SIAM J. SCI. STATIST. COMPUT.
Vol. 0, No. 0, pp. xxx–xxx, Month 1992

012

STABLE PARALLEL ALGORITHMS FOR TWO-POINT BOUNDARY VALUE PROBLEMS*

STEPHEN J. WRIGHT†

Abstract. Some of the most widely used algorithms for two-point boundary value ODEs, namely finite difference and collocation methods and standard multiple shooting, proceed by setting up and solving a structured system of linear equations. It is well known that the linear system can be set up efficiently in parallel; we show here that a structured orthogonal factorization technique can be used to solve this system, and hence the overall problem, in an efficient, parallel, and stable way.

Key words. parallel algorithms, two-point boundary value problems, error analysis and stability

AMS(MOS) subject classifications. 65F05, 65G05, 65L10, 65L20, 65W05

1. Introduction.

Many numerical algorithms for solving the linear two-point boundary value problem

$$y' = M(t)y + q(t), \qquad t \in [a,b], \qquad y \in R^n, \tag{1}$$

$$B_a y(a) + B_b y(b) = d, \tag{2}$$

have been proposed and studied over the last 30 years. Many of these methods require a structured linear algebraic system (for example, a block-tridiagonal or staircase system) to be solved as a "core" operation, and considerable effort has been devoted to minimizing the amount of computer time and storage required during the factorization of the coefficient matrix of this system. Efficient factorization schemes, based on structured Gaussian elimination, have been implemented and are widely available (see §2, and Varah [19], Diaz, Fairweather and Keast [7], Lentini and Pereyra [13], and Keller [9].)

During the last 10 years, the question of stability of algorithms for (1),(2) has received a great deal of attention (see, for example, Mattheij [15].) It has been recognized that in a well-conditioned problem (that is, one whose solution is not too sensitive to perturbations in M, q or the boundary conditions (2)), the fundamental solution space generally contains both exponentially increasing and exponentially decreasing modes. The stability of a numerical method for (1),(2) depends on its ability to at some point perform a "decoupling" of these modes. For the standard multiple shooting and finite difference algorithms, this decoupling is performed implicitly, during the factorization of the structured linear system, through the use of a pivoting strategy that prevents element growth in the factors. Unfortunately, parallel and vectorizable algorithms that have been proposed for solving the linear system invariably place some restriction on the choice of pivots. This can lead to undesirable element growth in the factors, and such methods are in fact similar to compactification algorithms for (1),(2) which are known to be unstable. In this paper, we use instead a structured orthogonal factorization technique which is stable, has an identical serial complexity to the best-known algorithms, and can be efficiently implemented on a wide variety of parallel architectures. A variant of the algorithm vectorizes efficiently,

* Received by the editors September xx, 1990; accepted for publication (in revised form) January xx, 1991. This research was supported by the Applied Mathematical Sciences subprogram of the Office of Energy research, United States Department of Energy, contract W-31-109-Eng-38. A grant of computer time at the North Carolina Supercomputing Center is gratefully acknowledged.

† Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439.

in much the same way as cyclic reduction for block-tridiagonal linear systems. No separability of the boundary conditions is needed. Although we focus on matrices arising from multiple shooting and one-step differencing schemes, the technique can be applied equally well to the more general staircase matrix structures which arise in other numerical schemes, such as collocation.

We assume throughout that n is too small to allow efficient parallel or vector implementation of order-n matrix and vector operations. Instead, parallelism is sought by partitioning the domain $[a,b]$ of the independent variable.

The remainder of the paper is organized as follows: in §2 we review the multiple shooting and finite difference algorithms, the structured linear systems that result from them, and the efficient Gaussian elimination techniques used to solve them on serial computers. In §3 we briefly review previous work on parallel and vectorizable algorithms. The new algorithm is presented in §4, together with a stability result and analysis of serial and parallel complexity. In §5 we describe parallel implementation of a standard technique for estimating the condition number of the coefficient matrix, which can be used for purposes of *a posteriori* error analysis. Finally in §6 we describe the results of implementation of the scheme on shared-memory and vector architectures.

2. Serial algorithms. The "standard" multiple shooting technique for (1),(2) proceeds by defining a mesh

$$a = t_1 < t_2 < \ldots < t_{k+1} = b, \tag{3}$$

and finding a fundamental and particular solution on each interval of the mesh:

$$Y_i' = M(t)Y_i, \qquad Y_i(t) \in R^{n\times n}, \qquad t \in [t_i, t_{i+1}], \qquad Y_i(t_i) = I,$$

$$y_{pi}' = M(t)y_{pi} + q(t), \qquad y_{pi}(t) \in R^{n}, \qquad t \in [t_i, t_{i+1}], \qquad y_{pi}(t_i) = 0.$$

Then, we try to find $s_i \in R^n$, $i = 1, \ldots, k+1$, such that

$$y(t) = Y_i(t)s_i + y_{pi}(t), \qquad t \in [t_i, t_{i+1}], \quad i = 1, \ldots, k.$$

(Note in particular that $s_i = y(t_i)$, $i = 1, \ldots, k+1$.) By applying the boundary conditions (2), and continuity at the meshpoints, we obtain the following linear system in $s_1, \ldots, s_{k+1}$:

$$\begin{bmatrix} B_a & & & & B_b \\ Y_1(t_2) & -I & & & \\ & Y_2(t_3) & -I & & \\ & & \ddots & \ddots & \\ & & & Y_k(t_{k+1}) & -I \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_{k+1} \end{bmatrix} = \begin{bmatrix} d \\ -y_{p1}(t_2) \\ -y_{p2}(t_3) \\ \vdots \\ -y_{pk}(t_{k+1}) \end{bmatrix}. \tag{4}$$

In the (quite common) case in which the boundary conditions are separated, this system can be rearranged into a block-banded form. If p is the number of initial conditions and q is the number of final conditions ($p+q=n$), we can assume without loss of generality that the boundary data (2) can be partitioned as follows:

$$B_a = \begin{bmatrix} \bar{B}_a \\ 0 \end{bmatrix}, \quad B_b = \begin{bmatrix} 0 \\ \bar{B}_b \end{bmatrix}, \quad d = \begin{bmatrix} d_a \\ d_b \end{bmatrix}.$$

If the rows $p+1,\ldots,n$ of (4) are moved to the bottom, we obtain the form

$$\begin{bmatrix} \bar{B}_a & & & & & \\ Y_1(t_2) & -I & & & & \\ & Y_2(t_3) & -I & & & \\ & & \ddots & \ddots & & \\ & & & Y_k(t_{k+1}) & -I & \\ & & & & \bar{B}_b \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_k \\ s_{k+1} \end{bmatrix} = \begin{bmatrix} d_a \\ -y_{p1}(t_2) \\ -y_{p2}(t_3) \\ \vdots \\ -y_{pk}(t_{k+1}) \\ d_b \end{bmatrix}. \tag{5}$$

One-step finite difference methods proceed by again choosing a mesh of the form (3) and seeking $s_1,\ldots,s_{k+1}$ to approximate $y(t_1),\ldots,y(t_{k+1})$. On the interval $[t_i,t_{i+1}]$, (1) is approximated by the linear relationship

$$\frac{s_{i+1}-s_i}{h_i} = \bar{A}_i s_i + \bar{C}_i s_{i+1} + \bar{f}_i, \tag{6}$$

where $h_i = t_{i+1}-t_i$. Two schemes with a local truncation error of $\mathrm{O}(h_i^2)$ are the "box scheme" ($\bar{A}_i = \bar{C}_i = \frac{1}{2}M(t_{i+\frac{1}{2}})$, $\bar{f}_i = q(t_{i+\frac{1}{2}})$) and the trapezoidal rule ($\bar{A}_i = \frac{1}{2}M(t_i)$, $\bar{C}_i = \frac{1}{2}M(t_{i+1})$, $\bar{f}_i = \frac{1}{2}(q(t_i)+q(t_{i+1}))$.) By including (2), we obtain a linear system of the form

$$\begin{bmatrix} B_a & & & & B_b \\ A_1 & C_1 & & & \\ & A_2 & C_2 & & \\ & & \ddots & \ddots & \\ & & & A_k & C_k \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_{k+1} \end{bmatrix} = \begin{bmatrix} d \\ f_1 \\ f_2 \\ \vdots \\ f_k \end{bmatrix}, \tag{7}$$

where $A_i = -I - h_i\bar{A}_i$, $C_i = I - h_i\bar{C}_i$, and $f_i = h_i\bar{f}_i$. If the boundary conditions are separated, we obtain

$$\begin{bmatrix} \bar{B}_a & & & & \\ A_1 & C_1 & & & \\ & A_2 & C_2 & & \\ & & \ddots & \ddots & \\ & & & A_k & C_k \\ & & & & \bar{B}_b \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_k \\ s_{k+1} \end{bmatrix} = \begin{bmatrix} d_a \\ f_1 \\ f_2 \\ \vdots \\ f_k \\ d_b \end{bmatrix}. \tag{8}$$

The accuracy of finite difference schemes is often enhanced by the use of deferred correction techniques (see, for example, Pereyra [18]).

As has been observed, the two algorithms are closely related, in the sense that for a reasonable choice of the approximation (6), $-C_i^{-1}A_i$ should be close to $Y_i(t_{i+1})$. Hence, the conditioning of the matrices in (4) and (7) is quite similar when the h_i are small. If we quantify the conditioning of the problem (1),(2) by a bound κ on its Green's function (see Ascher, Mattheij, and Russell [3, §3.2]), it has been shown by Osborne [16] (and also by Mattheij [14] and Lentini, Osborne, and Russell [12]) that the inverse of A_S from (4) satisfies the bound

$$\|A_S^{-1}\|_\infty \le k\kappa.$$

Hence if we define γ by

$$\gamma = 1 + \max_{i=1,\ldots,k} \|Y_i(t_{i+1})\|_\infty$$

and assume that B_a and B_b are scaled such that $\|[B_a, B_b]\|_\infty = 1$, then from (4),(5),

$$\text{cond}_\infty(A_S) \leq \gamma k \kappa. \tag{9}$$

For the finite-difference coefficient matrix A_D in (7), a similar analysis applies if we note that $\gamma = 2 + O(\bar{h})$ for small $\bar{h} = \max_{i=1,\ldots,k} h_i$. We use this fact, together with $A_D = A_S + O(\bar{h})$, to derive the bound

$$\text{cond}_\infty(A_D) \leq (2 + O(\bar{h}))k\kappa. \tag{10}$$

The slightly different form of the bounds (9) and (10) is motivated by the fact that k is typically larger in a finite difference algorithm than in a multiple shooting algorithm. Bounds for the coefficient matrices in (5) and (8) are, of course, identical to their non-separated counterparts.

As stated earlier, Gaussian elimination algorithms with various forms of pivoting have been previously proposed for solving (4),(5),(7),(8). The practical stability of such algorithms for general matrices is well known, but is also well known that the worst-case behavior can be very bad, as a result of possible growth of elements in the factors which is exponential in the dimension $(k+1)n$ of the system. It has been shown (see, for example, Mattheij [15]) that the presence of an exponential dichotomy in (1),(2) ensures that this worst-case behavior does not arise when elimination algorithms are applied to the matrices in (4),(5),(7),(8). In the partially separated cases (5),(8), similar results can be proved, without referring to the dichotomy at all, by using the results of Bohte [4]. Bohte shows that for banded linear systems, the bound on element growth in partial pivoting algorithms is exponential only in the bandwidth.

In the simplest case of Gaussian elimination with row partial pivoting (with coefficient matrices and right-hand sides denoted by A_{SP} and f_{SP} in (5) and A_{DP} and f_{DP} in (8)), possible fill-in of kpn elements can occur in the upper triangular factor. Element growth is, however, bounded by 2^{2n-1}, and so we obtain the following theorem:

THEOREM 2.1. *Let $s = (s_1^T, \ldots, s_{k+1}^T)^T$ denote the true solution of the system (8), and suppose that $\hat{s}$ is the approximate solution obtained by Gaussian elimination with row partial pivoting. Then provided that*

(i) $\bar{h}$ is chosen small enough that $\text{cond}_\infty(A_{DP}) \leq 4k\kappa$,

(ii) $8c_1\kappa\mathbf{u} \leq 1$, where $c_1 = (1.12)2^{2n}n^3k(k+1)(k+8) = O(k^3n^3 2^{2n})$ and $\mathbf{u}$ is the unit roundoff error,

the following relative error bound applies:

$$\frac{\|\hat{s} - s\|_\infty}{\|s\|_\infty} \leq 16c_1\kappa\mathbf{u}. \tag{11}$$

Suppose that the fundamental and particular solutions that are used to construct A_{SP} and f_{SP} in (5) are calculated to a tolerance of τ, that is,

$$\begin{aligned} \|A_{SP} - \bar{A}_{SP}\|_\infty &\leq \tau\|A_{SP}\|_\infty \\ \|f_{SP} - \bar{f}_{SP}\|_\infty &\leq \tau\|f_{SP}\|_\infty. \end{aligned} \tag{12}$$

Let $\tilde{s}$ be the solution obtained by using Gaussian elimination with row partial pivoting. Then provided that

(iii) $\tau k\kappa\gamma \leq 1/2$,

(iv) $6c_1\mathbf{u}\kappa\gamma \leq 1$,

we have

$$\frac{\max_{i=1,\ldots,k+1} \|\tilde{s}_i - y(t_i)\|_\infty}{\max_{i=1,\ldots,k+1} \|y(t_i)\|_\infty} \leq 4\gamma\kappa[k\tau + 3c_1\mathbf{u} + c_2\mathbf{u}\tau], \tag{13}$$

where $c_2 = 12c_1k\kappa\gamma$.

Proof. For the first part of the theorem, the proof follows from Theorem 2.7.2 in Golub and Van Loan [8], and §4 of Bohte [4]. The latter assumes that

$$\mathbf{u} \leq .009 \qquad \text{and} \qquad (k+1)n\mathbf{u} \leq 0.1,$$

which are clearly implied by assumption (ii), since $\kappa \geq 1$, $k \geq 1$ and $n \geq 1$. Bohte then shows that $\hat{s}$ solves the system

$$(A_{DP} + E)\hat{s} = f_{DP},$$

where E satisfies the bound

$$\|E\|_\infty \leq \Lambda\mathbf{u}(0.56)(4n-1)n(k+1)(kn+7n+3),$$

and Λ is a bound on the maximal element which arises during the elimination. By simplifying this expression, and noting from the discussion above that

$$\Lambda \leq 2^{2n-1} \max_{i,j} |(A_{DP})_{ij}| \leq 2^{2n-1}\|A_{DP}\|_\infty,$$

we have that

$$\|E\|_\infty \leq (c_1/k)\mathbf{u}\|A_{DP}\|_\infty.$$

The result now follows by setting $r = 1/2$ in [8, Theorem 2.7.2].

For the second part, note first that

$$\|\bar{A}_{SP}\|_\infty \leq (1+\tau)\|A_{SP}\|_\infty$$

and

$$\|\bar{A}_{SP}^{-1}\|_\infty \leq \|[I + A_{SP}^{-1}(\bar{A}_{SP} - A_{SP})]^{-1}\|_\infty \|A_{SP}^{-1}\|_\infty.$$

Provided

$$\|A_{SP}^{-1}(\bar{A}_{SP} - A_{SP})\|_\infty \leq 1, \tag{14}$$

we have from [8, Lemma 2.3.3] that

$$\|\bar{A}_{SP}^{-1}\|_\infty \leq \frac{\|A_{SP}^{-1}\|_\infty}{1 - \|A_{SP}^{-1}\|_\infty\tau} \leq \frac{k\kappa}{1-\tau k\kappa} \leq 2k\kappa.$$

The inequality (14) is implied by (iii). Since k, γ and κ are all at least 1, assumption (iii) implies that $\tau \leq 1/2$, and so

$$\text{cond}_\infty(\bar{A}_{SP}) \leq 2(1+\tau)k\kappa\gamma \leq 3k\kappa\gamma. \tag{15}$$

Now let $\bar{s}$ be the exact solution of $\bar{A}_{SP}\bar{s} = \bar{f}_{SP}$. Direct application of [8, Theorem 2.7.2] with $r = 1/2$ shows, again using (iii), that

$$\frac{\max_{i=1,\ldots,k+1} \|\bar{s}_i - y(t_i)\|_\infty}{\max_{i=1,\ldots,k+1} \|y(t_i)\|_\infty} \leq 4\text{cond}_\infty(A_{SP})\tau \leq 4k\kappa\gamma\tau. \tag{16}$$

Application of Bohte's results shows that the computed solution $\tilde{s}$ satisfies $(\bar{A}_{SP} + E)\tilde{s} = \bar{f}_{SP}$, where

$$\|E\|_\infty \leq \frac{c_1}{k}\mathbf{u}\|\bar{A}_{SP}\|_\infty.$$

Assumption (iv) and (15) imply that

$$\frac{c_1}{k}\mathbf{u}\text{cond}_\infty(\bar{A}_{SP}) \leq 1/2,$$

and so

$$\frac{\|\tilde{s} - \bar{s}\|_\infty}{\|\bar{s}\|_\infty} \leq \frac{4c_1\mathbf{u}}{k}\text{cond}_\infty(\bar{A}_{SP}) \leq 12c_1\mathbf{u}\kappa\gamma. \tag{17}$$

The result follows by combining (16) and (17) in an elementary way. ∎

A more economical scheme for solving (5) and (8), described by Varah [19] and implemented by Diaz, Fairweather, and Keast [7], is the method of alternate row and column elimination. For the first p stages of the process, we use *column* pivoting and elimination (involving columns 1 through n); this produces no fill-in. For stages $p+1$ through n we use *row* pivoting and elimination for the same reason. Column and row elimination alternate in this way until a factorization of the form

$$A_{DP} = PLBU\Pi \tag{18}$$

is produced, where P and Π are permutation matrices, L and U are lower and upper triangular matrices of multipliers, and B has the form

$$B = \begin{bmatrix} L & & & & & & & & \\ X & R & X & X & & & & & \\ X & & L & & & & & & \\ & & X & R & X & X & & & \\ & & X & & L & & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & & X & 0 & L & \\ & & & & & & & & X & R \end{bmatrix}.$$

(L denotes a lower triangular $p\times p$ block, R denotes an upper triangular $(n-p)\times(n-p)$ block, and X denotes a dense block.) It is easy to show that element growth in the B factor is bounded by 2^{n-1} and, hence, that the scheme is very stable. We can prove the following:

THEOREM 2.2. *Let $s = (s_1^T, \ldots, s_{k+1}^T)^T$ denote the true solution of the system (8), and let $\hat{s}$ be the approximate solution calculated by alternate row and column elimination. Suppose further that*

(i) $\bar{h}$ is chosen small enough that $\text{cond}_\infty(A_{DP}) \leq 4k\kappa$, *and*

(ii) $8c_4\kappa\mathbf{u} \leq 1$,

where

$$c_4 = k(k+2)n(5+6n)(1+n^2 2^{n-1}) = O(k^2 n^4 2^n).$$

Then

$$\frac{\|\hat{s} - s\|_\infty}{\|s\|_\infty} \leq 16c_4\kappa\mathbf{u}.$$

Let $\tilde{s}$ be obtained by multiple shooting with alternate row and column elimination, and suppose that (12) holds, with

(iii) $\tau k \kappa \gamma \le 1/2$,
(iv) $6 c_4 \mathbf{u} \kappa \gamma \le 1$.
Then

$$\frac{\max_{i=1,\ldots,k+1} \|\tilde{s}_i - y(t_i)\|_\infty}{\max_{i=1,\ldots,k+1} \|y(t_i)\|_\infty} \le \gamma k \kappa [k\tau + 3c_4\mathbf{u} + c_5 \mathbf{u}\tau],$$

where $c_5 = 12 c_4 k \kappa \gamma$.

Proof. The assumptions of Lemma A.1 are consequences of (*ii*) and (*iv*) above. The first part of the result follows directly from Lemma A.2 and [8, Theorem 2.7.2]. The second part is analogous to the second part of Theorem 2.1, with c_4 replacing c_1 and c_5 replacing c_2. ∎

3. Parallel elimination algorithms. Other parallel and vectorizable algorithms, based on Gaussian elimination, have recently been proposed for (4),(5),(7),(8). In general, they suffer either from poor stability properties or from limitations in the amount of parallelism which is possible.

Wright and Pereyra [21] describe variants of a block factorization algorithm applied to (7). In the most highly vectorized variant, a factorization of the form

$$\begin{bmatrix} \tilde{A}_1 & & & & \\ & \tilde{A}_2 & & & \\ & & \ddots & & \\ & & & \tilde{A}_k & \\ Z_1 & Z_2 & \ldots & Z_k & \tilde{A}_{k+1} \end{bmatrix} \begin{bmatrix} I & W_1 & & & \\ & I & W_2 & & \\ & & \ddots & \ddots & \\ & & & I & W_k \\ & & & & I \end{bmatrix} =$$

$$\begin{bmatrix} P_1 & & & & \\ & P_2 & & & \\ & & \ddots & & \\ & & & P_k & \\ & & & & P_{k+1} \end{bmatrix} \begin{bmatrix} A_1 & C_1 & & & \\ & A_2 & C_2 & & \\ & & \ddots & \ddots & \\ & & & A_k & C_k \\ B_a & & & & B_b \end{bmatrix}$$

(where the P_i are permutation matrices) is produced. It is easy to show that for $\bar{h}$ sufficiently small, this factorization exists. It is equivalent to compactification [3, page 153] which, because of its similarity to single shooting, is known to be potentially unstable. However, this instability can usually be recognized by the presence of large elements in the Z_i blocks. The strategy described in [21, §5] is to use this factorization where possible, and discard it in favor of a more stable method if the $\|Z_i\|$ are too large. In many applications (such as the one described in [21]) the lack of stability is not a problem.

Paprzycki and Gladwell [17] describe a partitioned elimination algorithm in which (8) is torn into P submatrices, each of which has the same form as the original A_{DP}, and alternate row and column elimination is applied to each piece. This corresponds to partitioning the interval $[a, b]$ and solving a number of sub-BVPs, each of which has p initial and q final conditions. Although the number of initial and final conditions is correct, this alone is not enough to ensure well-conditioning of the sub-BVPs. In a linear algebra sense, well-conditioning of the whole matrix A_{DP} does not guarantee well-conditioning of each of the P submatrices. Ascher and Mattheij [2] develop a "theoretical multiple shooting" framework in which they show how boundary values for the sub-BVPs should be chosen to ensure well-conditioning. Ascher and Chan [1] suggest how to implement this in a parallel environment.

Another possibility, which leads to near-perfect speedup on two processors (but cannot be generalized to a higher order of parallelism) is the "burn at both ends" or "twisted" factorization. Here, some form of pivoted Gaussian elimination is applied simultaneously from both ends of the matrix (either A_{DP} or A_{SP}). When the factorizations meet in the center, a small reduced system is formed and factored. This is analogous to the approach of Lentini [11].

Finally, we mention the approach of Ascher and Chan [1], who form the normal equations for (5) and (8), and factorize the resulting symmetric, positive definite, block-tridiagonal system using cyclic reduction. In exact arithmetic, this scheme produces a triangular factor which is identical to that given by the "cyclic reduction" variant of the algorithm to be described in the next section. The difference lies in the fact that by explicitly forming the normal equations, the condition number of a linear system is squared, an effect that is avoided when the algorithm from §4 is used.

4. Structured orthogonal factorization. We now describe the structured QR factorization algorithm, as applied to the system (7). It can of course be applied equally well to the systems (8),(4),(5), since it is indifferent to separability of the boundary conditions.

The first step is to partition the system into, say, P pieces of approximately equal size. We choose indices $k_1, k_2, \ldots, k_{P+1}$ to satisfy

$$0 = k_1 < k_2 < \ldots < k_{P+1} = k, \qquad k_{j+1} \geq k_j + 2, \;\; j = 1, \ldots, P.$$

(P could for instance be the number of processors on the physical machine being used to solve the problem.) Partition j then consists of rows $(k_j+1)n+1, \ldots, (k_{j+1}+1)n$ of (7). Each partition is now processed independently; in effect, the variables s_i for $i \neq k_j + 1$, $j = 0, \ldots, P$, are eliminated from the problem.

We describe this process in detail for the first partition, which, if augmented with its right-hand side, has the form

$$\left[\begin{array}{ccccc|c} A_1 & C_1 & & & & f_1 \\ & A_2 & C_2 & & & f_2 \\ & & \ddots & \ddots & & \vdots \\ & & & A_{k_2} & C_{k_2} & f_{k_2} \end{array}\right]. \tag{19}$$

We first find an orthogonal matrix $Q_1 \in R^{2n \times 2n}$ such that

$$Q_1^T \left[\begin{array}{c} C_1 \\ A_2 \end{array}\right] = \left[\begin{array}{c} R_1 \\ 0 \end{array}\right],$$

where $R_1 \in R^{n \times n}$ is upper triangular. If we form Q_1 as a product of n Householder transformations, the information needed to reconstruct Q_1 could be stored in the space formerly occupied by the "zeroed" elements of $[C_1^T, A_2^T]^T$, plus an additional n locations. We also need to apply Q_1^T to the other columns of the matrix (19), and the overall effect is

$$\left[\begin{array}{cc} Q_1^T & 0 \\ 0 & I \end{array}\right] \left[\begin{array}{ccccc|c} A_1 & C_1 & & & & f_1 \\ & A_2 & C_2 & & & f_2 \\ & & \ddots & \ddots & & \vdots \\ & & & A_{k_2} & C_{k_2} & f_{k_2} \end{array}\right] =$$

$$(20)\qquad \left[\begin{array}{ccccccc|c} G_1 & R_1 & E_1 & & & & & g_1 \\ \bar{G}_2 & 0 & \bar{C}_2 & & & & & \bar{f}_2 \\ & & A_3 & C_3 & & & & f_3 \\ & & & \ddots & \ddots & & & \vdots \\ & & & & & A_{k_2} & C_{k_2} & f_{k_2} \end{array}\right].$$

The next step is to find an orthogonal $Q_2 \in R^{2n\times 2n}$ such that

$$Q_2^T \left[\begin{array}{c} \bar{C}_2 \\ A_3 \end{array}\right] = \left[\begin{array}{c} R_2 \\ 0 \end{array}\right],$$

and to apply Q_2^T to rows $n+1,\ldots,3n$ of the reduced system (20). This process is repeated a total of $k_2 - 1$ times, until finally we obtain a system equivalent to (19) which has the form

$$(21)\qquad \left[\begin{array}{ccccccc|c} G_1 & R_1 & E_1 & & & & & g_1 \\ G_2 & & R_2 & E_2 & & & & g_2 \\ \vdots & & & \ddots & \ddots & & & \vdots \\ G_{k_2-1} & & & & R_{k_2-1} & E_{k_2-1} & & g_{k_2-1} \\ \tilde{A}_1 & & & & & \tilde{C}_1 & & \tilde{f}_1 \end{array}\right].$$

Formally, the reduction process for partition j can be specified as follows:

$$\bar{C}_{k_j+1} = C_{k_j+1},\quad \bar{G}_{k_j+1} = A_{k_j+1},\quad \bar{f}_{k_j+1} = f_{k_j+1}.$$

for $i = k_j + 1,\ldots,k_{j+1} - 1$

Find orthogonal Q_i such that

$$(22)\qquad Q_i^T \left[\begin{array}{c} \bar{C}_i \\ A_{i+1} \end{array}\right] = \left[\begin{array}{c} R_i \\ 0 \end{array}\right],$$

where $R_i \in R^{n\times n}$ is upper triangular;

Set

$$\left[\begin{array}{c} E_i \\ \bar{C}_{i+1} \end{array}\right] \leftarrow Q_i^T \left[\begin{array}{c} 0 \\ C_{i+1} \end{array}\right], \qquad \left[\begin{array}{c} G_i \\ \bar{G}_{i+1} \end{array}\right] \leftarrow Q_i^T \left[\begin{array}{c} \bar{G}_i \\ 0 \end{array}\right],$$

$$\left[\begin{array}{c} g_i \\ \bar{f}_{i+1} \end{array}\right] \leftarrow Q_i^T \left[\begin{array}{c} \bar{f}_i \\ f_{i+1} \end{array}\right].$$

end (for)

$$\text{Set } \tilde{A}_j = \bar{G}_{k_{j+1}},\qquad \tilde{C}_j = \bar{C}_{k_{j+1}},\qquad \tilde{f}_j = \bar{f}_{k_{j+1}}.$$

Clearly, if we knew the values of s_{k_j+1} and $s_{k_{j+1}+1}$, the values of $s_{k_j+2},\ldots,s_{k_{j+1}}$ could be recovered by the simple back-substitution:

for $i = k_{j+1} - 1, k_{j+1} - 2,\ldots,k_j + 1$

$$(23)\qquad \text{Solve } R_i s_{i+1} = g_i - G_i s_{k_j} - E_i s_{i+2}$$

In fact, these "separator" variables can be found by forming and solving a reduced system by taking the last n rows of each (processed) partition, and the boundary conditions. This reduced system has the form

$$\begin{bmatrix} B_a & & & & B_b \\ \tilde{A}_1 & \tilde{C}_1 & & & \\ & \tilde{A}_2 & \tilde{C}_2 & & \\ & & \ddots & \ddots & \\ & & & \tilde{A}_P & \tilde{C}_P \end{bmatrix} \begin{bmatrix} s_{k_1+1} \\ s_{k_2+1} \\ s_{k_3+1} \\ \vdots \\ s_{k_{P+1}+1} \end{bmatrix} = \begin{bmatrix} d \\ \tilde{f}_1 \\ \tilde{f}_2 \\ \vdots \\ \tilde{f}_P \end{bmatrix}. \tag{24}$$

The form of (24) is obviously identical to the form of the original system (7), so this immediately suggests that it may be possible to apply the whole process recursively, that is, (24) could be partitioned into $P_2 \leq P/2$ pieces, and the algorithm described above could be applied to obtain a smaller reduced system. This process could be repeated for, say, L levels, so at the innermost level a $(P_L + 1)n$-dimensional system would remain.

In a parallel implementation of the algorithm, the number of levels L to be used and the number of processors to be used at each level depend on the number of available processors on the physical machine, and, on a distributed-memory machine, on the cost of interprocessor communication. The "extreme" cases are as follows:

- A one-level version ($P = 1, k_2 = k$), in which (7) is reduced to the $2n \times 2n$ system

$$\begin{bmatrix} B_a & B_b \\ \tilde{A}_1 & \tilde{C}_1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_{k+1} \end{bmatrix} = \begin{bmatrix} d \\ \tilde{f}_1 \end{bmatrix}, \tag{25}$$

  which is then solved by QR factorization. This is the "serial" version of the algorithm.
- A two-level version, in which P processors are used to do the reduction and back-substitution (23), and the system (24) is solved by the one-level algorithm just described. It is easy to see that, assuming that each partition contains about k/P rows of blocks, the time required for the reduction phase is proportional to $(k/P - 1)n^3$. The time required to solve (24) on a single processor is proportional to $(P + 1)n^3$. The latter operation will not be a bottleneck for the whole process, provided that $k \gg P^2$.
- A "cyclic reduction" version, in which P is equal to the number of available processors (assuming that $P \leq k/2$), and then $P_2 = P/2$, $P_3 = P/4$, ..., $P_L = 1$. Here clearly $L = \log_2 P + 1$. The total computation time required will be proportional to $(k/P - 1 + \log_2 P)n^3$, while the communication time will be proportional to $n^2 \log_2 P$. With regard to complexity, the algorithm is optimal: when $k = P \log_2 P$, the execution time is $O(\log_2 P)$ on P processors, while the serial time is $O(P \log_2 P)$.

The cyclic reduction variant is also appropriate for implementation on a single *vector* processor. In this environment, we could choose $P = k/2, P_2 = k/4, P_3 = k/8, \ldots$, and so $L = \log_2 k + 1$. At level l, the reduction and back-substitution processes can be coded so that most of the arithmetic involves vectors of length $k/2^l$. At low levels, a reversal of the loop nesting can be used to ensure that vector lengths do not fall below n.

It is not difficult to see that the factorization and solution phases can be separated, provided that the Householder vectors are stored (in the locations vacated by the

zeroed elements). This feature is useful when quasi-linearization is used to solve a first-order nonlinear BVP. The same coefficient matrix (and its factorization) can be used for a number of consecutive iterations to produce "chord method" approximations to Newton steps.

It is important to note that the scheme proposed above is simply standard Householder QR factorization applied to an initial row- and column-permuted form of the original matrix. Thus, we can apply the standard stability analysis for $(k+1)n$-dimensional matrices from Lawson and Hanson [10] to obtain error bounds for the computed solutions. We have refined these bounds to take into account the structure of the matrix. In addition, to allay any possible concerns about instability at the level of the $O(\mathbf{u}^2)$ terms, we have removed these terms using the style of analysis in Wilkinson [20]. The relevant results are stated and proved in Appendix B. Here, we summarize the analysis in the following Lemma and Theorem:

LEMMA 4.1. *Let $s = (s_1^T, s_2^T, \ldots, s_{k+1}^T)^T$ denote the true solution of the system (7), and suppose that $\hat{s}$ is the approximate solution obtained by using the structured QR factorization and back-substitution. Suppose that the orthogonal matrices used in the factorization are all constructed from Householder reflectors, and that n, k and $\mathbf{u}$ satisfy*

$$(12n+51)n(k+1)\mathbf{u} \leq 0.1. \tag{26}$$

Then $\hat{s}$ is the exact solution of a perturbed system

$$(A_D + \delta A_D)\hat{s} = f_D + \delta f_D, \tag{27}$$

where

$$\|\delta A_D\|_F \leq (1.106)(12n+51)(k+2)n\mathbf{u}\|A_D\|_F, \tag{28}$$

$$\|\delta f_D\|_2 \leq (1.106)(12n+51)(k+1)n\mathbf{u}\|f_D\|_2. \tag{29}$$

Proof. See Appendix B. ∎

THEOREM 4.2. *Suppose that the conditions of Lemma 4.1 are satisfied. Then provided that*

(i) $\bar{h}$ is chosen small enough that $\mathrm{cond}_\infty(A_{DP}) \leq 4k\kappa$,

(ii) $8c_6\kappa\mathbf{u} \leq 1$, where $c_6 = (1.106)k(k+2)^2n^2(12n+51)$,

the following relative error bound applies:

$$\frac{\|\hat{s}-s\|_\infty}{\|s\|_\infty} \leq 16c_6\kappa\mathbf{u}.$$

Suppose that the fundamental and particular solutions that are used to construct A_S and f_S in (4) are calculated to a tolerance of τ, that is,

$$\|A_S - \bar{A}_S\|_\infty \leq \tau\|A_S\|_\infty$$
$$\|f_S - \bar{f}_S\|_\infty \leq \tau\|f_S\|_\infty.$$

Let $\tilde{s}$ be the solution obtained by using structured QR algorithm, where (26) holds. Then provided that

(iii) $\tau k\kappa\gamma \leq 1/2$,

(iv) $6c_6\mathbf{u}\kappa\gamma \leq 1$,

we have

$$\frac{\max_{i=1,\ldots,k+1} \|\tilde{s}_i - y(t_i)\|_\infty}{\max_{i=1,\ldots,k+1} \|y(t_i)\|_\infty} \leq 4\gamma\kappa[k\tau + 3c_6\mathbf{u} + c_7\mathbf{u}\tau],$$

where $c_7 = 12c_6k\kappa\gamma$.

Proof. The proof is identical to that of Theorem 2.1, once we convert the Frobenius norm bounds (28) and (29) into ∞-norm bounds. This is done by noting that for an $N \times N$ matrix A,

$$\frac{1}{N^{1/2}}\|A\|_\infty \leq \|A\|_F \leq N^{1/2}\|A\|_\infty.$$

Hence

$$\|\delta A_D\|_\infty \leq (1.106)(k+2)^2n^2(12n+51)\mathbf{u}\|A_D\|_\infty.$$

■

5. Condition estimation. For purposes of assessing the reliability of the computed solution, it is useful to have an estimate of the conditioning of the discrete system. Such an estimate can be obtained, simultaneously with the factorization and solution process, by adapting the procedure described in Cline et al. [6] to our situation.

We aim to compute an estimate of the quantity

$$\hat{\kappa} = \|A\|_\infty\|R^{-1}\|_\infty,$$

where A is one of the coefficient matrices from (4),(7),(5),(8), and R is the upper triangular factor produced by the procedure just described. It is easy to show that

$$\frac{1}{\sqrt{(k+1)n}}\hat{\kappa} \leq \mathrm{cond}_\infty(A) \leq \sqrt{(k+1)n}\hat{\kappa}.$$

$\|A\|_\infty$ can of course be calculated directly, and so computation of the estimate of $\|R^{-1}\|_\infty$ is the major part the task of finding $\hat{\kappa}$. Following [6], we do this by first finding vectors z and v such that

$$R^Tv = z,$$

where the components of z are all ± 1 and are chosen by a heuristic that attempts to maximize $\|v\|_\infty$. This is done during the factorization of A; as reduction of each partition into a single row of blocks is performed, the heuristic can be applied to find the components of z and v corresponding to the rows and columns of the original system that are eliminated. Next, the solution of

$$Rw = v$$

is found concurrently with the solution of the original linear system $Rs = Q^Tf$. We then use the estimate

$$\|R^{-1}\|_\infty \approx \frac{\|w\|_\infty}{\|v\|_\infty}.$$

The operation count for calculation of $\hat{\kappa}$ is approximately four times that of doing a single backsolve with R, and the parallel complexity is the same as that of the factorization and solution. Comparisons with the LINPACK condition number estimator of $\mathrm{cond}_1(A)$ (which is based on an LU factorization of A but uses similar heuristics) show that the two estimates are usually within a factor of 3 of each other.

TABLE 1
Operation counts and storage requirements for four algorithms, assuming separated end conditions. k = number of meshpoints, n = dimension of y, p = number of left-hand end conditions, R = number of right-hand sides

Algorithm	Operation count	Storage
LU (row pivoting)	$k[\frac{5}{3}n^3 + 3pn^2 + R(4n^2 + 2pn)]$	$kn(2n + p)$
Structured QR	$k[\frac{46}{3}n^3 + (15R + 30)n^2]$	$4kn^2$
Normal equations	$k[\frac{38}{3}n^3 + 12Rn^2]$	$4kn^2$
`DECOMP/SOLVE`	$k[\frac{2}{3}n^3 + (4R + 5p)n^2 - 2np^2]$	$2kn^2$

TABLE 2
Operation counts and storage requirements for four algorithms, assuming non-separated end conditions. k = number of meshpoints, n = dimension of y, R = number of right-hand sides

Algorithm	Operation count	Storage
LU (row pivoting)	$k[\frac{23}{3}n^3 + 8Rn^2]$	$4kn^2$
Structured QR	$k[\frac{46}{3}n^3 + (15R + 30)n^2]$	$4kn^2$
Normal equations	$k[\frac{62}{3}n^3 + 12Rn^2]$	$5kn^2$
`DECOMP/SOLVE`	$k[\frac{14}{3}n^3 + 4Rn^2]$	$3kn^2$

6. Computational results. Versions of the structured QR algorithm have been implemented on the Alliant FX/8 vector multiprocessor at Argonne National Laboratory and on the CRAY Y-MP at the North Carolina Supercomputing Center. Performance comparisons were made with

- a row partial pivoting code (two versions, for separated and non-separated boundary conditions), and
- the `DECOMP` and `SOLVE` routines from the `PASVA` codes [13]. The `DECOMP` routine uses alternate row and column pivoting (as does the algorithm described in Varah [19] and in §2) but always eliminates by rows.

Approximate operation counts and storage requirements for these algorithms and the normal equations method of Ascher and Chan [1] are given in Tables 1 and 2, for separated and non-separated boundary conditions, respectively. In tabulating storage requirements, it has been assumed that intermediate information generated during the factorization — namely, the multipliers and Householder vectors — is stored for possible later use with a different right-hand side. In general, the structured QR algorithms require the most operations. This result is not surprising, since it is well known that orthogonal factorization of dense matrices requires about twice as much work as Gaussian elimination. Moreover, in the case of separated end conditions, structured QR generates fill-in that is avoided by the elimination-based methods, and this adds further to the operation count. On the other hand, the operation counts and storage requirements for strutured QR are not affected if the end conditions are non-separated rather than separated, while for the other methods, they increase substantially. The method based on normal equations requires about the same amount of work as structured QR; however, as we noted earlier, it is less stable. We add the caveat that operation counts are notoriously bad predictors of run time for factorizations of narrow-banded matrices. For small n, much of the CPU time is taken up with non-numerical operations. This is borne out by our results, which show that the QR algorithm does not do as badly as predicted in serial mode.

The linear system solvers described above have been incorporated into both a multiple shooting and finite difference (box method) framework. The `dverk` code from netlib was used to solve the IVPs on each interval of multiple shooting, with the global error tolerance parameter set to 10^{-10}. A user-specified number k of equally-spaced intervals is used for both methods. For practical codes, the choice of the number of intervals and their lengths (and efficient parallel implementation of this) are important issues, but we focus here on the linear algebra, which typically is the most computationally intensive part of a finite difference-based code. In accord with the theoretical results of §2 and §4, virtually no difference was noted between the stability properties of structured QR and row-pivoted LU. As evidence of this, we quote results for the following test problem:

Problem 1 (Ascher and Chan [1]) $a = 0$, $b = 1$, $n = 2$,

$$y'(t) = \begin{bmatrix} -\lambda\cos 2\omega t & \omega + \lambda \sin 2\omega t \\ -\omega + \lambda \sin 2\omega t & \lambda \cos 2\omega t \end{bmatrix} y(t) + f(t),$$

$$B_a = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad B_b = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix},$$

with $f(t)$ and d chosen so that $y(t) = e^t(1,1)^T$.

A fundamental solution is

$$Y(t) = \begin{bmatrix} \cos\omega t & \sin\omega t \\ -\sin\omega t & \cos\omega t \end{bmatrix} \begin{bmatrix} e^{-\lambda t} & 0 \\ 0 & e^{\lambda t} \end{bmatrix},$$

so clearly there exist one growing and one decaying mode. The problem was solved using both multiple shooting and the box method for $\lambda = 200$ and $\omega = 1$. Four different algorithms were used to solve the linear system, namely,

- `SQR-1` — one-level structured QR;
- `ROWPP` — LU factorization with row partial pivoting;
- `DECOMP` — the `DECOMP` and `SOLVE` routines from `PASVA`; and
- `COMPACT` — compactification, as implemented in the codes `D4/S4` described in [21].

Tables 3 and 4 show the maximum error in the first component of the computed result. Because of its failure to decouple the fundamental solution modes, compactification performs poorly. The accuracy of box method solutions is limited by discretization error, while the multiple shooting solutions are accurate up to the conditioning of the discrete system and the tolerance imposed on the IVP solver. The `DECOMP` code gives accurate results here because the end conditions are separated. When this is not the case, as in problem 3 below, `DECOMP` is known to be unstable.

To test the relative speed of the linear solvers, two further problems from the literature were used in addition to problem 1. These were

Problem 2 (Brown and Lorenz [5]) $a = -1$, $b = 1$, $n = 4$,

$$\begin{aligned} -\epsilon y'' - \frac{t}{2}y' + \frac{t}{2}z' + z &= \epsilon\pi^2 \cos \pi t \ + \frac{1}{2}\pi t \sin \pi t, \\ \epsilon z'' &= z \\ y(-1) = -1 \qquad & \qquad y(1) = e^{-2/\sqrt{\epsilon}} \\ z(-1) = 1 \qquad & \qquad z(1) = e^{-2/\sqrt{\epsilon}}. \end{aligned}$$

TABLE 3
Box method, error in first component of computed solution for Problem 1 for four different linear system solvers

	$k = 16$	$k = 64$	$k = 1024$
$\hat{\kappa}$	.17(+2)	.51(+1)	.21(+1)
ROWPP	.21(-2)	.10(-3)	.32(-6)
DECOMP	.22(-2)	.10(-3)	.32(-6)
SQR-1	.21(-2)	.10(-3)	.32(-6)
COMPACT	.22(-2)	.93(+27)	.16(+72)

TABLE 4
Multiple shooting, error in first component of computed solution for Problem 1 for four different linear system solvers

	$k = 16$	$k = 32$	$k = 128$
$\hat{\kappa}$	.85(+6)	.17(+4)	.13(+2)
ROWPP	.45(-3)	.67(-6)	.64(-7)
DECOMP	.45(-3)	.67(-6)	.64(-7)
SQR-1	.45(-3)	.67(-6)	.64(-7)
COMPACT	.51(+72)	.21(+72)	.11(+72)

(We use $\epsilon = .001$.)

Problem 3 (Mattheij [15]) $a = 0$, $b = \pi$, $n = 3$,

$$y'(t) = \begin{bmatrix} 1 - 19\cos 2t & 0 & 1 + 19\sin 2t \\ 0 & 19 & 0 \\ -1 + 19\sin 2t & 0 & 1 + 19\cos 2t \end{bmatrix} y(t) + e^t \begin{bmatrix} -1 + 19(\cos 2t - \sin 2t) \\ -18 \\ 1 - 19(\cos 2t + \sin 2t) \end{bmatrix},$$

$$\begin{aligned} y_1(0) &= 1 \\ y_3(0) + y_3(\pi) &= 1 + e^\pi \\ y_2(0) + y_2(\pi) &= 1 + e^\pi . \end{aligned}$$

The solution is $y(t) = e^t(1, 1, 1)^T$.

Problems 1 and 2 have separated end conditions, while two of the three end conditions for problem 3 are non-separated. We report on five cases (two different values of k were tried for problems 1 and 2, and values of $\lambda = 1$ and $\omega = 50$ were used in problem 1). Table 5 gives condition estimates for the multiple shooting and finite difference matrices.

Results from "scalar" implementations on one processor of the Alliant FX/8 are shown in Table 6. We have tabulated the times required to solve the linear systems. The `-Og` compiler option was used with each code, so the vector processing capabilities of the Alliant were not used. In addition to the linear solvers already mentioned, we tested `SQR-CR`, which was the cyclic-reduction variant of structured QR. Note that the `SQR` codes typically take two to three times as long as `ROWPP`, though the penalty is

TABLE 5
Dimensions of the five test cases, and conditioning of the multiple shooting and finite difference matrices

Problem	n	k	ms conditioning	fd conditioning
1a	2	64	.41(+2)	.17(+2)
1b	2	1024	.30(+2)	.30(+2)
2a	4	64	.37(+9)	.83(+3)
2b	4	1024	.26(+2)	.17(+2)
3	3	1024	.20(+1)	.34(+1)

TABLE 6
Alliant FX/8, one-processor timings for linear system solvers (times in seconds)

Problem	ROWPP	DECOMP	SQR-1	SQR-CR
1a	.041	.081	.071	.098
1b	.439	.660	.813	1.04
2a	.094	.181	.220	.312
2b	1.32	1.95	3.22	4.55
3	1.23	1.44	1.72	2.34

TABLE 7
CRAY Y-MP, one-processor timings for linear system solvers. Vectorized code (times in milliseconds)

Problem	ROWPP	DECOMP	SQR-1	SQR-CR
1a	2.19	3.05	8.63	1.45
1b	34.6	48.3	139.	10.9
2a	5.32	8.38	21.1	6.40
2b	84.4	133.	341.	51.6
3	136.	116.	232.	26.3

TABLE 8
Alliant FX/8, eight-processor timings for linear system solvers (times in seconds)

Problem	ROWPP	DECOMP	SQR-2
1a	.029	.072	.031
1b	.367	.697	.136
2a	.053	.150	.067
2b	.739	1.67	.463
3	.685	1.39	.262

TABLE 9
Alliant FX/8, Ratio of times for ROWPP *(one-processor) to times for* SQR-2 *(eight processors)*

Problem	Speedup
1a	1.3
1b	3.2
2a	1.4
2b	2.9
3	4.7

much smaller when the end conditions are not separated (as in problem 3). In either case, the overhead for using structured QR is not as great as the operation counts in Tables 1 and 2 would suggest.

Timings for a vectorized implementation on one processor of a CRAY Y-MP are shown in Table 7. In general, the `SQR-CR` code becomes very competitive, particularly when $n = 2$ or 3, k is large, and/or the end conditions are not separated. This code performs extremely well on problems 1b and 3. When $n = 4$ (problem 2), the small amount of vectorization that occurs in the other codes lessens the advantage of `SQR-CR`, while in problems 1a and 2a the value of k makes the overall computational task to small to benefit from vectorization.

Table 8 gives results for an eight-processor parallel implementation on the Alliant FX/8. The `-Ogc` option was used during compilation. Here, `SQR-2` refers to the two-level version of structured QR, in which the original system is broken into eight partitions of equal size, which are factorized concurrently. On the largest problem, the parallel efficiency of structured QR (measured by comparing serial `SQR-1` to parallel `SQR-2`) is 87% — quite acceptable, given that the solution of the reduced system is an unavoidable bottleneck. The efficiency improves further for still larger problems. Defining speedup as the ratio of the one-processor time for the best serial algorithm to the eight-processor time for the best parallel algorithm, we see, from Table 9, that in three of the five cases good parallel efficiency is attained. The remaining two problems were too small for parallelism to have much effect.

Comparing Tables 6 and 8, it can be seen that `ROWPP` and `DECOMP` also speed up a little when extra processors are available. This is because the Alliant is a shared-memory machine. It is important to note that on the current generation of message-passing machines, these algorithms will *not* benefit from multiprocessing unless n is large enough that rows or columns *within each block* can profitably be distributed around the processor array. This is unlikely to happen until n is at least 50 or 100. On the other hand, efficient implementations of multilevel `SQR` on these machines will be possible for much more typical problem sizes.

To summarize, we conclude that the structured QR codes are useful in the following circumstances:

- when the computational task of solving the linear equations is substantial enough to benefit from vectorization or parallelism;
- especially, when the end conditions are not separated;
- on a vector processor, when the value of n is too small (say, only 2 or 3) to allow efficient vectorized factorization of $n \times n$ blocks;
- on the current generation of distributed-memory multiprocessors, unless n is very large and the number of processors is very small;
- on a shared-memory multiprocessor, unless n is quite large (say, greater than 8) and there are fewer than four processors.

A. Appendix A. We start with a result which is similar to [8, Theorem 3.3.1]:

LEMMA A.1. *Suppose that alternate row and column elimination, without pivoting, is applied to an $N \times N$ matrix A with bandwidth b_w to produced computed factors $\hat{L}$, $\hat{B}$, $\hat{U}$. Assume that N, b_w and the unit roundoff error* $\mathbf{u}$ *satisfy*

$$N\mathbf{u} < 0.1 \tag{30}$$

$$N\mathbf{u}(2 + 1.06b_w + 2.12b_w\mathbf{u}) < 0.5. \tag{31}$$

Then

$$\hat{L}\hat{B}\hat{U} = A + H,$$

where

$$|H| \leq c_3(N-1)\mathbf{u}\{|A| + |\hat{L}||\hat{B}||\hat{U}|\} \tag{32}$$

and

$$c_3 = 5 + 3b_w.$$

Proof. The result is trivially true for $N = 1$. Suppose for induction that (30) holds for matrices of size up to $N-1$. Let

$$A = \begin{bmatrix} \alpha & w^T \\ v & A_1 \end{bmatrix},$$

where $\alpha \in R$, $A_1 \in R^{(N-1)\times(N-1)}$, etc., and suppose that row elimination will be used to eliminate v. We compute

$$\hat{z} = \tfrac{1}{\alpha}v + f, \qquad |f| \leq \tfrac{|v|}{\alpha}\mathbf{u}, \tag{33}$$

$$\hat{A}_1 = A_1 - \hat{z}w^T + F, \qquad |F| \leq 2\mathbf{u}(|A_1| + |\hat{z}||w|^T), \tag{34}$$

It follows immediately from (34) that

$$|\hat{A}_1| \leq (1 + 2\mathbf{u})(|A_1| + |\hat{z}||w|^T). \tag{35}$$

An LBU factorization of $\hat{A}_1$ is then performed, yielding

$$\hat{L}_1\hat{B}_1\hat{U}_1 = \hat{A}_1 + H_1,$$

with

$$|H_1| \leq c_3(N-2)\mathbf{u}\{|\hat{A}_1| + |\hat{L}_1||\hat{B}_1||\hat{U}_1|\}. \tag{36}$$

The calculated factors of A are therefore

$$\hat{L} = \begin{bmatrix} 1 & 0 \\ \hat{z} & \hat{L}_1 \end{bmatrix}, \qquad \hat{B} = \begin{bmatrix} \alpha & \hat{w}^T \\ 0 & \hat{B}_1 \end{bmatrix}, \qquad \hat{U} = \begin{bmatrix} 1 & 0 \\ 0 & \hat{U}_1 \end{bmatrix},$$

where $\hat{w}$ is the computed solution of the system $\hat{U}_1^T\tilde{w} = w$. Defining $\bar{b}_w = 1.06b_w$, and noting that $\hat{U}_1^T$ has lower bandwidth b_w, it is easy to show that $\hat{w}$ *exactly* solves

$$(\hat{U}_1^T + \delta_U)\hat{w} = w,$$

where

$$|\delta_U| \leq \bar{b}_w\mathbf{u}|\hat{U}_1|^T.$$

Hence

$$|\hat{U}_1^T\hat{w} - w| = |\delta_U\hat{w}| \leq |\delta_U||\hat{w}| \leq \bar{b}_w\mathbf{u}|\hat{U}_1|^T|\hat{w}|, \tag{37}$$

and so

$$|w| \leq (1+\bar{b}_w\mathbf{u})|\hat{U}_1|^T|\hat{w}|. \tag{38}$$

Now

$$\begin{aligned} \hat{A}_1 &= A_1 - \hat{z}w^T + F \\ \Rightarrow \hat{z}\hat{w}^T\hat{U}_1 + \hat{L}_1\hat{B}_1\hat{U}_1 &= A_1 + F + H_1 - \hat{z}[w - \hat{U}_1^T\hat{w}]^T. \end{aligned}$$

Combining this with (33),

$$\hat{L}\hat{B}\hat{U} = A + \begin{bmatrix} 0 & \hat{w}^T\hat{U}_1 - w^T \\ \alpha f & F + H_1 - \hat{z}[w - \hat{U}_1^T\hat{w}]^T \end{bmatrix}. \tag{39}$$

Now, combining (34), (35), (36) and (38), we find that

$$\begin{aligned} & |F + H_1 - \hat{z}[w - \hat{U}_1^T\hat{w}]^T| \\ \leq\ & |F| + |H_1| + |\hat{z}||w - \hat{U}_1^T\hat{w}|^T \\ \leq\ & [c_3(N-2)\mathbf{u}(1+2\mathbf{u}) + 2\mathbf{u}]|A_1| \\ & + [c_3(N-2)\mathbf{u}(1+2\mathbf{u})(1+\bar{b}_w\mathbf{u}) + 2\mathbf{u}(1+\bar{b}_w\mathbf{u}) + \bar{b}_w\mathbf{u}]|\hat{z}||\hat{w}|^T|\hat{U}_1| \\ & + c_3(N-2)\mathbf{u}|\hat{L}_1||\hat{B}_1||\hat{U}_1|. \end{aligned} \tag{40}$$

We now show that

$$c_3(N-2)\mathbf{u}(1+2\mathbf{u})(1+\bar{b}_w\mathbf{u}) + 2\mathbf{u}(1+\bar{b}_w\mathbf{u}) + \bar{b}_w\mathbf{u} \leq c_3(N-1)\mathbf{u}. \tag{41}$$

This is equivalent to

$$\begin{aligned} c_3(N-2)\mathbf{u}(2\mathbf{u} + \bar{b}_w\mathbf{u} + 2\bar{b}_w\mathbf{u}^2) + (2\mathbf{u} + \bar{b}_w\mathbf{u} + 2\bar{b}_w\mathbf{u}^2) &\leq c_3\mathbf{u} \\ \Leftrightarrow c_3[1 - (N-2)\mathbf{u}(2 + \bar{b}_w + 2\bar{b}_w\mathbf{u})] &\geq (2 + \bar{b}_w + 2\bar{b}_w\mathbf{u}). \end{aligned}$$

Since by assumption (31),

$$1 - (N-2)\mathbf{u}(2 + \bar{b}_w + 2\bar{b}_w\mathbf{u}) \geq 1/2,$$

inequality (41) will hold provided that

$$c_3 \geq 2(2 + \bar{b}_w + 2\bar{b}_w\mathbf{u}). \tag{42}$$

From (30), and using $b_w < N$, it is clear that $2\bar{b}_w\mathbf{u} \leq 0.5$, so (42) follows trivially.

Since the left-hand side of (41) is the largest of the three coefficients in (40), we can combine (40) and (41) to obtain

$$|F + H_1 - \hat{z}[w - \hat{U}_1^T\hat{w}]^T| \leq c_3(N-1)\mathbf{u}\{|A_1| + |\hat{z}||\hat{w}|^T|\hat{U}_1| + |\hat{L}_1||\hat{B}_1||\hat{U}_1|\}. \tag{43}$$

Combining (39), (33), (37) and (43), we therefore find that

$$\begin{aligned} & |A - \hat{L}\hat{B}\hat{U}| \\ &= \left| \begin{bmatrix} 0 & \hat{w}^T\hat{U}_1 - w^T \\ \alpha f & F + H_1 - \hat{z}[w - \hat{U}_1^T\hat{w}]^T \end{bmatrix} \right| \\ &\leq c_3(N-1)\mathbf{u}\left\{ \begin{bmatrix} |\alpha| & |w|^T \\ |v| & |A_1| \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ |\hat{z}| & |\hat{L}_1| \end{bmatrix} \begin{bmatrix} |\alpha| & |\hat{w}|^T \\ 0 & |\hat{B}_1| \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & |\hat{U}_1| \end{bmatrix} \right\} \\ &= c_3(N-1)\mathbf{u}\{|A| + |\hat{L}||\hat{B}||\hat{U}|\}. \end{aligned}$$

This proves the result for the case in which row elimination is used at stage N. When column elimination is used instead, the proof is analogous. ∎

LEMMA A.2. *If the alternate row and column elimination is used to solve (5) or (8), and the assumptions of Lemma A.1 hold, with $N = (k+1)n$ and $b_w = 2n$, then the computed solution $\hat{s}$ is the exact solution of the perturbed system*

$$(A + E)\hat{s} = f,$$

where

$$\|E\|_\infty \leq (k+2)n(5+6n)(1+n^2 2^{n-1})\|A_{DP}\|_\infty \mathbf{u}.$$

Proof. Note first that the pivoting does not alter the sparsity structure of A. We can, therefore, view alternate row and column elimination as being applied to $P^T A \Pi^T$ (where P and Π are permutation matrices) to produce a computed factorization

$$P^T A \Pi^T + H = \hat{L}\hat{B}\hat{U}.$$

It is easy to show that the procedure leading to $\hat{s}$ results in the following sequence of perturbed problems:

$$\begin{aligned}
(\hat{L} + \delta_L)\hat{w} &= P^T f, & |\delta_L| &\leq (1.06)n\mathbf{u}|\hat{L}|,\\
(\hat{B} + \delta_B)\hat{v} &= \hat{w}, & |\delta_B| &\leq 2(1.06)n\mathbf{u}|\hat{B}|,\\
(\hat{U} + \delta_U)\hat{z} &= \hat{v}, & |\delta_U| &\leq (1.06)n\mathbf{u}|\hat{U}|,\\
\hat{s} &= \Pi^T \hat{z}.
\end{aligned}$$

(The bounds on $|\delta_L|$, $|\delta_B|$, $|\delta_U|$ are a consequence of the maximum number of nonzeros in each row of $\hat{L}$, $\hat{B}$, $\hat{U}$, respectively.) Hence

$$\begin{aligned}
P^T E \Pi^T &= H + (\hat{L} + \delta_L)(\hat{B} + \delta_B)(\hat{U} + \delta_U) - \hat{L}\hat{B}\hat{U}\\
\Rightarrow \|E\|_\infty &\leq \|H\|_\infty + [4(1.06)n\mathbf{u} + 5(1.06)^2 n^2\mathbf{u}^2 + 2(1.06)^3 n^3 \mathbf{u}^3]\|\hat{L}\|_\infty \|\hat{B}\|_\infty \|\hat{U}\|_\infty.
\end{aligned}$$

It follows from (30) that $n\mathbf{u} \leq 0.1$, so the coefficient of $\|\hat{L}\|_\infty \|\hat{B}\|_\infty \|\hat{U}\|_\infty$ can be bounded above by $5n\mathbf{u}$. Since element growth in B is bounded by 2^{n-1}, and since all entries in $\hat{L}$ and $\hat{U}$ are bounded by 1, we have

$$\|\hat{B}\|_\infty \leq 2^{n-1}\|P^T A \Pi^T\|_\infty \leq 2^{n-1}\|A\|_\infty, \qquad \|\hat{L}\|_\infty \leq n, \qquad \|\hat{U}\|_\infty \leq n.$$

Combining these observations with the result of Lemma A.1, we obtain

$$\begin{aligned}
\|E\|_\infty &\leq (5+6n)(k+1)n\mathbf{u}\{\|A\|_\infty + \|\hat{L}\|_\infty \|\hat{B}\|_\infty \|\hat{U}\|_\infty\} + 5n\mathbf{u}\|\hat{L}\|_\infty \|\hat{B}\|_\infty \|\hat{U}\|_\infty\\
&\leq \{(5+6n)(k+1)n + [(5+6n)n(k+1) + 5n]n^2 2^{n-1}\}\|A\|_\infty \mathbf{u}\\
&\leq (k+2)n(5+6n)(1+n^2 2^{n-1})\|A\|_\infty \mathbf{u}.
\end{aligned}$$

as required. ∎

B. Appendix B. We start by stating two results on the rounding error due to Householder reduction. These results are similar to those in Lawson and Hanson [10, pp. 85–89] and Wilkinson [20, pp. 157–162]. They differ from Lawson and Hanson's results in that the $O(\mathbf{u}^2)$ term is explicitly accounted for at every stage, and from Wilkinson's in that we do not assume double-precision accumulation of inner products. Since the proofs are tedious and do not offer any new insight, they are omitted.

LEMMA B.1. *Suppose that an $m_1 \times m_2$ matrix X is multiplied by an $m_1 \times m_1$ Householder reflector Q. Then provided that*

$$(6m_1 + 18)\mathbf{u} \leq 0.1,$$

the computed result Y satisfies

$$Y = Q(X + E),$$

where

$$\|E\|_F \leq (7m_1 + 42)\mathbf{u}\|X\|_F.$$

LEMMA B.2. *If Q is a product of r Householder reflectors whose effect is to introduce zeros into the subdiagonals into the first r columns of the $m_1 \times m_2$ matrix X, then provided that*

$$(7m_1 + 42)r\mathbf{u} \leq 0.1,$$

the computed result Y satisfies

$$Y = Q(X + E),$$

where

$$\|E\|_F \leq (8m_1 - 4r + 51)r\mathbf{u}\|X\|_F.$$

For the purpose of this Appendix, it is simplest to view the structured factorization process as the application of $k-1$ orthogonal transformation matrices $Q_1, Q_2, \ldots, Q_{k-1}$ to a row- and column-reordered version of A_D (and the right-hand side f_D), followed by the application of another two matrices Q_k and Q_{k+1} to effect the final reduction of (25). (Q_k and Q_{k+1} reduce the first and last n columns of the coefficient matrix in (25), respectively.) Each of the Q_j, $j = 1, \ldots, k-1$ are products of n Householder reflectors, and each operates on only a small part of the matrix that it multiplies: to be precise, a $2n \times 3n$ submatrix. Since we wish to reduce (7) to (25), it follows that exactly $k-1$ such transformations are needed.

Proof. (Lemma 4.1) Let $\hat{A}_{D,i+1}$ be the transformed version of A_D after i stages of the structured factorization, and $\hat{A}_{D,1} = A_D$. Let $\hat{\hat{A}}_{D,i+1}$ be the submatrix which is actually affected at stage i, by the matrix Q_i from (22), and let $\hat{Q}_i$ be the orthogonal matrix which is obtained by embedding Q_i into a $(k+1)n$-dimensional identity matrix. For $i = 1, \ldots, k+1$, we have from Lemma B.2 that $\hat{A}_{D,i}$ and $\hat{A}_{D,i+1}$ are related by

$$\hat{A}_{D,i+1} = \hat{Q}_i(\hat{A}_{D,i} + E_i),$$

where E_i consists of the $2n \times 3n$ "error" submatrix corresponding to the factorization of $\hat{\hat{A}}_{D,i}$, padded out with zeros to dimension $(k+1)n$. Hence

$$\|E_i\|_F \leq (8(2n) - 4n + 51)n\mathbf{u}\|\hat{\hat{A}}_{D,i}\|_F \leq (12n + 51)n\mathbf{u}\|\hat{A}_{D,i}\|_F.$$

Hence

$$\|\hat{A}_{D,i+1}\|_F \leq [1 + (12n + 51)n\mathbf{u}]\|\hat{A}_{D,i}\|_F \leq [1 + (12n + 51)n\mathbf{u}]^i\|A_D\|_F.$$

The errors made in stages k and $k+1$ are bounded in the same way, since the submatrices affected at these stages are no larger than those affected at the earlier stages. Under the assumption $(12n+51)n(k+1)\mathbf{u} \leq 0.1$, we therefore obtain

$$\|\hat{A}_{D,i+1}\|_F \leq [1+(1.06)(12n+51)n(k+1)\mathbf{u}]\|A_D\|_F \leq (1.106)\|A_D\|_F,$$

for $i = 1, \ldots, k+1$. Hence

$$\|E_i\|_F \leq (1.106)(12n+51)n\mathbf{u}\|A_D\|_F.$$

The final (upper triangular) matrix is $\hat{A}_{D,k+2}$, which satisfies

$$\begin{aligned}
\hat{A}_{D,k+2} &= \hat{Q}_{k+1}(\hat{A}_{D,k+1} + E_{k+1}) \\
&= \hat{Q}_{k+1}\hat{Q}_k(\hat{A}_{D,k} + E_k) + \hat{Q}_{k+1}E_{k+1} \\
&\vdots \\
&= \hat{Q}_{k+1}\ldots\hat{Q}_1 A_D + \hat{Q}_{k+1}\ldots\hat{Q}_1 E_1 + \hat{Q}_{k+1}\ldots\hat{Q}_2 E_2 + \ldots + \hat{Q}_{k+1}E_{k+1} \\
&= \hat{Q}[A_D + E_D].
\end{aligned}$$

Here $\hat{Q} = \hat{Q}_{k+1}\hat{Q}_k\ldots\hat{Q}_1$ is orthogonal, and E_D satisfies the bound

$$\|E_D\|_F \leq \sum_{i=1}^{k+1}\|E_i\|_F = (1.106)(12n+51)n(k+1)\mathbf{u}\|A_D\|_F.$$

Similarly, application of $\hat{Q}$ to the right-hand side f_D results in a computed vector $\hat{f}_D$ which satisfies

$$\hat{f}_D = \hat{Q}[f_D + \delta f_D],$$

where

$$\|\delta f_D\|_2 \leq (1.106)(12n+51)n(k+1)\mathbf{u}\|f_D\|_2$$

(since $\|v\|_F = \|v\|_2$ when v is a vector.) Finally, back-substitution is used on the system with coefficient matrix $\hat{A}_{D,k+2}$ and right-hand side $\hat{f}_D$. The computed solution satisfies

$$(\hat{A}_{D,k+2} + E_S)\hat{s} = \hat{f}_D, \tag{44}$$

where, since $\hat{A}_{D,k+2}$ has at most $3n$ nonzeros per row,

$$\begin{aligned}
\|E_S\|_F &\leq 3(1.06)n\mathbf{u}\|\hat{A}_{D,k+2}\|_F \\
&\leq 3(1.06)n\mathbf{u}[1+(1.106)(12n+51)n(k+1)\mathbf{u}]\|A_D\|_F \\
&\leq 3(1.06)(1.1106)n\mathbf{u}\|A_D\|_F.
\end{aligned}$$

Substituting in (44),

$$\begin{aligned}
\left[\hat{Q}[A_D + E_D] + E_S\right]\hat{s} &= \hat{Q}[f_D + \delta f_D] \\
\Rightarrow (A_D + E_D + \hat{Q}^T E_S)\hat{s} &= (f_D + \delta f_D).
\end{aligned}$$

Defining

$$\delta A_D = E_D + \hat{Q}^T E_S,$$

we have

$$\begin{aligned} \|\delta A_D\|_F &\leq \|E_D\|_F + \|E_S\|_F \\ &\leq [(1.106)(12n+51)n(k+1) + 3(1.06)(1.1106)n]\,\mathbf{u}\|A_D\|_F \\ &\leq (1.106)(12n+51)n(k+2)\mathbf{u}\|A_D\|_F, \end{aligned}$$

as required. ∎

REFERENCES

[1] U. M. Ascher and P. S. Y. Chan, *On parallel methods for boundary value odes*, Computing, 46 (1991), pp. 1–17.

[2] U. M. Ascher and R. M. M. Mattheij, *General framework, stability and error analysis for numerical stiff boundary value problems*, Numerische Mathematik, 54 (1988), pp. 355–372.

[3] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, 1988.

[4] Z. Bohte, *Bounds for rounding errors in the Gaussian elimination for band systems*, Journal of the Institute of Mathematics and its Applications, 16 (1975), pp. 133–142.

[5] D. L. Brown and J. Lorenz, *A high order method for stiff boundary value problems with turning points*, SIAM Journal on Scientific and Statistical Computing, 8 (1987), pp. 790–805.

[6] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson, *An estimate for the condition number of a matrix*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 368–375.

[7] J. C. Diaz, A. Fairweather, and P. Keast, *FORTRAN packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination*, ACM Transactions on Mathematical Software, 9 (1983), pp. 358–375.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, second ed., 1989.

[9] H. B. Keller, *Accurate difference methods for two-point boundary value problems*, SIAM Journal on Numerical Analysis, 11 (1974), pp. 305–320.

[10] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[11] M. Lentini, *Parallel solution of special large block tridiagonal systems: Tpbvp*. Manuscript, 1989.

[12] M. Lentini, M. R. Osborne, and R. D. Russell, *The close relationships between methods for solving two-point boundary value problems*, SIAM Journal on Numerical Analysis, 22 (1985), pp. 280–309.

[13] M. Lentini and V. Pereyra, *An adaptive finite difference solver for nonlinear two-point boundary value problems with mild boundary layers*, SIAM Journal on Numerical Analysis, 14 (1977), pp. 91–111.

[14] R. M. M. Mattheij, *The conditioning of linear boundary value problems*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 963–978.

[15] ———, *Decoupling and stability of algorithms for boundary value problems*, SIAM Review, 27 (1985), pp. 1–44.

[16] M. R. Osborne, *Aspects of the numerical solution of boundary value problems with separated boundary conditions*. Manuscript, 1978.

[17] M. Paprzycki and I. Gladwell, *Solving almost block diagonal systems on parallel computers*, Parallel Computing, 17 (1991), pp. 133–153.

[18] V. Pereyra, *Iterated deferred corrections for nonlinear boundary value problems*, Numerische Mathematik, 8 (1968), pp. 111–125.

[19] J. M. Varah, *Alternate row and column elimination for solving certain linear systems*, SIAM Journal on Numerical Analysis, 13 (1976), pp. 71–75.

[20] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

[21] S. J. Wright and V. Pereyra, *Adaptation of a two-point boundary value problem solver to a vector-multiprocessor environment*, SIAM Journal on Scientific and Statistical Computing, 11 (1990), pp. 425–449.