\|_2},$$

where $\bar{a}$ denotes the original or the corrected solution, and $T = [\mathrm{T}_{i-1}(\alpha_{j-1})] \in \mathbb{R}^{(n+1)\times(n+1)}$ denotes the 'Chebyshev Vandermonde'. Here the exact solution a to the machine problem is approximated by the solution computed using Algorithm 1 in double precision. Note that numerical stability as defined in (4.4) corresponds to RES $\leq d_n$, for some constant d_n (since $\|T\|_2 \leq \|T\|_F \leq n + 1$).

We also computed, for the original solution only, the quantity

$$\text{EST} = \max_{i,k} |a_i^{(k)}| / \|a\|_\infty,$$

which may be interpreted as a 'growth factor' for stage II of Algorithm 1, and which the analysis of Section 3 suggests should provide an approximate lower bound for the relative error, ERR.

Twelve test problems were solved, comprising all combinations of the point distributions

A1: $\alpha_{n-i} = \cos(i\pi/n)$, (extrema of $T_n(x)$)

A2: $\alpha_{n-i} = \cos[(i+\frac{1}{2})\pi/(n+1)]$, (zeros of $T_{n+1}(x)$)

A3: $\alpha_i = -1 + 2i/n$,

A4: $\alpha_i = i/n$,

and the right-hand sides

F1: $f_i = (-1)^i$,

F2: $f = [1, 0, \dots, 0]^T$,

F3: $f_i = 1/(1 + 25\alpha_i^2)$,

The conditioning of the Chebyshev Vandermonde for these four point distributions is summarized as follows. First, it can be shown that

A1: $\kappa_2(T) \leq 2$ for all n,

A2: $\kappa_2(T) = 2$ for all n (cf. [7]).

Second, the following values for $\kappa_2(T)$ were obtained by computing the singular-value decomposition (in single precision).

n:	5	10	20	30
A3	2·9	23·7	8·6 E3	5·1 E6
A4	4·2 E3	1·0 E8	5·4 E14	2·0 E15

It can be seen that T is singular to working precision for the distribution A4 when $n = 30$.

We present a representative selection of the results in Tables 4.1–4.5. Note that problem A1/F1 (Table 4.1) is the same as problem (3.5).

TABLE 4.1
Problem A1/F1

			Original		Corrected	
n	$\|a\|_\infty$	EST	ERR	RES	ERR	RES
5	1·0	3·8 E1	6·4 E1	1·1 E2	2·5	5·1
10	1·0	4·6 E3	2·1 E4	5·7 E4	9·8	3·1 E1
20	1·0	5·9 E7	4·7 E9	1·8 E10	2·7 E1	1·3 E2
30	1·0	6·9 E11	1·0 E15	1·2 E15	1·3 E2	5·3 E2

TABLE 4.2
Problem A2/F2

			Original		Corrected	
n	$\|a\|_\infty$	EST	ERR	RES	ERR	RES
5	0·32	1·9 E1	7·0 E1	1·4 E2	2·1	2·4
10	0·18	1·5 E3	2·1 E3	5·1 E3	1·8 E1	4·1 E1
20	9·5 E −2	1·4 E7	4·3 E8	1·5 E9	2·8 E1	9·0 E1
30	6·4 E −2	1·4 E11	9·1 E12	3·7 E13	1·2 E2	4·6 E2

TABLE 4.3
Problem A3/F1

			Original		Corrected	
n	$\|a\|_\infty$	EST	ERR	RES	ERR	RES
5	1·6	2·9 E1	4·8 E1	9·3 E1	4·0	9·1
10	9·3	1·3 E3	1·3 E3	3·5 E3	1·0 E1	3·0 E1
20	2·8 E3	3·3 E5	1·7 E6	5·0 E6	2·2 E3	6·0 E1
30	1·4 E6	6·5 E7	1·3 E10	3·8 E10	1·7 E5	3·6 E1

TABLE 4.4
Problem A4/F1

			Original		Corrected	
n	$\|a\|_\infty$	EST	ERR	RES	ERR	RES
5	2·0 E3	1·1	2·5	0·66	4·2 E2	1·0
10	3·5 E7	1·1	1·9	1·9	6·7 E5	1·3
20	2·0 E16	1·0	1·4	1·7	2·0 E14	8·7
30	1·4 E25	1·0	1·3	4·8	2·7 E24	2·9

TABLE 4.5
Problem A4/F3

			Original		Corrected	
n	$\|a\|_\infty$	EST	ERR	RES	ERR	RES
5	1·1 E1	1·3	1·1 E1	0·97	4·1 E2	0·77
10	3·1 E3	1·1	8·2	3·0	2·2 E6	1·6
20	3·8 E8	1·0	5·3 E5	1·6	2·1 E13	2·1
30	1·1 E13	1·0	3·6 E9	3·6	6·8 E23	4·0

We make several comments on the results.

(1) Severe numerical instability is observed in problems A1/F1, A2/F2, and A3/F1, as evidenced by the large relative residuals (RES $\gg$ 1) in Tables 4.1–4.3. It is clear that the example in Section 3 is not an isolated 'bad case'.

(2) In all the problems (including those not shown in the tables), the relative residual after one step of iterative refinement in single precision is of an

acceptable size; the refinement stage reduces the residual by many orders of magnitude in several cases.

(3) In Tables 4.1–4.3, a consequence of the reduction in the residuals brought about by the refinement stage is a reduction of the error: in norm terms, the error is bounded by the product of the condition number and the residual, and the condition number is very small for the point distributions A1 and A2, as noted above.

(4) For the problems in Tables 4.4 and 4.5, the points α_i satisfy (3.6) and so the bound (3.8) is applicable. For the problem A4/F1, it can be shown that $|T^{-T}|\,|f| = |a|$, so that (3.8) takes the form

$$|\hat{a} - a| \leq q_n u\,|a| + O(u^2),$$

which implies very small relative errors, as indeed are observed in Table 4.4 (ERR ≈ 1). Although iterative refinement does not worsen the already small residuals, it clearly has a disastrous effect on the error in Tables 4.4 and 4.5! This behaviour is explained by the fact that, in these two examples, the computed residual for the original solution has few (if any) correct digits, due to cancellation; consequently, Algorithm 1 solves (however accurately) the 'wrong' system, producing a correction consisting mainly of noise.

(5) The approximate lower bound EST for the relative error provides reasonable order-of-magnitude estimates, except in Table 4.5, where the bounds are far too small. However, it is interesting to note that, in all cases, EST is of the same order of magnitude as the relative residual RES.

For the test problems above, we also investigated the effect of performing more than one step of iterative refinement (in single precision). The second and subsequent refinement steps were found to bring about little or no reduction in the size of the residual or the error.

We mention briefly a further experiment motivated by [1], wherein the dual Chebyshev Vandermonde system is solved by converting it into a monomial Vandermonde system, using the relationship

$$T = LV, \qquad V = [\alpha_{j-1}^{i-1}],$$

where L is a lower triangular matrix of the form illustrated

$$L = \begin{bmatrix} 1 & & & & \\ 0 & 1 & & & \\ -1 & 0 & 2 & & \\ 0 & -3 & 0 & 4 & \\ 1 & 0 & -8 & 0 & 8 \end{bmatrix}.$$

We wanted to see whether this approach circumvents the instability of the Chebyshev-polynomial version of Algorithm 1. On the same test problems as above, we found that the errors and residuals were generally of the same order of magnitude as those for the direct-solution approach. When one step of iterative refinement was used with both methods, the conversion approach generally gave larger errors and residuals. These results suggest that, when using Algorithm 1, it is preferable to solve a Chebyshev Vandermonde system directly, rather than to convert to monomial form.

5. Conclusions

Based on our analysis and numerical experiments, we recommend that Algorithm 1 be used in the following way, for all choices of the parameters (similarly for Algorithm 2).

(1) Obtain the computed solution $\hat{\boldsymbol{a}}$ from Algorithm 1.

(2) Compute, in single precision, the residual vector $\boldsymbol{r}=\boldsymbol{f}-P^{T}\hat{\boldsymbol{a}}$. If $\|\boldsymbol{r}\|_2 \approx u\,\|P\|_2\,\|\hat{\boldsymbol{a}}\|_2$ then accept $\hat{\boldsymbol{a}}$ (here $\|P\|_2$ can be approximated by $\|P\|_F$, for example).

(3) Otherwise, apply one step of iterative refinement, using the $\boldsymbol{r}$ from step (2), and accept the corrected solution.

It is important to terminate in step (2) if the residual is relatively small, because not to do so risks losing the very favourable accuracy properties which hold for certain classes of problem (see (3.8) and Tables 4.4–4.5).

Finally, we note that, for the original monomial versions of Algorithms 1 and 2, there have been no reports of unstable behaviour, despite wide experience with the algorithms [1, 4, 9, 13, 14]. It would be interesting to investigate to what extent the instability observed in the Chebyshev versions of the algorithms is prevalent for other choices of the parameters. An error analysis encompassing all classes of method and problem, not just those defined by (3.6) and (3.7), is desirable in order to answer this question fully.

Acknowledgements

I thank Dr T. L. Freeman and and Dr I. Gladwell for their valuable comments on the manuscript.

References

1. Almacany, M., Dunham, C. B., & Williams, J. 1984 Discrete Chebyshev approximation by interpolating rationals. *IMA J. Numer. Anal.* **4,** 467–477.
2. Ballester, C., & Pereyra, V. 1967 On the construction of discrete approximations to linear differential expressions. *Math. Comp.* **21,** 297–302.
3. Björck, Å. 1987 Stability analysis of the method of seminormal equations for linear least squares problems. *Linear Algebra & Applics* **88/89,** 31–48.
4. Björck, Å., & Pereyra, V. 1970 Solution of Vandermonde systems of equations. *Math. Comp.* **24,** pp. 893–903.
5. Conte, S. D., & de Boor, C. 1980 *Elementary Numerical Analysis* (Third edition). McGraw-Hill, Tokyo.
6. Dunham, C. B. 1982 Choice of basis for Chebyshev approximation, *ACM Trans. Math. Soft.* **8,** 21–25.
7. Gautschi, W. 1983 The condition of Vandermonde-like matrices involving orthogonal polynomials. *Linear Algebra and Applics* **52/53,** 293–300.
8. Golub, G. H. & Van Loan, C. F. 1983 Matrix Computations. Johns Hopkins University Press, Baltimore, Maryland.

9. HIGHAM, N. J. 1987 Error analysis of the Björck–Pereyra algorithms for solving Vandermonde systems. *Numer. Math.* **50,** 613–632.
10. JANKOWSKI, M., & WOZNIAKOWSKI, H. 1977 Iterative refinement implies numerical stability. *BIT* **17,** 303–311.
11. LYNESS, J. N. & MOLER, C. B. 1966 Van der Monde systems and numerical differentiation. *Numer. Math.* **8,** 458–464.
12. SKEEL, R. D. 1980 Iterative refinement implies numerical stability for Gaussian elimination. *Math. Comp.* **35,** 817–832.
13. TANG, W. P., & GOLUB, G. H. 1981 The block decomposition of a Vandermonde matrix and its applications. BIT **21,** 505–517.
14. TRAPP, G. E., & SQUIRE, W. 1975 Solving nonlinear Vandermonde systems. *Comput. J.* **18,** 373–374.